이학박사 학위논문

Deep Learning from Structured Data

구조적 자료의 심층 학습

2020년 8월

서울대학교 대학원

통계학과

최 영 원

이학박사 학위논문

Deep Learning from Structured Data

구조적 자료의 심층 학습

2020년 8월

서울대학교 대학원

통계학과

최 영 원

Deep Learning from Structured Data

구조적 자료의 심층 학습

지도교수 원 중 호

이 논문을 이학박사 학위논문으로 제출함

2020 년 5 월

서울대학교 대학원

통계학과

최 영 원

최영원의 이학박사 학위논문을 인준함

2020 년 7 월

위 원 장	Myunghee Cho Paik	(인)
부위원장	원중호	(인)
위 원	장원철	(인)
위 원	김용대	(인)
위 원	한재준	(인)

Abstract

Youngwon Choi
Department of Statistics
The Graduate School
Seoul National University

When data possess some structure, a framework implementing the known structures of data can alleviate prominent challenges of deep learning such as robustness, generalizability, and explainability. This dissertation proposes deep learning frameworks for structured data in two tasks. The first task is to develop a representation learning model to simulate nested data. For example, the VGGFace2 dataset consists of more than 300 portraits per person on average. Interpreting such data with a nested structure as i.i.d. observations of a random process provides a fruitful viewpoint on disentangling representations. In this point of view, this thesis proposes the Ornstein auto-encoder (OAE), a promising new family of models for representation learning when data have a nested structure. The key attraction of OAE is its ability to generate samples nested within an observational unit, even if the unit is unknown to the model. This feature distinguishes OAE from conditional models. Furthermore, when the data exhibit exchangeability, OAE's reparametrization of Ornstein's d-bar distance, an infinite-dimensional optimal transport distance on which the OAE framework lies, produces a tractable learning algorithm. OAE has successfully demonstrated high performance in the three types of tasks that have been advocated in assessing the quality of generative models, namely exemplar generation, style transfer, and unit generation. This performance implies that

the framework using the structures of data can handle the generalizability issues of deep learning.

The second part of this dissertation includes a study for learning a predictive model for capturing a hierarchical correlation in microbiome taxonomic abundance data. Since bacteria are classified at a hierarchy of taxonomic levels, microbiome abundance data have a hierarchical correlation structure. `DeepBiome` is a deep-neural-network-based predictive model for capturing microbiome signals at different phylogenetic depths. By leveraging the phylogenetic information, `DeepBiome` relieves the heavy burden of tuning for the optimal deep learning architecture, avoids overfitting, and most importantly enables visualizing the path from microbiome counts to disease. The second part contributes to the development of the software for `DeepBiome`. Comprehensive simulation experiments have demonstrated the ability of the software. The `DeepBiome` model trained with the developed software shows better generalizability than other deep learning models. For both regression and classification tasks, compared to sparse regression and other deep learning models, `DeepBiome` has competitive performance particularly when microbiome taxa associated with the outcome are clustered at different phylogenetic levels. More importantly, `DeepBiome` enables an explainable visualization of the microbiome-phenotype association network. In real-life data analysis, `DeepBiome` software shows the ability to train a high-performance predictive model and select taxa that are related to the disease according to previous clinical research.

Keywords: deep learning, representation learning, structured data, face generation, identity-preserving, microbiome, phylogenetic tree, mixed taxonomic levels

Student Number: 2016-30094

Contents

List of Figures

List of Tables

Chapter 1

Introduction

A deep learning framework implementing the known structures of data can alleviate the prominent challenges such as robustness, generalizability, and explainability. This dissertation proposes deep learning frameworks for structured data in two tasks. The first task is to develop a representation learning model to simulate *nested* data, i.e., data collected from grouped observational units. The second task is to learn a predictive model for capturing a hierarchical correlation in the covariates.

1.1 Representation learning from nested data

Many real-world data are collected in grouped observation units. The resulting sample naturally possesses a nested structure. As a concrete example, consider the VGGFace2 dataset (Cao et al., 2018), an expansion of the famous VGGFace dataset (Parkhi et al., 2015). VGGFace2 is a large-scale face dataset containing 3.31 million images of 9,131 people. For each person, it contains 362.6 images

on average, with a minimum of 30 and a maximum of 843. Unquestionably, portrayals of the same person are highly correlated. Likewise, the images from another famous MNIST dataset also retain correlated structure for each digit. For such nested data, representation learning aims to find a disentangled representation that can effectively describe the correlation structure of the data. Observational units should be well-separated in the representation space. Also, the model must deal with an unknown total number of observational units when they are randomly sampled, including the case where the number of observational units is not bounded. In the VGGFace2 data, for example, the identity of a portrait can be considered an observational unit. The number of these units is possibly infinite, and the available sample may not include all the units. The structure that a representation learning method captures can be demonstrated by observational-unit-preserving sample generation. Three types of tasks have been advocated to assess the quality of a generative model (Lake et al., 2019): 1) exemplar generation, which conditionally generates new samples given observations of a new unit, 2) style transfer, which transfers the variation within a given observation unit to another unit, and 3) unit generation, which is to simulate a new observational unit.

Generative latent variable models (LVMs) such as the generative adversarial networks (GAN, Goodfellow et al., 2014) and the Wasserstein auto-encoders (WAE, Tolstikhin et al., 2018) have delivered promising outcomes. However, these approaches are not designed for structured data and, thus cannot generate correlated samples from a specific unit. InfoGAN (Chen et al., 2016) mitigates this problem by introducing additional latent codes that have high mutual information within the generated sample. While this approach can generate correlated samples within each latent code, one cannot determine the units (codes) since the method is designed in a totally unsupervised manner.

Class-conditional approaches like conditional adversarial auto-encoders (CAAE, Makhzani et al., 2016) can be thought of as learning an LVM for each conditional distribution, thereby enabling within-observational-unit sample generation. This approach, however, assumes a fixed number of units. Thus, in exemplar generations and style-transfer tasks, class-conditional approaches cannot generate such samples from new units unknown to the model.

To obtain a single representation that addresses all the three tasks, a more sensible approach is to model the nested structure in the latent space directly, and find appropriate mappings between them. For example, the random intercept model (Diggle et al., 2002; Fitzmaurice et al., 2012; Dundar et al., 2007) is a common approach in statistics when there are correlations among observations within a unit:

$$Z_j^i = B^i + E_j^i, \quad B^i \overset{\text{i.i.d.}}{\sim} \mathcal{N}(0, \tau_0^2 \mathbf{I}), \quad E_j^i \overset{\text{i.i.d.}}{\sim} \mathcal{N}(0, \sigma_0^2 \mathbf{I}), \quad B^i \perp\!\!\!\perp E_j^i, \tag{1.1}$$

where Z_j^i denotes the jth observation in unit i. Each unit is represented by the random intercept B^i. Clearly, observed sequence from unit i, i.e., $(z_j^i)_{j=0}^{n_i}$, are correlated. Differing numbers of samples between units are also naturally handled. A noticeable feature of model (1.1) is that it defines an *exchangeable* sequence: within a unit, the order of observation does not matter. Thus the nested data can be considered an independent, identically distributed (i.i.d.) copy of the exchangeable sequence.

Interpreting data with a nested structure as i.i.d. observations of a random process, or, moreover, those of an exchangeable random process, provides a fruitful viewpoint on disentangling representations. For both VGGFace2 and MNIST data, permuting the order of portraits in each person or handwritings in each digit does not notably affect any learning tasks commonly undertaken,

so we can see them as exchangeable sequences.

The goal of this task is to bring the nested data structure that arises from various applications down to generative latent variable modeling. If the latent variables share the nested structure of the observed variables, then the generative power of the latent space representation is likely to increase. As discussed above, this nesting often translates to i.i.d. observations of a correlated random process. Then, we can use the optimal transport distance between stationary random processes to seek a latent space representation of the observed sequences.

Contributions The main contribution of the first part of this thesis is to propose the Ornstein auto-encoder (OAE), a new family of models for representation learning when data have a nested structure, that is presented in Choi and Won (2019). The contributions are as follows:

- Ornstein auto-encoder (OAE), which can be thought of as a stationary random process version of the Wasserstein auto-encoder (WAE) (Tolstikhin et al., 2018), is proposed by introducing Orstein's d-bar distance, an infinite-dimensional optimal transport distance on which the OAE framework lies. To this, a stationary process version of the theorem by Bousquet et al. (2017) is proved.

- When the data exhibit exchangeability, OAE provided a tractable learning algorithm.

- High-performance of OAE is demonstrated in the three types of tasks that have been advocated for assessing the quality of generative models: exemplar generation, style transfer, and unit generation. Importantly, OAE is robust to data imbalance and can generate samples nested within an

observational unit, even if the unit is unknown to the model, which has been impossible with other methods.

- It is demonstrated that OAE can provide disentangled representations, i.e., latent variables are well-clustered by subjects. This capability has potential applications in classification and recognition.

1.2 Learning a predictive model for capturing a hierarchical correlation

The latter part of this dissertation includes a study for learning a predictive model to capture a hierarchical correlation in microbiome taxonomic abundance data. Microbiome profile can be a novel predictive biomarker for many diseases (Sartor, 2008; Gilbert et al., 2016; Ni et al., 2017; Franzosa et al., 2019). However, bacteria count tables are typically sparse and bacteria are classified at a hierarchy of taxonomic levels, ranging from species to phylum. For such hierarchical data, it is challenging to infer and visualize the bacteria-to-disease path across taxonomic levels.

Multiple approaches have been proposed to incorporate the phylogenetic structure into the analysis. For example, Garcia et al. (2013) proposed a sparse regression model using ℓ_1 and ℓ_2 regularizations to achieve sparsity at multiple taxonomic levels. However, this approach can only select taxa at up to three levels and therefore cannot cover the entire range of phylogenetic depths. Xiao et al. (2018) have developed a generalized linear mixed model that used the evolutionary rate as a tuning parameter. This approach can identify clustered signals without prior knowledge of phylogenetic depth. However, it is not desirable when microbiome effects are clustered at a mixture of taxonomic levels. Reiman et al. (2017) embedded the phylogenetic tree into a convolutional neural

network (CNN) architecture to predict the outcome of interest. In that study, the embedding algorithm translated taxa abundances at every phylogenetic level into an abundance matrix. This abundance matrix captured the spatial information of taxa in the phylogenetic tree. However, after embedding, only a fixed number of layers (three) were used and the taxa lineage information is lost. The identified neurons do not have any biological meaning and the result lacks interpretability.

The second part contributes to the development of the software for `DeepBiome`, a deep-neural-network-based predictive model, for capturing microbiome signals at different phylogenetic depths. This model is applicable to both regression and classification problems. It takes microbiome taxonomic abundance data as input. By regularizing the neural network architecture towards the phylogenetic structure, `DeepBiome` greatly reduces the number of parameters and tuning burden compared to the conventional neural networks. It is able to identify important taxa that are associated with outcomes at all taxonomic levels. Phylogeny regularization is achieved by weight decay, a popular technique (Mundie and Massengill, 1991; Krogh and Hertz, 1992; Gupta and Lam, 1998) to prevent overfitting and boost performances of deep neural networks (DNNs) (Zhang et al., 2019). Currently, all existing weight decay schemes assume a global rate of decay. `DeepBiome` instead incorporates the bacteria evolutionary relationship into a differential weight decay regularization matrix, thus generating an interpretable effect transfer network in modeling and analyzing microbiome data. Simulation studies and analyses using the datasets from the American Gut Project (AGP) and a lung microbiome study demonstrate the superior performance of `DeepBiome` over commonly used tools , including support vector machine (SVM), regression with ℓ_1 (lasso) or $\ell_1 + \ell_2$ (elastic net) penalties, DNN without tree regularization, and DNN with ℓ_1 penalty.

Contributions The main contributions of the second part are:

- `DeepBiome` software, a Tensorflow- and Keras-based open-source Python package using GPUs, which has the following features:
  - Adam optimization method with phylogenetic tree weight decay regularizer,
  - User-friendly interface for training, testing, and taxa selection with huge data such as the American Gut Project (AGP) (McDonald et al., 2018),
  - Automated visualization tool that can show the selected taxa on the phylogenetic tree based on the trained weight to infer the microbiome-disease path,
- Simulation studies, which demonstrate the ability of the `DeepBiome` software.

1.3 Outline of the thesis

This thesis is organized as follows. Part 1 proposes a representation learning model for nested data. Chapter 2 introduces Ornstein's d-bar distance and development of OAE for stationary and exchangeable processes, respectively. Chapters 3 and 4 suggest two latent variable models implementing the structured data. Variational formulations of the OAE and the algorithms derived from latent variable models are introduced and discussed in both chapters, respectively. Part 2 describes learning a predictive model for capturing a hierarchical correlation in the covariates. This part mainly introduces the `DeepBiome` method, a DNN-based predictive model for capturing microbiome signals at different phylogenetic depths. `DeepBiome` software and accompanying the sim-

ulation studies that demonstrate the ability of the software are described in Chapter 5.

Part I

Representation Learning from Nested Data

Chapter 2

Ornstein Auto-Encoders

This chapter proposes Ornstein auto-encoder (OAE), a LVM that employs a distance measure between two stationary random processes. After fixing notations and a brief review of LVMs, Ornstein's d-bar distance, an infinite-dimensional optimal transport distance on which the OAE framework lies, is introduced.

2.1 Notation

The spaces of observable variables and latent variables are denoted as $\mathcal{X}$ and $\mathcal{Z}$, respectively. Both spaces are assumed to be complete, separable metric spaces in which (regular) conditional distributions are well-defined. The metric associated with $\mathcal{X}$ is denoted d. A Cartesian product space of $\mathcal{X}$ is denoted by $\mathcal{X}^n$ for $n = 1, 2, \ldots$, with $n = \infty$ permitted, where $\mathcal{X}^\infty = \{(\ldots, x_{-1}, x_0, x_1, \ldots) : x_j \in \mathcal{X}\}$. With the Boral σ-field and a probability measure on $\mathcal{X}$ (resp. $\mathcal{Z}$), event space and probability measure on any product space, including $\mathcal{X}^\infty \times \mathcal{Z}^\infty$, are well-defined. (Regular) conditional distributions related to random variables

(processes) defined on these probability spaces are also well-defined. (Event spaces are omitted unless necessary.) Capital letters (e.g., X) indicate random variables, and their realizations are noted in lower case letters (e.g., x). Doubly-infinite random processes and their realizations are denoted by boldfaces: e.g., $\mathbf{X}$ and $\mathbf{x}$. Superscripts, as in $\mathbf{X}^i$ (resp. X^i), are used to indicate an (ith) i.i.d. copy of $\mathbf{X}$ (resp. X). Subscripts are used to represent coordinates of a random process, e.g., $\mathbf{X} = (X_1, X_2, \cdots) \in \mathcal{X}^\infty$. A finite-length random sequence is denoted as $\mathbf{X}_{1:n} = (X_1, \cdots, X_n)$. The probability distribution of random process $\mathbf{X}$ is denoted by $P_{\mathbf{X}}$, etc.; we use Q in place of P if the distribution is subject to optimization.

2.2 Preliminaries

Generative latent variable models (LVMs) are a family of parametric models trained to transform samples drawn from an unknown distribution P_X on $\mathcal{X}$ to latent variables in a lower-dimensional space $\mathcal{Z}$. In many real-world data, especially images, we cannot estimate the density of P_X, which may not exist because the distribution is supported by low-dimensional manifolds. To overcome this problem, LVMs define a latent random variable $Z \in \mathcal{Z}$ with a prior distribution P_Z such as the standard Gaussian. The latent variable model $P_{Y,Z} = Q_{Y|Z} P_Z$ can be learned by

$$\inf_{\substack{Q_{Y|Z} \in \mathcal{G} \\ P_Y = \int_{\mathcal{Z}} Q_{Y|Z} dP_Z}} \mathcal{D}(P_X, P_Y), \tag{2.1}$$

Here, $\mathcal{G}$ is some set of conditional distributions (decoders) $Q_{Y|Z}$, and $\mathcal{D}$ is an objective function which can reduce some divergence measure between the distributions P_X of observations and P_Y of their reconstruction. Different choices

of $\mathcal{D}$ and regularizer yield different models. This chapter focuses on LVMs with deterministic decoders. Let $\mathcal{G}_{\text{det}}$ be the set of deterministic decoders $Q_{Y|Z}$ such that $Q_{Y|Z}(\cdot|z)$ is the Dirac measure on $g(z)$ for all z, i.e. $Y = g(Z)$ a.s. for $g \in \mathcal{NN}$ where $\mathcal{NN}$ is the set of functions $g : \mathcal{Z} \to \mathcal{X}$ that can be parametrized by a neural network. If we assume $\mathcal{G} = \mathcal{G}_{\text{det}}$, then LVM seeks

$$\inf_{g \in \mathcal{NN}} \mathcal{D}(P_X, P_Y; g, P_Z) \tag{2.2}$$

where $P_Y = \int_{\mathcal{Z}} Q_{Y|Z} dP_Z = P_Z g^{-1}$ is the marginal distribution of Y induced by g and P_Z. There exist several famous works for training generative latent variable models. Generative adversarial networks (GAN, Goodfellow et al., 2014) has led this field. Recently, Wasserstein generative adversarial networks (WGAN, Arjovsky et al., 2017) and Wasserstein auto-encoders (WAE, Tolstikhin et al., 2018) based on optimal transport (OT) have demonstrated performances surpassing other existing LVMs.

Generative adverserial networks (GAN). Goodfellow et al. (2014) use

$$\mathcal{D}_{\text{GAN}}(P_X, P_Y; g, P_Z) \triangleq \sup_{f \in \mathcal{F}_{NN}} \mathrm{E}_{P_X} \log f(X) + \mathrm{E}_{P_Z} \mathrm{E}_{Q_{Y|Z}} \log(1 - f(Y)),$$

where $Q_{Y|Z} \in \mathcal{G}_{\text{det}}$. Here, $f : \mathcal{X} \to (0, 1)$ is the "discriminator", $\mathcal{F}_{NN}$ is the set of functions $f : \mathcal{X} \to (0, 1)$ that can be parametrized by a neural network. Thus problem (2.2) becomes a minimax game between f and g. It is known that minimizing $\mathcal{D}_{\text{GAN}}$ is equivalent to minimizing the Jensen-Shannon divergence (Goodfellow et al., 2014).

Wasserstein GAN (WGAN). Arjovsky et al. (2017) use

$$\mathcal{D}_{\mathrm{WGAN}}(P_X, P_Y; g, P_Z) \triangleq \sup_{f \in \mathcal{F}_{NN,1}} \mathrm{E}_{P_X} f(X) - \mathrm{E}_{P_Z} \mathrm{E}_{Q_{Y|Z}} \log(1 - f(Y)).$$

where $Q_{Y|Z} \in \mathcal{G}_{\mathrm{det}}$ and $\mathcal{F}_{NN,1}$ is the set of 1-Lipschitz functions $f : \mathcal{X} \to \mathbb{R}$ which can be parameterized by a neural network. This objective function is the dual formulation of the 1-Wasserstein distance between X and Y. The p-Wasserstein distance is defined as

$$\bar{d}_p(P_X, P_Y) \triangleq \left(\inf_{\pi \in \mathcal{P}(P_X, P_Y)} \mathrm{E}_\pi d^p(X, Y) \right)^{\min(1, 1/p)} \tag{2.3}$$

for some metric d defined on $\mathcal{X}$ and $p \geq 0$; d^0 represents the 0-1 loss. Here, $\mathcal{P}(P_X, P_Y)$ is the set of all joint distributions of (X, Y) whose marginals on X and Y are P_X and P_Y, respectively (Villani, 2008).

Adversarial auto-encoders (AAE). Makhzani et al. (2016) use

$$\begin{aligned} \mathcal{D}_{\mathrm{AAE}}(P_X, P_Y; g, P_Z) = &\inf_{Q_{Z|X} \in \mathcal{Q}} \mathrm{E}_{P_X} \mathrm{E}_{Q_{Z|X}} \mathrm{E}_{Q_{Y|Z}} \|X - Y\|_2^2 \\ &+ \mathcal{D}_{\mathrm{GAN}}(P_Z, \int_{\mathcal{X}} Q_{Z|X} dP_X), \end{aligned} \tag{2.4}$$

where $Q_{Y|Z} \in \mathcal{G}_{\mathrm{det}}$. The $\mathcal{Q}$ is the set of all conditional distributions $Q_{Z|X}$.

Conditional adversarial autoencoder (CAAE). CAAE minimizes a class-conditional version of (2.4),

$$\begin{aligned} \mathcal{D}_{\mathrm{cAAE}}(P_{X|C}, P_{Y|C}; g, P_{Z|C}) = &\inf_{Q_{Z|X,C}} \mathrm{E}_{P_{X|C}} \mathrm{E}_{Q_{Z|X,C}} \mathrm{E}_{Q_{Y|Z,C}} \|X - Y\|_2^2 \\ &+ \mathcal{D}_{\mathrm{GAN}}(P_{Z|C}, Q_{Z|C}), \end{aligned}$$

where $Q_{Y|Z,C} \in \mathcal{G}_{\text{det}}$ and C is a class label. The $\mathcal{Q}_C$ is the set of all conditional distributions $Q_{Z|X,C}$ and $P_{Y|C} = \int_{\mathcal{Z}} Q_{Y|Z,C} dP_{Z|C}$.

Wasserstein auto-encoders (WAE). Tolstikhin et al. (2018) directly utilizes the primal formulation (2.3) of the p-Wasserstein distance $\bar{d}_p$ through its pth power

$$\mathcal{D}_{\text{WAE}}(P_X, P_Y) \triangleq \inf_{\pi \in \mathcal{P}(P_X, P_Y)} \mathrm{E}_{\pi} d^p(X, Y). \tag{2.5}$$

for $p \geq 1$. By using Theorem 1 of Bousquet et al. (2017), they reparametrize (2.5) in terms of the probabilistic encoder $Q_{Z|X}$:

$$\mathcal{D}_{\text{WAE}}(P_X, P_Y; g, P_Z) = \inf_{Q_{Z|X} \in \mathcal{Q}_Z} \mathrm{E}_{P_X} \mathrm{E}_{Q_{Z|X}} \mathrm{E}_{Q_{Y|Z}} d^p(X, Y), \tag{2.6}$$

where $Q_{Y|Z}$ is in $\mathcal{G}_{\text{det}}$, and $\mathcal{Q}_Z$ is the set of all conditional distributions $Q_{Z|X}$ such that $Q_Z \triangleq \int Q_{Z|X} dP_X$ equals to the given P_Z. In practice, the resulting constrained optimization problem is relaxed to an unconstrained problem:

$$\inf_{g} \inf_{Q_{Z|X}} \mathrm{E}_{P_X} \mathrm{E}_{Q_{Z|X}} \mathrm{E}_{Q_{Y|Z}} d^p(X, Y) + \lambda \mathcal{D}_Z(P_Z, Q_Z) \tag{2.7}$$

for $\lambda > 0$ and some optimization function $\mathcal{D}_Z$ that can minimize the associated divergence measure. Relaxation (2.7) of WAE is equivalent to the adversarial auto-encoders (AAE) (Makhzani et al., 2016) if $p = 2$, $\mathcal{X}$ is Euclidean, $d(x, y) = \|x - y\|$ is the standard Euclidean norm, and $\mathcal{D}_Z$ is $\mathcal{D}_{\text{GAN}}$. The min-min formulations of AAE and WAE may lead to more stable training than those based on a minimax game. The potential of the WAE is reported with results on a benchmark disentanglement task (Rubenstein et al., 2018a).

2.3 Ornstein's d-bar distance

Suppose that $\mathbf{X}, \mathbf{Y}$ are two stationary processes in $\mathcal{X}^\infty$. Let

$$\bar{\rho}_n(P_{\mathbf{X}_{1:n}}, P_{\mathbf{Y}_{1:n}}) \triangleq \inf_{\pi \in \mathcal{P}(P_{\mathbf{X}_{1:n}}, P_{\mathbf{Y}_{1:n}})} E_\pi \rho_n(\mathbf{X}_{1:n}, \mathbf{Y}_{1:n})$$

where $\mathcal{P}(P_{\mathbf{X}_{1:n}}, P_{\mathbf{Y}_{1:n}})$ is the set of all joint distributions of sequences $(\mathbf{X}_{1:n}, \mathbf{Y}_{1:n}) \in \mathcal{X}^n \times \mathcal{X}^n$ having $P_{\mathbf{X}_{1:n}}$ and $P_{\mathbf{Y}_{1:n}}$ as marginals, and

$$\rho_n(\mathbf{X}_{1:n}, \mathbf{Y}_{1:n}) \triangleq n^{-1} \sum_{j=1}^{n} d^p(X_j, Y_j)$$

for some metric d defined on $\mathcal{X}$ and $p \geq 0$; d^0 represents the 0-1 loss. Ornstein's d-bar distance for random processes is defined as

$$\bar{d}_p(P_{\mathbf{X}}, P_{\mathbf{Y}}) \triangleq \bar{\rho}^{\min(1,1/p)}(P_{\mathbf{X}}, P_{\mathbf{Y}}), \tag{2.8}$$

where

$$\bar{\rho}(P_{\mathbf{X}}, P_{\mathbf{Y}}) \triangleq \sup_n \bar{\rho}_n(P_{\mathbf{X}_{1:n}}, P_{\mathbf{Y}_{1:n}})$$

(Ornstein, 1973; Gray et al., 1975). The $\bar{d}_p$ or d-bar distance for random processes was introduced by Ornstein (1973) for the special case of $p = 0$ and discrete $\mathcal{X}$, and was extended to $p \geq 0$ with complete and separable $\mathcal{X}$ by Gray, Neuhoff, and Shields (1975). Note that this is a random process version of the p-Wasserstein distance (see, e.g., Bousquet et al., 2017). Gray et al. (1975) show that the d-bar distance is a true distance for all possible stationary processes in $\mathcal{X}^\infty$, and furthermore, the equality

$$\bar{\rho}(P_{\mathbf{X}}, P_{\mathbf{Y}}) = \inf_{\pi \in \mathcal{P}_s(P_{\mathbf{X}}, P_{\mathbf{Y}})} \mathrm{E}_\pi d^p(X_0, Y_0) \tag{2.9}$$

holds, where $\mathcal{P}_s(P_{\mathbf{X}}, P_{\mathbf{Y}})$ is the set of all distributions of jointly stationary processes $(\mathbf{X}, \mathbf{Y}) \in \mathcal{X}^\infty \times \mathcal{X}^\infty$ having $P_{\mathbf{X}}$ and $P_{\mathbf{Y}}$ as marginals; X_0 (resp. Y_0) is identically distributed to X_j (resp. Y_j), $j = 1, 2, \cdots$. It is also shown that

$$\bar{\rho}_1(P_{X_0}, P_{Y_0}) = \inf_{\pi \in \mathcal{P}(P_{X_0}, P_{Y_0})} \mathrm{E}_\pi d^p(X_0, Y_0) \leq \bar{\rho}(P_{\mathbf{X}}, P_{\mathbf{Y}}), \tag{2.10}$$

where equality holds if both $P_{\mathbf{X}}$ and $P_{\mathbf{Y}}$ are i.i.d. Here, $\mathcal{P}(P_{X_0}, P_{Y_0})$ is a set of all joint distributions on $(X_0, Y_0) \in \mathcal{X} \times \mathcal{X}$ each having P_{X_0} and P_{Y_0} as marginals. Thus $\bar{\rho}_1(P_{X_0}, P_{Y_0})$ is equal to (2.3).

2.4 Ornstein auto-encoders

2.4.1 From Ornstein's d-bar distance to OAE

If Ornstein's d-bar distance or $\bar{\rho}(P_{\mathbf{X}}, P_{\mathbf{Y}})$ is employed in place of $\mathcal{D}(P_X, P_Y)$, then the resulting LVM may be called an Ornstein auto-encoder (OAE). OAE defines a latent random process $\mathbf{Z} \in \mathcal{Z}^\infty$ with prior distribution $P_{\mathbf{Z}}$ and learns a conditional distribution $Q_{\mathbf{Y}|\mathbf{Z}}$ of the reconstructed process $\mathbf{Y} \in \mathcal{X}^\infty$ given $\mathbf{Z}$ by reducing the d-bar distance under the constraint $P_{\mathbf{Y}} = \int Q_{\mathbf{Y}|\mathbf{Z}} dP_{\mathbf{Z}}$. Clearly, the WAE is a special case of OAE for an i.i.d. sequence.

The optimization problem involved with (2.9) is intractable, however, since it has to deal with the set of all jointly stationary distributions of infinite-length random processes. Thus, it is required to restrict the decoder to be a deterministic function of $\mathbf{Z}$ and reparametrize $\bar{\rho}(P_{\mathbf{X}}, P_{\mathbf{Y}})$ in terms of probabilistic "encoder" $Q_{\mathbf{Z}|\mathbf{X}}$, in much of the same way as the finite-dimensional counterpart WAE (Tolstikhin et al., 2018). From the resemblance of equation (2.9) to the finite-dimensional Wasserstein metric (2.3), a reparametrization similar to (2.6) can be made:

Theorem 1. *Assume process distributions $P_{\mathbf{X}}$ on $\mathcal{X}^\infty$ and $P_{\mathbf{Z}}$ on $\mathcal{Z}^\infty$ are both stationary. Also assume that $Q_{\mathbf{Y}|\mathbf{Z}}(\cdot|\mathbf{z})$ is the Dirac measure on $\mathbf{g}(\mathbf{z})$ for all $\mathbf{z}$, i.e., $\mathbf{Y} = \mathbf{g}(\mathbf{Z})$ with a.s. for $\mathbf{g} : \mathcal{Z}^\infty \to \mathcal{X}^\infty$ that maps a stationary sequence to a stationary sequence and is measurable. Then,*

$$\bar{\rho}(P_{\mathbf{X}}, P_{\mathbf{Y}}; \mathbf{g}, P_{\mathbf{Z}}) = \inf_{Q_{\mathbf{Z}|\mathbf{X}} \in \mathcal{Q}_{\mathbf{Z}|\mathbf{X}}} \mathrm{E}_{P_{\mathbf{X}}} \mathrm{E}_{Q_{\mathbf{Z}|\mathbf{X}}} \mathrm{E}_{Q_{\mathbf{Y}|\mathbf{Z}}} d^p(X_0, Y_0),$$

where $P_{\mathbf{Y}} = \int Q_{\mathbf{Y}|\mathbf{Z}} dP_{\mathbf{Z}} = P_{\mathbf{Z}}\mathbf{g}^{-1}$ and $\mathcal{Q}_{\mathbf{Z}|\mathbf{X}}$ is the set of encoders $Q_{\mathbf{Z}|\mathbf{X}}$ such that $Q_{\mathbf{Z}|\mathbf{X}} P_{\mathbf{X}}$ is jointly stationary in $(\mathbf{X}, \mathbf{Z})$ and $Q_{\mathbf{Z}} \triangleq \int Q_{\mathbf{Z}|\mathbf{X}} dP_{\mathbf{X}}$ is equal to $P_{\mathbf{Z}}$.

Proof. Recall that process distributions $P_{\mathbf{X}}$ and $P_{\mathbf{Z}}$ are given. Let $Q_{\mathbf{Y}|\mathbf{Z}}$ be a given conditional distribution such that the joint distribution $Q_{\mathbf{Y}|\mathbf{Z}} P_{\mathbf{Z}}$ of the pair process $(\mathbf{Y}, \mathbf{Z})$ is stationary. Let $\mathcal{P}_{\mathbf{X},\mathbf{Y},\mathbf{Z}}$ be the set of all stationary distributions $\pi_{\mathbf{X},\mathbf{Y},\mathbf{Z}}$ having $P_{\mathbf{X}}$ and $Q_{\mathbf{Y}|\mathbf{Z}} P_{\mathbf{Z}}$ as marginals, satisfying that $\mathbf{X}$ and $\mathbf{Y}$ are conditionally independent given $\mathbf{Z}$. Also denote by $\mathcal{P}_{\mathbf{X},\mathbf{Y}}$ and $\mathcal{P}_{\mathbf{X},\mathbf{Z}}$ the sets of marginal distributions on $(\mathbf{X}, \mathbf{Y})$ and $(\mathbf{X}, \mathbf{Z})$ induced by $\mathcal{P}_{\mathbf{X},\mathbf{Y},\mathbf{Z}}$, respectively. Note that while $\mathcal{P}_s(P_{\mathbf{X}}, P_{\mathbf{Y}})$, $\mathcal{P}_{\mathbf{X},\mathbf{Y},\mathbf{Z}}$, and $\mathcal{P}_{\mathbf{X},\mathbf{Y}}$ depend on the choice of $Q_{\mathbf{Y}|\mathbf{Z}}$, but $\mathcal{P}_{\mathbf{X},\mathbf{Z}}$ does not. It follows that $\mathcal{P}_{\mathbf{X},\mathbf{Z}} = \mathcal{P}_s(P_{\mathbf{X}}, P_{\mathbf{Z}})$.

We then show that $\mathcal{P}_{\mathbf{X},\mathbf{Y}} = \mathcal{P}_s(P_{\mathbf{X}}, P_{\mathbf{Y}})$ if $Q_{\mathbf{Y}|\mathbf{Z}}(\cdot|\mathbf{z})$ is a Dirac measure on $\mathbf{g}(\mathbf{z})$ for all $\mathbf{z}$. That $\mathcal{P}_{\mathbf{X},\mathbf{Y}} \subset \mathcal{P}_s(P_{\mathbf{X}}, P_{\mathbf{Y}})$ is obvious. To show the opposite direction, suppose $\pi_{\mathbf{X},\mathbf{Y}} \in \mathcal{P}_s(P_{\mathbf{X}}, P_{\mathbf{Y}})$. Let $\pi_{\mathbf{X},\mathbf{Y},\mathbf{Z}}$ be any extension of $\pi_{\mathbf{X},\mathbf{Y}}$ such that its marginal on $(\mathbf{Y}, \mathbf{Z})$ is $Q_{\mathbf{Y}|\mathbf{Z}} P_{\mathbf{Z}}$. Such an extension is possible since if we let $\tilde{\mathbf{g}} : \mathbf{Z} \to (\mathbf{Z}, \mathbf{g}(\mathbf{Z}))$, then $Q_{\mathbf{Y}|\mathbf{Z}} P_{\mathbf{Z}} = P_{\mathbf{Z}}\tilde{\mathbf{g}}^{-1}$, and also $P_{\mathbf{Y}} = P_{\mathbf{Z}}\mathbf{g}^{-1}$. Then for any event $F \in \sigma(\mathbf{Y})$, we have $\pi_{\mathbf{Y}|\mathbf{X},\mathbf{Z}}(F) = \pi_{\mathbf{Y}|\mathbf{Z}}(F)$ because $\mathbf{Y} = \mathbf{g}(\mathbf{Z})$ a.s. This implies that $\mathbf{X}$ and $\mathbf{Y}$ are conditionally independent given $\mathbf{Z}$, thus $\pi_{\mathbf{X},\mathbf{Y},\mathbf{Z}} \in \mathcal{P}_{\mathbf{X},\mathbf{Y},\mathbf{Z}}$. It follows that $\pi_{\mathbf{X},\mathbf{Y}} \in \mathcal{P}_{\mathbf{X},\mathbf{Y}}$.

Finally,

$$
\begin{aligned}
\bar{\rho}(P_{\mathbf{X}}, P_{\mathbf{Y}}; \mathbf{g}, P_{\mathbf{Z}}) &= \inf_{\pi_{\mathbf{X},\mathbf{Y}} \in \mathcal{P}_s(P_{\mathbf{X}}, P_{\mathbf{Y}})} E_{\pi_{\mathbf{X},\mathbf{Y}}} d^p(X_0, Y_0) \\
&= \inf_{\pi_{\mathbf{X},\mathbf{Y}} \in \mathcal{P}_{\mathbf{X},\mathbf{Y}}} E_{\pi_{\mathbf{X},\mathbf{Y}}} d^p(X_0, Y_0) \\
&= \inf_{\pi_{\mathbf{X},\mathbf{Y},\mathbf{Z}} \in \mathcal{P}_{\mathbf{X},\mathbf{Y},\mathbf{Z}}} E_{\pi_{\mathbf{X},\mathbf{Y},\mathbf{Z}}} d^p(X_0, Y_0) \\
&= \inf_{\pi_{\mathbf{X},\mathbf{Y},\mathbf{Z}} \in \mathcal{P}_{\mathbf{X},\mathbf{Y},\mathbf{Z}}} E_{\pi_{\mathbf{Z}}} E_{\pi_{\mathbf{X},\mathbf{Y}|\mathbf{Z}}} d^p(X_0, g(\mathbf{Z})_0) \\
&= \inf_{\pi_{\mathbf{X},\mathbf{Y},\mathbf{Z}} \in \mathcal{P}_{\mathbf{X},\mathbf{Y},\mathbf{Z}}} E_{\pi_{\mathbf{Z}}} E_{\pi_{\mathbf{X}|\mathbf{Z}}} E_{\pi_{\mathbf{Y}|\mathbf{Z}}} d^p(X_0, g(\mathbf{Z})_0) \qquad (2.11) \\
&= \inf_{\pi_{\mathbf{X},\mathbf{Y},\mathbf{Z}} \in \mathcal{P}_{\mathbf{X},\mathbf{Y},\mathbf{Z}}} E_{\pi_{\mathbf{Z}}} E_{\pi_{\mathbf{X}|\mathbf{Z}}} d^p(X_0, g(\mathbf{Z})_0) \\
&= \inf_{\pi_{\mathbf{X},\mathbf{Z}} \in \mathcal{P}_{\mathbf{X},\mathbf{Z}}} E_{\pi_{\mathbf{X},\mathbf{Z}}} d^p(X_0, g(\mathbf{Z})_0) \\
&= \inf_{\pi_{\mathbf{X},\mathbf{Z}} \in \mathcal{P}_s(P_{\mathbf{X}}, P_{\mathbf{Z}})} E_{\pi_{\mathbf{X}}} E_{\pi_{\mathbf{Z}|\mathbf{X}}} d^p(X_0, g(\mathbf{Z})_0) \qquad (2.12) \\
&= \inf_{\substack{\pi_{\mathbf{Z}|\mathbf{X}} : \int \pi_{\mathbf{Z}|\mathbf{X}} dP_{\mathbf{X}} = P_{\mathbf{Z}}, \\ \pi_{\mathbf{Z}|\mathbf{X}} P_{\mathbf{X}} \in \mathcal{P}_s(P_{\mathbf{X}}, P_{\mathbf{Z}})}} E_{P_{\mathbf{X}}} E_{\pi_{\mathbf{Z}|\mathbf{X}}} d^p(X_0, g(\mathbf{Z})_0) \\
&= \inf_{\pi_{\mathbf{Z}|\mathbf{X}} \in \mathcal{Q}_{\mathbf{Z}|\mathbf{X}}} E_{P_{\mathbf{X}}} E_{\pi_{\mathbf{Z}|\mathbf{X}}} d^p(X_0, g(\mathbf{Z})_0),
\end{aligned}
$$

where equation (2.11) is due to the conditional independence. Equation (2.12) follows from the observed $\mathcal{P}_{\mathbf{X},\mathbf{Z}} = \mathcal{P}_s(P_{\mathbf{X}}, P_{\mathbf{Y}})$. □

Let $\mathcal{G}^{\infty}_{\text{det}}$ be the set of deterministic decoders $Q_{\mathbf{Y}|\mathbf{Z}}$ such that $Q_{\mathbf{Y}|\mathbf{Z}}(\cdot|\mathbf{z})$ is the Dirac measure on $\mathbf{g}(\mathbf{z})$ for all $\mathbf{z}$, i.e., $\mathbf{Y} = \mathbf{g}(\mathbf{Z})$ a.s. for a function $\mathbf{g} : \mathcal{Z}^{\infty} \to \mathcal{X}^{\infty}$ that maps a stationary sequence to a stationary sequence. By minimizing the reparametrization of Theorem 1 over $\mathcal{G}^{\infty}_{\text{det}}$, we obtain an OAE model. Similar to relaxation (2.7), we may solve an unconstrained problem,

$$
\inf_{Q_{\mathbf{Y}|\mathbf{Z}} \in \mathcal{G}^{\infty}_{\text{det}}} \inf_{Q_{\mathbf{Z}|\mathbf{X}} \in \mathcal{Q}_{\mathbf{Z}|\mathbf{X}}} \mathrm{E}_{P_{\mathbf{X}}} \mathrm{E}_{Q_{\mathbf{Z}|\mathbf{X}}} \mathrm{E}_{Q_{\mathbf{Y}|\mathbf{Z}}} d^p(X_0, Y_0) + \lambda \mathcal{D}_{\mathbf{Z}}(P_{\mathbf{Z}}, Q_{\mathbf{Z}}). \qquad (2.13)
$$

The additional constraint of $Q_{\mathbf{Z}|\mathbf{X}}P_{\mathbf{X}}$ being stationary can be satisfied by restricting $Q_{\mathbf{Z}|\mathbf{X}}$ to be stationary (the latter implies the former).

Despite the apparent similarity to WAE (2.7), problem (2.13) has two practical issues. First, the decoder $\mathbf{g}$ for OAE needs to map an *infinite* sequence to another infinite sequence. Learning such a map with infinite memory may face computational challenges. Second, since $P_{\mathbf{Z}}$ and $Q_{\mathbf{Z}}$ are both *process* distributions, computing the divergence $\mathcal{D}_{\mathbf{Z}}$ may also be difficult.

2.4.2 OAE for exchangeable data

If we can assume that the process $\mathbf{X}$ is exchangeable, we can find a tractable upper bound of $\bar{\rho}(\mathbf{X}, \mathbf{Y}; \mathbf{Z})$ by bringing the nested structure of the sequence $\mathbf{X}$ down to the latent sequence $\mathbf{Z}$. Recall that a version of De Finetti's theorem (Olshen, 1974) ensures the existence of a real-valued random variable conditioned on which the coordinates of $\mathbf{X}$ are i.i.d. when the sequence $\mathbf{X}$ is exchangeable.

Theorem 2. *Assume process distributions $P_{\mathbf{X}}$ on $\mathcal{X}^\infty$ and $P_{\mathbf{Z}}$ on $\mathcal{Z}^\infty$ are both exchangeable. Also, assume there exists a distribution P_B on another complete, separable metric space $\mathcal{B}$ (e.g., $\mathcal{B} = \mathbb{R}$) such that its joint distributions $P_{\mathbf{X},B}$ and $P_{\mathbf{Z},B}$ satisfy $P_{\mathbf{X_{1:n}},B} = \big[\prod_{j=1}^n P_{X_0|B}\big]P_B$ and $P_{\mathbf{Z_{1:n}},B} = \big[\prod_{j=1}^n P_{Z_0|B}\big]P_B$ for any n, respectively. Then, for any $Q_{\mathbf{Y}|\mathbf{Z}} \in \mathcal{G}^\infty_{det}$,*

$$\bar{\rho}(P_{\mathbf{X}}, P_{\mathbf{Y}}; \mathbf{g}, P_{\mathbf{Z}}) \leq \inf_{Q_{\mathbf{Z}|\mathbf{X},B} \in \mathcal{Q}} \mathrm{E}_{P_{\mathbf{X},B}} \mathrm{E}_{Q_{\mathbf{Z}|\mathbf{X},B}} \mathrm{E}_{Q_{\mathbf{Y}|\mathbf{Z}}} d^p(X_0, Y_0),$$

where $P_{\mathbf{Y}} = \int_{\mathcal{Z}^\infty} Q_{\mathbf{Y}|\mathbf{Z}} dP_{\mathbf{Z}} = P_{\mathbf{Z}}\mathbf{g}^{-1}$. Here, $\mathcal{Q}$ is the set of all conditional distributions $Q_{\mathbf{Z}|\mathbf{X},B}$ such that the joint distribution $Q_{\mathbf{Z}|\mathbf{X},B}P_{\mathbf{X},B}$ of $(\mathbf{X}, \mathbf{Z}, B)$ has the marginals $\big[\prod_{j=1}^n Q_{Z_0|X_0,B}\big]P_{\mathbf{X}_{1:n},B}$ and $P_{\mathbf{Z_{1:n}}}$ on $(\mathbf{X}_{1:n}, \mathbf{Z}_{1:n}, B)$ and $\mathbf{Z}_{1:n}$ for any n, respectively.

Proof. We first need to check if $\mathcal{Q}$ is not empty. Consider a distribution $Q_{Z_0,X_0|B}$

having marginals $P_{Z_0|B}$ and $P_{X_0|B}$. A trivial example of such a conditional distribution is $P_{Z_0|B}P_{X_0|B}$. Then, we can find a conditional distribution $Q_{Z_0|X_0,B}$ satisfying that $Q_{Z_0|X_0,B}P_{X_0|B} = Q_{Z_0,X_0|B}$, so that $\int_{\mathcal{X}} Q_{Z_0|X_0,B} dP_{X_0|B} = P_{Z_0|B}$. With this distribution $Q_{Z_0|X_0,B}$, we can construct a conditional distribution $Q_{\mathbf{Z}|\mathbf{X},B}$ such that the joint distribution $Q_{\mathbf{Z}|\mathbf{X},B}P_{\mathbf{X},B}$ of $(\mathbf{X}, \mathbf{Z}, B)$ has a marginal $\big[\prod_{j=1}^n Q_{Z_0|X_0,B}\big] P_{\mathbf{X}_{1:n},B}$ on $(\mathbf{X}_{1:n}, \mathbf{Z}_{1:n}, B)$ for any n. Then, its marginal on $\mathbf{Z}_{1:n}$ is

$$
\begin{aligned}
&\int_{\mathcal{X}^n \times \mathcal{B}} \big[\prod_{j=1}^n Q_{Z_0|X_0,B}\big] dP_{\mathbf{X}_{1:n},B} \\
&= \int_{\mathcal{X}^n \times \mathcal{B}} \big[\prod_{j=1}^n Q_{Z_0|X_0,B}\big] d\big[P_B \prod_{j=1}^n P_{X_0|B}\big] \\
&= \int_{\mathcal{B}} \prod_{j=1}^n \big[\int_{\mathcal{X}} Q_{Z_0|X_0,B} dP_{X_0|B}\big] dP_B \\
&= \int_{\mathcal{B}} \big[\prod_{j=1}^n P_{Z_0|B}\big] dP_B \\
&= P_{\mathbf{Z}_{1:n}},
\end{aligned}
$$

for any n. From the Kolmogorov extension theorem, it follows that $Q_{\mathbf{Z}|\mathbf{X},B} \in \mathcal{Q}$. Thus $\mathcal{Q}$ is not empty.

Recall that $\mathcal{P}_s(P_{\mathbf{X}}, P_{\mathbf{Z}})$ is the set of all jointly stationary distributions of $(\mathbf{X}, \mathbf{Z}) \in \mathcal{X}^\infty \times \mathcal{Z}^\infty$ having $P_{\mathbf{X}}$ and $P_{\mathbf{Z}}$ as marginals. Let $\mathcal{P}_{\mathbf{X},\mathbf{Z},B}$ be the set of all joint distributions $Q_{\mathbf{Z}|\mathbf{X},B}P_{\mathbf{X},B}$ of $(\mathbf{X}, \mathbf{Z}, B) \in \mathcal{X}^\infty \times \mathcal{Z}^\infty \times \mathcal{B}$ for any $Q_{\mathbf{Z}|\mathbf{X},B} \in \mathcal{Q}$, and $\mathcal{P}_{\mathbf{X},\mathbf{Z}}$ be the set of marginal distributions on $(\mathbf{X}, \mathbf{Z})$ induced by $\mathcal{P}_{\mathbf{X},\mathbf{Z},B}$. For any $\pi_{\mathbf{X},\mathbf{Z}} \in \mathcal{P}_{\mathbf{X},\mathbf{Z}}$, there exists $\pi_{\mathbf{X},\mathbf{Z},B} \in P_{\mathbf{X},\mathbf{Z},B}$ such that its marginal is $\pi_{\mathbf{X},\mathbf{Z}}$. Since any joint distribution in $\mathcal{P}_{\mathbf{X},\mathbf{Z},B}$ has marginals $P_{\mathbf{X}}$ and $P_{\mathbf{Z}}$, it follows that $\pi_{\mathbf{X},\mathbf{Z}}$ has marginals $P_{\mathbf{X}}$ and $P_{\mathbf{Z}}$. Also, $\pi_{\mathbf{X},\mathbf{Z},B}$ has a marginal $\big[\prod_{j=1}^n Q_{Z_0|X_0,B}\big] P_{\mathbf{X}_{1:n},B} = \big[\prod_{j=1}^n P_{X_0|B} Q_{Z_0|X_0,B}\big] P_B$ on $\mathcal{X}^n \times \mathcal{Z}^n \times \mathcal{B}$ for some

$Q_{Z_0|X_0,B}$ and for any n. This means that $\pi_{\mathbf{X},\mathbf{Z}}$ is jointly exchangeable i.e., $\{(X_j, Y_j)\}_{j=-\infty}^{j=\infty}$ is exchangeable, thus stationary. Then, $\pi_{\mathbf{X},\mathbf{Z}} \in \mathcal{P}_s(\mathbf{X}, \mathbf{Z})$, so that $\mathcal{P}_s(\mathbf{X}, \mathbf{Z}) \supset \mathcal{P}_{\mathbf{X},\mathbf{Z}}$. Now starting from (2.12) in the proof of Theorem 1, we can conclude that,

$$\begin{aligned}
\bar{\rho}(P_{\mathbf{X}}, P_{\mathbf{Y}}; \mathbf{g}, P_{\mathbf{Z}}) &= \inf_{\pi_{\mathbf{X},\mathbf{Z}} \in \mathcal{P}_s(P_{\mathbf{X}}, P_{\mathbf{Z}})} E_{\pi_{\mathbf{X}}} E_{\pi_{\mathbf{Z}|\mathbf{X}}} d^p(X_0, \mathbf{g}(\mathbf{Z})_0) \\
&= \inf_{\pi_{\mathbf{X},\mathbf{Z}} \in \mathcal{P}_s(P_{\mathbf{X}}, P_{\mathbf{Z}})} E_{\pi_{\mathbf{X},\mathbf{Z}}} d^p(X_0, \mathbf{g}(\mathbf{Z})_0) \\
&\leq \inf_{\pi_{\mathbf{X},\mathbf{Z}} \in \mathcal{P}_{\mathbf{X},\mathbf{Z}}} E_{\pi_{\mathbf{X},\mathbf{Z}}} d^p(X_0, \mathbf{g}(\mathbf{Z})_0) \\
&= \inf_{\pi_{\mathbf{X},\mathbf{Z},B} \in \mathcal{P}_{\mathbf{X},\mathbf{Z},B}} E_{\pi_{\mathbf{X},\mathbf{Z},B}} d^p(X_0, \mathbf{g}(\mathbf{Z})_0) \\
&= \inf_{\pi_{\mathbf{Z}|\mathbf{X},B} \in \mathcal{Q}} E_{P_{\mathbf{X},B}} E_{\pi_{\mathbf{Z}|\mathbf{X},B}} d^p(X_0, \mathbf{g}(\mathbf{Z})_0).
\end{aligned}$$

□

Now, let us define,

$$\mathcal{D}_{\mathrm{OAE}}(P_{\mathbf{X}}, P_{\mathbf{Y}}; P_{\mathbf{Z}}) = \inf_{Q_{\mathbf{Z}|\mathbf{X},B} \in \mathcal{Q}} \mathrm{E}_{P_{\mathbf{X},B}} \mathrm{E}_{Q_{\mathbf{Z}|\mathbf{X},B}} \mathrm{E}_{Q_{\mathbf{Y}|\mathbf{Z}}} d^p(X_0, Y_0). \tag{2.14}$$

Then, minimizing (2.14) corresponds to minimizing an upper bound of $\bar{\rho}(P_{\mathbf{X}}, P_{\mathbf{Y}}; P_{\mathbf{Z}})$.

To handle the complexity of decoder $\mathbf{g}$, which is a map from an infinite sequence to another infinite sequence, we may restrict the domain of the optimization problem. Let $\mathcal{G}^*_{\mathrm{det}}$ be the set of deterministic decoders $Q_{\mathbf{Y}|\mathbf{Z}}$ such that $Q_{\mathbf{Y}|\mathbf{Z}}(\cdot|\mathbf{z})$ is the Dirac measure on $\mathbf{g}(\mathbf{z})$ for all $\mathbf{z}$, i.e. $\mathbf{Y} = \mathbf{g}(\mathbf{Z})$ a.s., where the

ith element of $\mathbf{g}(\mathbf{Z})$ is $Y_i = g(Z_i)$, for $g \in \mathcal{NN}$. For such $Q_{\mathbf{Y}|\mathbf{Z}} \in \mathcal{G}^*_{\text{det}}$, we have

$$\begin{aligned}\mathcal{D}_{\text{OAE}}(P_{\mathbf{X}}, P_{\mathbf{Y}}; g, P_{\mathbf{Z}}) &= \inf_{Q_{\mathbf{Z}|\mathbf{X},B} \in \mathcal{Q}} \mathrm{E}_{P_{\mathbf{X},B}} \mathrm{E}_{Q_{\mathbf{Z}|\mathbf{X},B}} \mathrm{E}_{Q_{\mathbf{Y}|\mathbf{Z}}} d^p(X_0, Y_0) \\ &= \inf_{Q_{\mathbf{Z}|\mathbf{X},B} \in \mathcal{Q}} \mathrm{E}_{P_{\mathbf{X},B}} \mathrm{E}_{Q_{\mathbf{Z}|\mathbf{X},B}} d^p(X_0, g(Z_0))\end{aligned}$$

If we minimize $\mathcal{D}_{\text{OAE}}(P_{\mathbf{X}}, P_{\mathbf{Y}}; g, P_{\mathbf{Z}})$ over $\mathcal{NN}$:

$$\inf_{g \in \mathcal{NN}} \inf_{Q_{\mathbf{Z}|\mathbf{X},B} \in \mathcal{Q}} \mathrm{E}_{P_{\mathbf{X}}} \mathrm{E}_{Q_{\mathbf{Z}|\mathbf{X},B}} d^p(X_0, g(Z_0)), \tag{2.15}$$

we obtain a reduced OAE model. Since $\mathcal{Q}$ is parametrized by $Q_{Z_0|X_0,B}$, similar to relaxation (2.7), we may solve an unconstrained problem,

$$\inf_{g \in \mathcal{NN}} \inf_{Q_{\mathbf{Z}|\mathbf{X},B} \in \mathcal{Q}} \mathrm{E}_{P_{\mathbf{X}}} \mathrm{E}_{Q_{\mathbf{Z}|\mathbf{X},B}} d^p(X_0, g(Z_0)) + \lambda \mathcal{D}_{Z_0}(P_{Z_0}, \int_{\mathcal{X} \times \mathcal{B}} Q_{Z_0|X_0,B} dP_{X_0,B}). \tag{2.16}$$

A notably unique feature of the OAE is that it can generate samples from a given observational unit *whether or not the unit is present in the training dataset*. Variations of a new given unit can be generated as well as known units, and even a new unit can be generated from the latent space. This has not been possible with other existing methods — notably, conditional LVMs.

Chapter 3

Random-Intercept Ornstein Auto-Encoders

3.1 Random-intercept OAE

A way of enabling the unique feature of the OAE discussed in the last paragraph of the previous chapter is to specify $P_{\mathbf{Z}}$ through the random intercept model (1.1), where the latter completely specifies an exchangeable distribution of the random process $\mathbf{Z}$, and a realized random intercept corresponds to an observational unit. For distributions P_B and P_{E_0} on $\mathcal{Z}$, let

$$Z_j^i = B^i + E_j^i, \quad B^i \overset{\text{i.i.d.}}{\sim} P_B, \quad E_j^i \overset{\text{i.i.d.}}{\sim} P_{E_0}, \quad B^i \perp\!\!\!\perp E_j^i.$$

Obviously, $\mathbf{Z}_{1:n}^i$ has distribution $\int_{\mathcal{Z}} \left[\prod_{j=1}^n P_{Z_0|B}\right] dP_B$, where $Z_0 \overset{d}{=} Z_j^i$. Since each unit is represented by the random intercept B^i, B^i should encode the "identity" of the observational unit shared among the coordinates of sequence $\mathbf{Z}^i$ in unit i. For given B^i, then, Z_j^i encodes "within-unit variation" of unit i.

This random intercept model can explicitly model exchangeability in the latent space. Hereafter, we call this approach to OAE the *random-intercept OAE*. Note that B should satisfy $P_{\mathbf{X_{1:n}},B} = \big[\prod_{j=1}^{n} P_{X_0|B}\big] P_B$ for all n by the assumption. The latter condition well-defines the conditional distributions $P_{B|\mathbf{X}}$ such that $P_{B|\mathbf{X}_{1:n}} P_{\mathbf{X}_{1:n}} = P_{\mathbf{X}_{1:n},B}$ for all n. In order to satisfy the assumption on B, we need to introduce an encoder $Q_{B|\mathbf{X}}$ that mimics $P_{B|\mathbf{X}}$. Now, the probability encoder is a pair $(Q_{Z_0|X_0,B}, Q_{B|\mathbf{X}})$. Constraining $\int_{\mathcal{X}} Q_{Z_0|B,X_0} dP_{X_0|B} = P_{Z_0|B}$ and $\int_{\mathcal{X}^\infty} Q_{B|\mathbf{X}} dP_{\mathbf{X}} = P_B$ ensures $P_{Z_0} = \int_{\mathcal{X}} Q_{Z_0|X_0,B} dP_{X_0,B}$. By minimizing the objective of (2.15) under these new constraints, the random-intercept OAE solves the following relaxation:

$$\begin{aligned}\inf_{g\in\mathcal{NN}} \inf_{Q_{B|\mathbf{X}}} \inf_{Q_{Z_0|X_0,B}} \big[& \mathrm{E}_{P_{\mathbf{X}}} \mathrm{E}_{Q_{B|\mathbf{X}}} \mathrm{E}_{Q_{Z_0|B,X_0}} d^p(X_0, g(Z_0)) \\ & + \lambda_1 \mathrm{E}_{P_B} \mathcal{D}_1(P_{Z_0|B}, \int_{\mathcal{X}} Q_{Z_0|B,X_0} dP_{X_0}) + \lambda_2 \mathcal{D}_2(P_B, \int_{\mathcal{X}^\infty} Q_{B|\mathbf{X}} dP_{\mathbf{X}}) \big].\end{aligned} \tag{3.1}$$

The penalties relax the constraints,

$$P_{Z_0|B} = \int_{\mathcal{X}} Q_{Z_0|B,X_0} dP_{X_0} =: Q_{Z_0|B}, \quad P_B = \int_{\mathcal{X}^\infty} Q_{B|\mathbf{X}} dP_{\mathbf{X}} =: Q_B.$$

For the divergence measures $\mathcal{D}_1$ and $\mathcal{D}_2$, that utilized by GAN (see Bousquet et al., 2017)

$$\mathcal{D}_{\mathrm{GAN}}(P_{Z_0|B}, Q_{Z_0|B}) = \sup_{f\in\mathcal{F}_{NN}} \mathrm{E}_{P_{Z_0|B}} \log f(Z_0) + \mathrm{E}_{Q_{Z_0|B}} \log(1 - f(Z_0)),$$

where $\mathcal{F}_{NN}$ is the set of the "discriminator" functions $f : \mathcal{X} \to (0, 1)$ that can be parametrized by a neural network, and the maximum mean discrepancy (MMD,

Gretton et al., 2012)

$$\mathcal{D}_{\mathrm{MMD},\kappa}(P_B, Q_B) = \|\mathrm{E}_{P_B}\kappa(\cdot, B) - \mathrm{E}_{Q_B}\kappa(\cdot, B)\|_{\mathcal{H}}$$

for a positive definite kernel $\kappa : \mathcal{Z} \times \mathcal{Z} \to \mathbb{R}$ that induces a reproducing kernel Hilbert space $\mathcal{H}$ equipped with the norm $\|\cdot\|_{\mathcal{H}}$, are promoted.

In practice, using sample estimates of the terms in (3.1) is convenient to devise an implementable algorithm. The most difficult part is to sample from the encoder $Q_{B|\mathbf{X}}$, which requires infinite-length data. We take a mean-field-like approach. Instead of $Q_{B|\mathbf{X}}$, we employ $Q_{B|X_0}$, an encoder that takes only a single coordinate of data as input. For each identity i with m_i repeated measurements, we sample $\hat{b}^i_j \sim Q_{B|X_0}(\cdot|x^i_j)$ for each $j = 1, \ldots, m_i$. We then aggregate $\hat{b}^i_j$'s to obtain $\hat{b}^i = \frac{1}{m_i}\sum_{j=1}^{m_i} \hat{b}^i_j$ to approximate a sample from $Q_{B|\mathbf{X}}$. Sampling from $Q_{Z_0|X_0,B}$ is relatively simple. We sample $\hat{z}^i_j$ independently from $Q_{Z_0|X_0,B}(\cdot|x^i_j, \hat{b}^i)$ given this $\hat{b}^i$ and the observations x^i_j of unit i, for $j = 1, \ldots, m_i$. We may refer to $Q_{B|X_0}$ and $Q_{Z_0|X_0,B}$ as "identity encoder" and "within-unit variation encoder", respectively.

The resulting algorithm is summarized in Algorithm 1.

Algorithm 1 Random-intercept OAE training

Input: Exchangeable sequences $(x_1^i, ..., x_{n_i}^i)$ for $i = 1, ..., L$

Output: Encoder pair $(Q_{B|X_0}, Q_{Z_0|X_0,B})$ and decoder g

Require: $P_B, P_{Z_0|B}$, regularization coefficients λ_1, λ_2, and positive definite kernel κ

1: **Initialize**: parameters of $(Q_{Z_0|X_0,B}, Q_{B|X_0})$, g, and discriminator f

2: **while** $Q_{B|X_0}$, $Q_{Z_0|X_0,B}$, f, g not converged **do**

3: Sample observational unit $i = 1, \ldots, n$ and sequence $(x_1^i, \ldots, x_{m_i}^i)$ for each unit i from the training set

4: Sample b^i from P_B for $i = 1, \ldots, n$

5: Sample $(z_1^i, \ldots, z_{m_i}^i)$ from $P_{Z_0|B}$ given b^i for $i = 1, \ldots, n$

6: Sample $\hat{b}_j^i \sim Q_{B|X_0}(\cdot|x_j^i)$ for each $j = 1, \ldots, m_i$ and aggregate $\hat{b}^i = \frac{1}{m_i} \sum_{j=1}^{m_i} \hat{b}_j^i$ for $i = 1, \ldots, n$.

7: Sample $(\hat{z}_1^i, \cdots, \hat{z}_{m_i}^i)$ from $Q_{Z_0|X_0,B}$ given $\hat{b}^i$ and $(x_1^i, \cdots, x_{m_i}^i)$ for $i = 1, ..., n$.

8: Update $Q_{Z_0|X_0,B}$, $Q_{B|X_0}$, and g by descending:

$$\frac{1}{n} \sum_{i=1}^{n} \frac{1}{m_i} \sum_{j=1}^{m_i} d^p(x_j^i, g(\hat{z}_j^i)) - \frac{\lambda_1}{n} \sum_{i=1}^{n} \frac{1}{m_i} \sum_{j=1}^{m_i} \log f(\hat{z}_j^i)$$

$$+ \frac{\lambda_2}{n(n-1)} \Big(\sum_{i \neq l} \kappa(b^i, b^l) + \sum_{i \neq l} \kappa(\hat{b}^i, \hat{b}^l) \Big) - \frac{2\lambda_2}{n^2} \sum_{i,l} \kappa(b^i, \hat{b}^l)$$

9: Update f by ascending: $\sum_{i=1}^{n} \sum_{j=1}^{m_i} \log f(z_j^i) + \log(1 - f(\hat{z}_j^i))$

10: **end while**

3.2 Empirical results

The performance of the random-intercept OAE with the proposed training method is assessed for the identity-preserving generation tasks discussed in Sect 1.1. **Exemplar generation**: Given a few observations $(x_j^i)_{j=1}^m$ from a new unit i, sample the "identity variables" $(\hat{b}_j^i)_{j=1}^m$ from $Q_{B|X_0}(\cdot|x_j^i)$ to take an average $\hat{b}^i = \frac{1}{m}\sum_{j=1}^m \hat{b}_j^i$. Draw a "within-unit variation" z_j^{new} from $P_{Z_0|B}(\cdot|\hat{b}^i)$. An exemplar is the reconstruction $g(z_j^{\text{new}})$. If $m = 1$, this task is called one-shot exemplar generation. **Style transfer**: From observations $(x_l^k)_{l=1}^L$ of another unit $k \neq i$, sample $\hat{b}^k$ as in exemplar generation. Draw "within-unit variation" $\hat{z}_l^k$ from $Q_{Z_0|B,X_0}(\cdot|\hat{b}^k, x_l^k)$. Then sequence $(g(\hat{z}_l^k))_{l=1}^L$ transfers the style of unit k to i. If $m = 1$, this task is called one-shot style transfer.

3.2.1 Implementation

In all the experiments in the following, it was assumed that $\mathcal{X}$ and $\mathcal{Z}$ are Euclidean spaces with dimensions $d_{\mathcal{X}}$ and $d_{\mathcal{Z}}$, respectively; accompanied Euclidean metric $d(x_0, x_0') = \|x_0 - x_0'\|_2$ on $\mathcal{X}$ and $p = 2$ were used. The prior distribution $P_{\mathbf{Z}}$ of the latent variable $\mathbf{Z}$ was set as a random intercept model:

$$Z_j^i|\{B^i = b^i\} \overset{\text{i.i.d.}}{\sim} \mathcal{N}(b^i, \mathbf{I}_{d_{\mathcal{Z}}}), \quad B^i \overset{\text{i.i.d.}}{\sim} \mathcal{N}(0, 100\mathbf{I}_{d_{\mathcal{Z}}}). \tag{3.2}$$

The encoder pair $(Q_{B|X_0}, Q_{Z|X_0,B})$ was designed to be another random intercept model:

$$\begin{aligned} Z_j^i|\{B^i = b^i, X_j^i = x_j^i\} &\overset{\text{i.i.d.}}{\sim} \mathcal{N}(\mu(x_j^i) + b^i, \sigma^2(x_j^i)\mathbf{I}_{d_{\mathcal{Z}}}) \\ B^i|\{X_j^i = x_j^i\} &\overset{\text{i.i.d.}}{\sim} \mathcal{N}(\nu(x_j^i), \tau^2\mathbf{I}_{d_{\mathcal{Z}}}), \end{aligned} \tag{3.3}$$

where the mean functions $\mu : \mathcal{X} \to \mathcal{Z}$, $\nu : \mathcal{X} \to \mathcal{Z}$, and the variance function $\sigma^2 : \mathcal{X} \to \mathbb{R}_{++}$ were parameterized by deep neural networks. The hyperparameter τ was kept small. Although Gaussian encoders are suboptimal to our optimization problem (3.1) due to the restricted search space, Rubenstein et al. (2018b) has shown empirically that such a restriction produces better outcomes when the appropriate number of dimensions for the latent space is not known. The decoder g was also parameterized by deep neural networks.

In the case where all the observational units are present in the training data, the quality of samples from the random-intercept OAE was compared with CAAE, by interpreting each unit as a class. For CAAE, the conditional Gaussian latent variables are set:

$$Z_j^i | \{C^i = k\} \overset{\text{i.i.d.}}{\sim} \mathcal{N}(0, \mathbf{I}_{d_{\mathcal{Z}}}),$$

for $k = 1, \ldots, K$ where C is a given class label of X_0 and K is the number of subjects. For each class $k = 1, \ldots, K$, the encoder $Q_{Z_0|X_0,C}$ of CAAE were designed to be a Gaussian encoder:

$$Z_j^i | \{X_0 = x_j^i, C = k\} \overset{\text{i.i.d.}}{\sim} \mathcal{N}\big(\mu_k(x^i, c^i), \sigma_k^2(x^i, c^i)\mathbf{I}_{d_{\mathcal{Z}}}\big),$$

where $\mu_k : \mathcal{X} \times \{1, \ldots, K\} \to \mathcal{Z}$, $\sigma_k^2 : \mathcal{X} \times \{1, \ldots, K\} \to \mathbb{R}_{++}$ are parameterized by a deep neural network. The decoder $g : \mathcal{Z} \times \{1, \ldots, K\} \to \mathcal{X}$ was also parameterized by a deep neural network.

The Adam (Kingma and Ba, 2014) optimizer $\beta_1 = 0.5$ for updating the first moment estimate and $\beta_2 = 0.999$ for updating the second moment estimate was used for optimization. When generating new variations of a given subject from the test dataset, one image per subject was used. For all convolutional layers, the batch normalization (Ioffe and Szegedy, 2015), padding, and truncated normal

initialization were used. Further implementation details are given in Section 3.3.

3.2.2 A toy model

To see if random-intercept OAE can learn a known low dimensional distribution embeded in a higher dimension, training samples $Z_j^i = B^i + E_j^i$ were generated from the two-dimensional latent space for $i = 1, 2, ..., 100$, $j = 1, 2, ..., 5000$ with

$$E_j^i \sim \mathcal{N}\big(\left[\begin{smallmatrix} 0 \\ 0 \end{smallmatrix}\right], \left[\begin{smallmatrix} 0.009 & 0 \\ 0 & 0.007 \end{smallmatrix}\right]\big), B^i \sim \mathcal{N}\big(\left[\begin{smallmatrix} 0.2 \\ -0.4 \end{smallmatrix}\right], \left[\begin{smallmatrix} 1.018 & 0.12 \\ 0.12 & 0.745 \end{smallmatrix}\right]\big),$$

and embedded into four-dimensional Euclidean space by $X_j^i = AZ_j^i$ with

$$A = \begin{bmatrix} 0.027 & 0.171 & 0.084 & 0.290 \\ 0.252 & 0.388 & 0.248 & 0.371 \end{bmatrix}^T.$$

For learning the representation, the two-dimensional latent variable was misspecified for $i = 1, 2, ..., n$, $j = 1, 2, ..., n_i$ as

$$Z_j^i \sim B^i + E_j^i, \quad E_j^i \sim \mathcal{N}\big(\mathbf{0}, 0.01\mathbf{I}\big), \quad B^i \sim \mathcal{N}\big(\mathbf{0}, \mathbf{I}\big).$$

Then, the random-intercept OAE model with a simple architecture (4.8k parameters) was trained. The linear decoder was used to restrict the generated sample distribution to be normal. The model was trained for 50 epochs with mini-batch size 3000, $\lambda_1 = 10$, $\lambda_2 = 10$ and the learning rates of 0.01 for the encoder-decoder and 0.005 for the discriminator. After training, samples were generated through the decoder with $n = 100$ and $n_i = 500$, and measured the error between the samples and the true moments. The root-mean squared error (RMSE) of the mean $\mathrm{E}_{B^i}[\mathrm{E}[Y_j^i|B^i]]$ was 0.0233, and the RMSE of the covariance

matrix $\mathrm{E}_{B^i}[\mathrm{cov}[Y_j^i|B^i]]$ was 0.0001. This result shows that the random-intercept OAE works well on this toy but informative model.

3.2.3 VGGFace2 dataset

Recall that in the VGGFace2 dataset the portraits of each individual are highly correlated and exchangeable. It is also highly imbalanced, with number of portraits per person varying from 30 to 843. The goal of this experiment is to examine the capability of random-intercept OAE in exemplar generation of both known and unknown units in the presence of many units (classes) and data imbalance. As emphasized in the previous chapter, generating images from an unknown person is impossible with existing (conditional) LVMs, e.g., CAAE. For known subjects, the sample quality of OAE was compared with that of CAAE. For an unknown subject, CAAE cannot generate samples, and the quality of the generated samples was compared with WAE, which ignores the unit information.

Algorithm parameters. The latent space dimension was chosen as $d_{\mathcal{Z}} = 128$. The encoder-decoder architecture had 13.6M parameters and the discriminator had 12.8M parameters. $\lambda_1 = 10$, $\lambda_2 = 10$ were set for OAE, and $\lambda = 10$ was set for WAE and CAAE. All models were trained for 100 epochs with mini-batches of size 200. A constant learning rate was 0.0005 for the encoder and decoder, and 0.001 for the discriminator.

Training. For pre-processing, the faces were cropped and rescaled to a common size of 64 by 64. Then, a training set of 146,519 images was constructed from 500 randomly chosen subjects. Since the number of units far exceeded the mini-batch size and the dataset is highly imbalanced, importance sampling was

	Known units (Testset 1)		Unknown units (Testset 2)	
	FID	Sharpness	FID	Sharpness
Random-intercept OAE	151.994	1×10^{-4}	156.935	1×10^{-4}
CAAE	152.077	1×10^{-4}	-	-
WAE	-	-	163.612	1×10^{-4}
Testset	-	4×10^{-3}	-	3×10^{-3}

Table 3.1: VGGFace2 performance measures.

used to limit both the number of subjects and the maximum number of variations per mini-batch in early training epochs. For data augmentation, we either added white Gaussian noise to or vertically flipped randomly chosen images in a mini-batch.

Evaluation measures. The quality of generated samples was quantified by the sharpness using the Laplace filter (Rubenstein et al., 2018b), and the Frechet inception distance (FID) between the generated images distribution and the original image distributions (Heusel et al., 2017). Both are commonly used in the LVM literature. For FID, we picked 100 images from the generated samples and the test dataset.

One-shot exemplar generation from known units of VGGFace2. A test dataset (Testset 1) with 11,250 images of 49 people was constructed from the training dataset. From the random-intercept OAE and CAAE, 100 new variations for each units were generated. For the random-intercept OAE, only one image of each person was provided for generating new variations of that given unit. For CAAE, the label of each person was provided for both the encoder and decoder. The results, seen in Table 3.1, suggest that the random-intercept OAE could generate quality variations for known units better than CAAE.

One-shot exemplar generation from unknown subjects. Another test dataset (Testset 2), with 11,250 images of 49 people, was constructed from randomly chosen 500 units not used for training. From the random-intercept OAE, 100 new variations for each unit were generated. From WAE, in which unit information cannot be used, 4,900 images were generated. Table 3.1 shows that the random-intercept OAE can generate new variations for given but unknown units with sample quality comparable to WAE, which can only generate units but are not able to create their systemic variations. Figure 3.1 presents some one-shot exemplar generations of unknown subjects. Each row corresponds to a single person (unit). Column 1 shows the randomly chosen images of a unit, column 2 shows images decoded the identity variable, i.e., the encoded random intercepts, and each column from column 3 onwards represents a new variation applied on each estimated identity. Note that each row of Figure 3.1 should look like the same person.

Style transfer. Figure 3.2 demonstrates style transfer results when both target and base persons are chosen from an unknown person. The first row shows the observations of a given base person. Second row shows the reconstructions of the given observation, and the last row provides the style-transferred images. Note that the images of the last row of Figure 3.2 should look like the target person while the poses copy the corresponding image above in the first row. Figure 3.2 demonstrate that the random-intercept OAE can generate style-transferred images in high quality between two unknown persons. This generalizability is unique to OAE, and suggests that OAE can be a useful data augmentation tool for many applications such as face recognition in the presence of high data imbalance.

Unit-level disentanglement in representation from VGGFace2. Another benefit of our random process modeling is that units can be well-separated in the representation space. Figure 3.3 shows t-SNE maps (Maaten and Hinton, 2008) of the latent space representation of randomly selected 225 images of 10 people, known and unknown. For random-intercept OAE, t-SNE maps of the encoded images of known units (panel (a)) and unknown units (panel (b)) are shown in the Figure 3.3. Panel (c) shows t-SNE map of the encoded images from WAE. Each color represents a single person. For known units, clustering by unit is clear. Unknown units are also separated well, judged by visual inspection and by the ratio of within-group sum of squares (SSW) to between-group sum of squares (SSB); SSW/SSB is less than 1 for both cases. By design, WAE is incapable of separating units at all.

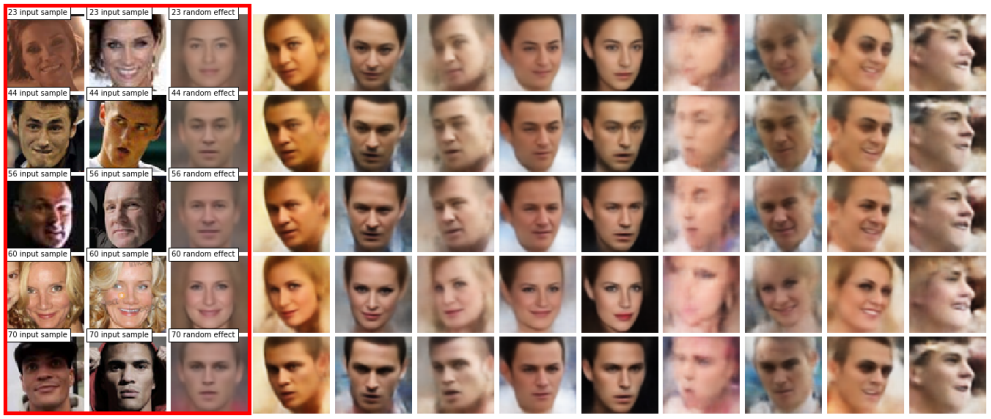


Figure 3.1: One-shot exemplar generation from unknown units of VGGFace2.

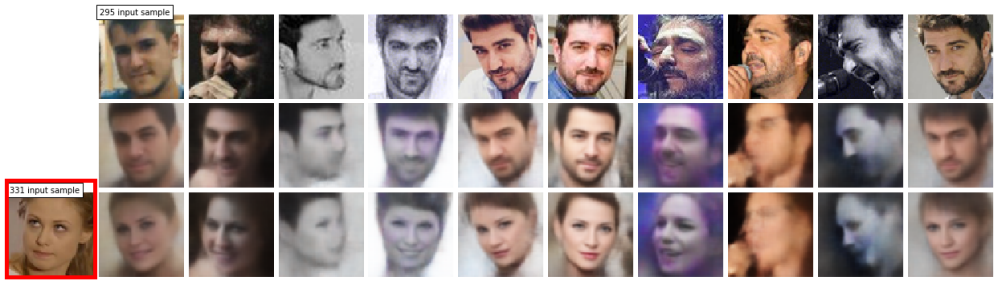
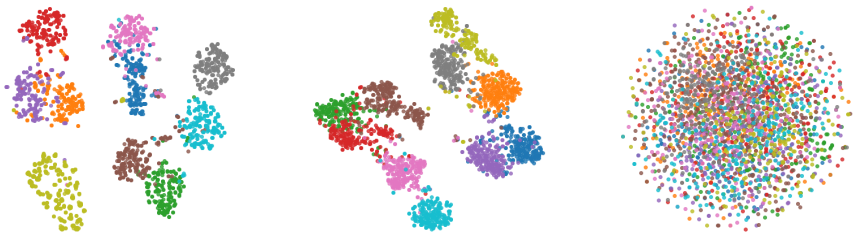


Figure 3.2: One-shot style transfer results for VGGFace2.



(a) OAE (known)
SSW/SSB = 0.362

(b) OAE (unknown)
SSW/SSB = 0.551

(c) WAE
SSW/SSB = 40.234

Figure 3.3: t-SNE map of the encoded images from VGGFace2.

3.2.4 MNIST dataset

The goal of this experiment is to examine the performance of random-intercept OAE when the number of subjects is given and fixed. With balanced data, conditional methods such as CAAE are expected to perform well. In the presence of class imbalance, however, random process-based OAE has an advantage due to its generalizability.

Algorithm parameters. The latent space dimension was chosen as $d_{\mathcal{Z}} = 8$. The encoder-decoder architecture had 6.1M parameters, and the discriminator had 265k parameters. $\lambda_1 = 10$, $\lambda_2 = 10$, and $\lambda = 10$ were set. All models were trained for 100 epochs with mini-batch size 100, with learning rates of 0.01 for the encoder-decoder and 0.005 for the discriminator which were manually halved at the 30th and 50th epochs. The network architectures for CAAE and WAE were kept mostly the same as random-intercept OAE, save the random-intercept components.

Evaluation measures. Similar to the VGGFace2 experiment, the sharpness of generated images were measured. To compare the class-conditional generation quality, class-conditional samples were generated from random-intercept OAE and CAAE. Then, the classification accuracy of the generated digits, measured by a pre-trained deep MNIST digit classifier with 99.2% accuracy, was calculated. Additionally, the diversity of the generated samples per class were measured and compared via structural similarity (SSIM), which is a perceptual similarity metric range between 0 and 1 (Wang et al., 2004; Odena et al., 2017). The mean SSIM score of 50 randomly chosen image pairs conditioned on each digit was evaluated, and the average of the digit-wise mean SSIM scores was recorded.

	Balaced training data			Imbalanced training data		
	Accuracy	SSIM	Sharpness	Accuracy	SSIM	Sharpness
Random-intercept OAE	0.992	0.318	2×10^{-2}	0.972	0.320	2×10^{-2}
CAAE	0.877	0.224	4×10^{-2}	0.839	0.190	4×10^{-2}
WAE	-	-	4×10^{-2}	-	-	4×10^{-2}
Testset	0.999	0.235	1×10^{-1}	0.999	0.235	1×10^{-1}

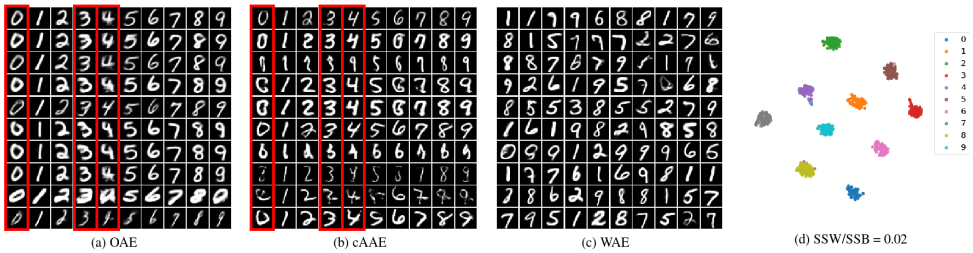
Table 3.2: MNIST performance measures.

Balanced training data. A balanced training dataset with 10 classes of 56,000 images and a balanced test dataset with 10 classes of 1,000 images were used in this experiment. For comparison, 10 classes of 1,000 images were generated from CAAE and random-intercept OAE, and 10,000 images were generated from WAE by ignoring classes. The accuracy shown in Table 3.2 suggests that random-intercept OAE mostly generated correct digits whereas CAAE sometimes failed. The diversity of generated samples were similar.

Imbalanced training data. An imbalanced dataset was created by dropping 90% of images in randomly chosen three classes (digits of 0, 3, and 4) from the balanced training set. The resulting set had 10 classes, 40,933 images. Table 3.2 reveals that the accuracy gap between random-intercept OAE and CAAE for the generated samples widened in the imbalanced setting.

One-shot exemplar generation from imbalanced training of MNIST. Figure 3.4, panels (a) and (b) illustrate that random-intercept OAE generated class-conditioned variations successfully even for the minority classes (0, 3, and 4) for which CAAE produced incorrect results. Note that WAE cannot generate class-conditional distributions, resulting in mixed digits (Figure 3.4 (c)).

Unit-level disentanglement in representation from MNIST. Figure 3.4 (d) shows the TSNE map of the latent space representation of the test images obtained from the encoder trained with imbalanced data. The well-separated digits in the plot explain the high accuracy of the random-intercept OAE even with imbalanced data; the ratio SSW/SSB is also very small, close to the ideal value of $\sigma_0^2/\tau_0^2 = 0.01$.



(a) OAE (b) cAAE (c) WAE (d) SSW/SSB = 0.02

Figure 3.4: One-shot exemplar generation from imbalanced training of MNIST.

3.3 Appendix

Implementation details for Section 3.2 are provided in this section.

3.3.1 Architectures

VGGFace2 Table 3.3, 3.4 and 3.5 summarize the details of the random-intercept OAE architecture trained for the VGGFace2 data. The images were transformed range from [0,225] to [-1,1]. A hyperbolic tangent activation was used for the decoder output. Except for the layers related to the random intercept, the CAAE and WAE architectures are mostly the same as random-intercept OAE; CAAE required additional input for the label information for both encoder and decoder.

Layer	Operation	Filters	Kernel	Strides	Batch norm	Activation	Linked layer
1	Convolution	128	5x5	2x2	Yes	ReLU	Input
2	Convolution	256	5x5	2x2	Yes	ReLU	1
3	Convolution	256	5x5	2x2	Yes	ReLU	2
4	Convolution	256	4x4	2x2	Yes	ReLU	3
5	Dense	512	-	-	Yes	ReLU	4
6	Dense	256	-	-	Yes	ReLU	5
7	Dense	128	-	-	Yes	ReLU	6
8	Dense	256	-	-	Yes	ReLU	7
μ	Dense	128	-	-	No	Linear	8
9	Dense	256	-	-	Yes	ReLU	7
σ^2	Dense	128	-	-	No	Linear	9
10	Dense	256	-	-	Yes	ReLU	7
ν	Dense	128	-	-	No	Linear	10
Output $(B\|X_0)$	Sample $B\|X_0$	-	-	-	-	-	ν
Output $(Z\|B, X_0)$	Sample $Z\|B, X_0$	-	-	-	-	-	μ, σ^2, $B\|X_0$

Table 3.3: VGGFace2 encoder pair $(Q_{Z|B,X_0}, Q_{B|X_0})$; $d_{\mathcal{Z}} = 128$

Layer	Operation	Filters	Kernel	Strides	Batch norm	Activation	Linked layer
1	Dense	256	-	-	Yes	ReLU	Input
2	Dense	512	-	-	Yes	ReLU	1
3	Dense	8x8x128	-	-	No	ReLU	2
4	Reshape to (8,8,128)	-	-	-	-	-	3
5	Transpose Convolution	256	5x5	2x2	Yes	ReLU	4
6	Transpose Convolution	256	5x5	2x2	Yes	ReLU	5
7	Transpose Convolution	128	5x5	2x2	Yes	ReLU	6
8	Convolution	64	5x5	1x1	Yes	ReLU	7
Output	Convolution	3	5x5	1x1	No	Hyperbolic tangent	8

Table 3.4: VGGFace2 decoder g

Layer	Operation	Filters	Kernel	Strides	Batch norm	Activation	Linked layer
1	Dense	2048	-	-	No	ReLU	Input
2	Dense	2048	-	-	No	ReLU	1
3	Dense	2048	-	-	No	ReLU	2
4	Dense	2048	-	-	No	ReLU	3
Output	Dense	1	-	-	No	Sigmoid	4

Table 3.5: VGGFace2 discriminator f

MNIST For MNIST, Table 3.6, 3.7 and 3.8 provide the details of the random-intercept OAE architecture. A sigmoid activation was used for the decoder because the image was transformed range from [0,225] to [0,1].

Layer	Operation	Filters	Kernel	Strides	Batch norm	Activation	Linked layer
1	Convolution	64	3x3	2x2	Yes	ReLU	Input
2	Convolution	128	3x3	2x2	Yes	ReLU	1
3	Convolution	256	3x3	2x2	Yes	ReLU	2
4	Convolution	512	4x4	4x4	Yes	ReLU	3
5	Convolution	1024	1x1	1x1	Yes	ReLU	4
μ	Dense	8	-	-	No	Linear	5
σ^2	Dense	8	-	-	No	Linear	5
ν	Dense	8	-	-	No	Linear	5
Output $(B\|X_0)$	Sample $B\|X_0$	-	-	-	-	-	ν
Output $(Z\|B, X_0)$	Sample $Z\|B, X_0$	-	-	-	-	-	μ, σ^2, $B\|X_0$

Table 3.6: MNIST encoder pair $(Q_{Z|B,X_0}, Q_{B|X_0})$; $d_{\mathcal{Z}} = 8$

Layer	Operation	Filters	Kernel	Strides	Batch norm	Activation	Linked layer
1	Dense	7x7x512	-	-	No	ReLU	Input
2	Reshape to (7,7,512)	-	-	-	-	-	1
3	Transpose Convolution	256	4x4	2x2	Yes	ReLU	2
4	Transpose Convolution	128	4x4	2x2	Yes	ReLU	3
Output	Convolution	1	4x4	1x1	No	Sigmoid	4

Table 3.7: MNIST decoder g

Layer	Operation	Filters	Kernel	Strides	Batch norm	Activation	Linked layer
1	Dense	256	-	-	No	ReLU	Input
2	Dense	256	-	-	No	ReLU	1
3	Dense	256	-	-	No	ReLU	2
4	Dense	256	-	-	No	ReLU	3
5	Dense	256	-	-	No	ReLU	4
Output	Dense	1	-	-	No	Sigmoid	5

Table 3.8: MNIST discriminator f

A toy model Table 3.9, 3.10 and 3.11 provide the details of the random-intercept OAE architecture for the four-dimensional toy model in Section 3.2.

Layer	Operation	Filters	Batch norm	Activation	Linked layer
1	Dense	32	Yes	ReLU	Input
2	Dense	32	Yes	ReLU	1
3	Dense	32	Yes	ReLU	2
μ	Dense	2	No	Linear	3
σ^2	Dense	2	No	Linear	3
ν	Dense	2	No	Linear	3
Output $(B\|X_0)$	Sample $B\|X_0$	-	-	-	ν
Output $(Z\|B, X_0)$	Sample $Z\|B, X_0$	-	-	-	μ, σ^2, $B\|X_0$

Table 3.9: Toy model encoder pair $(Q_{Z|B,X_0}, Q_{B|X_0})$; $d_{\mathcal{Z}} = 2$

Layer	Operation	Filters	Batch norm	Activation	Linked layer
1	Dense	2	No	ReLU	Input
Output	Dense	4	No	Linear	1

Table 3.10: Toy model decoder g

Layer	Operation	Filters	Batch norm	Activation	Linked layer
1	Dense	30	No	ReLU	Input
2	Dense	30	No	ReLU	1
3	Dense	30	No	ReLU	2
Output	Dense	1	No	Sigmoid	3

Table 3.11: Toy model discriminator f

Chapter 4

Product-Space Ornstein Auto-Encoders

4.1 Issues with random-intercept OAE

Despite the advantages of the OAE discussed in Chapter 2, the approach of the random-intercept OAE in Chapter 3 comes with several problems. First, the constraints in problem (3.1) do not fully incorporate the assumptions of the model. First of all, the objective in (3.1) is only an upper bound of $\mathcal{D}_{\text{OAE}}(P_{\mathbf{X}}, P_{\mathbf{Y}}; g, P_{\mathbf{Z}})$. Assume a process distribution $P_{\mathbf{X}}$ on $\mathcal{X}^\infty$ is exchangeable and the random variable B deconvolves $\mathbf{X}$ so that $P_{\mathbf{X_{1:n}},B} = \left[\prod_{j=1}^{n} P_{X_0|B}\right] P_B$ for any n. For any $Q_{\mathbf{Y}|\mathbf{Z}} \in \mathcal{Q}^*_{\text{det}}$ and $P_{\mathbf{Y}} = \int Q_{\mathbf{Y}|\mathbf{Z}} dP_{\mathbf{Z}} = P_{\mathbf{Z}} g^{-1}$, the random-intercept OAE uses

$$\mathcal{D}_{\text{RIOAE}}(P_{\mathbf{X}}, P_{\mathbf{Y}}; g, P_{\mathbf{Z}}) = \inf_{Q_{\mathbf{Z}|\mathbf{X},B} \in \mathcal{Q}^{RI}} \mathrm{E}_{P_{\mathbf{X},B}} \mathrm{E}_{Q_{\mathbf{Z}|\mathbf{X},B}} d^p(X_0, g(Z_0)).$$

Here, $\mathcal{Q}^{RI}$ is the set of all conditional distributions $Q_{\mathbf{Z}|\mathbf{X},B}$ such that the joint distribution $P_{\mathbf{X},B}Q_{\mathbf{Z}|\mathbf{X},B}$ of $(\mathbf{X}, \mathbf{Z}, B)$ has the marginal $\big[\prod_{j=1}^{n} Q_{Z_0|X_0,B}\big]P_{\mathbf{X}_{1:n},B}$ on $(\mathbf{X}_{1:n}, \mathbf{Z}_{1:n}, B)$ for any n, and $\int_{\mathcal{X}} Q_{Z_0|X_0,B}dP_{X_0|B} = P_{Z_0|B}$. For any $Q_{\mathbf{Z}|\mathbf{X},B} \in \mathcal{Q}^{RI}$, its marginal on $\mathbf{Z}_{1:n}$ is

$$
\begin{aligned}
&\int_{\mathcal{X}^n\times\mathcal{Z}} \big[\prod_{j=1}^{n} Q_{Z_0|X_0,B}\big] dP_{\mathbf{X}_{1:n},B} \\
&= \int_{\mathcal{X}^n\times\mathcal{Z}} \big[\prod_{j=1}^{n} Q_{Z_0|X_0,B}\big] d\big[P_B \prod_{j=1}^{n} P_{X_0|B}\big] \\
&= \int_{\mathcal{Z}} \prod_{j=1}^{n} \big[\int_{\mathcal{X}} Q_{Z_0|X_0,B}dP_{X_0|B}\big] dP_B \\
&= \int_{\mathcal{Z}} \big[\prod_{j=1}^{n} P_{Z_0|B}\big] dP_B \\
&= P_{\mathbf{Z}_{1:n}},
\end{aligned}
$$

for any n. Thus, $Q_{\mathbf{Z}|\mathbf{X},B} \in \mathcal{Q}$ and $\mathcal{Q} \supset \mathcal{Q}^{RI}$. It follows that

$$
\mathcal{D}_{\mathrm{OAE}}(P_{\mathbf{X}}, P_{\mathbf{Y}}; g, P_{\mathbf{Z}}) \leq \mathcal{D}_{\mathrm{RIOAE}}(P_{\mathbf{X}}, P_{\mathbf{Y}}; g, P_{\mathbf{Z}}). \tag{4.1}
$$

Furthermore, the independence and additivity assumptions of the random intercept model are not imposed directly. Second, the constraint $Q_{Z_0|B} = P_{Z_0|B}$ is imposed *for all observational units*, i.e., for all possible values of B. Because of this excessive number of constraints, there has to be a sufficient number of observations with diverse variation for every unit to ensure successful training, which would be an exceptional feature for real data. Lastly, the formulation (3.1) does not fully utilize the exchangeable nature of the data. The random-intercept OAE assumes that there exists a random variable B conditioned on which the coordinates of $\mathbf{X}$ are i.i.d. and imposes the constraint $Q_{B|\mathbf{X}}P_{\mathbf{X}} = P_{\mathbf{X},B}$. How-

ever, there is no mechanism in the formulation (3.1) that promotes this. The penalty $\mathcal{D}_2(P_B, \int_{\mathcal{X}^\infty} Q_{B|\mathbf{X}} dP_{\mathbf{X}})$ from formulation (3.1), which relaxes the constraint $P_B = \int_{\mathcal{X}^\infty} Q_{B|\mathbf{X}} dP_{\mathbf{X}}$, cannot achieve that constraint. This means that the random-intercept OAE cannot achieve the upper bound $\mathcal{D}_{\text{RIOAE}}$ in (4.1) by solving (3.1) even if the relaxation is tight, except when the trained "identity encoder" $Q_{B|\mathbf{X}}$ satisfies that $Q_{B|\mathbf{X}} P_{\mathbf{X}} = P_{\mathbf{X},B}$.

4.2 Product-space model for latent space

The random intercept model (1.1) is not the only way to imposing exchangeability to a sequence. More importantly, the additive structure of this model may limit the expressiveness of the entire generative model. A more flexible approach is to decompose $\mathcal{Z}$ into $\mathcal{I} \times \mathcal{V}$. Suppose P_B on $\mathcal{I}$ and P_{E_0} on $\mathcal{V}$. Now, specify $P_{\mathbf{Z}}$ on $\mathcal{Z}^\infty$ by

$$Z_j^i = (B^i, E_j^i), \quad B^i \overset{\text{i.i.d.}}{\sim} P_B, \; E_j^i \overset{\text{i.i.d.}}{\sim} P_{E_0}, \quad B^i \perp\!\!\!\perp E_j^i. \tag{4.2}$$

Ideally, B^i encodes the "identity" of the observational unit shared among the coordinates of sequence $\mathbf{Z}^i$, and E_j^i encodes the "within-unit variation" shared among all observational units. Clearly $\mathbf{Z}^i$ is exchangeable. Furthermore, the sequence $(g(B^i, E_1^i), g(B^i, E_2^i), \dots)$ is exchangeable for any function $g : \mathcal{I} \times \mathcal{V} \to \mathcal{X}$. Additivity, for example, is absorbed into the decoder and can be learned from data. We call this approach to OAE the *product-space OAE*.

The key advantage of the product-space OAE over the random-intercept OAE is that it directly optimizes the upper bound (2.14). To see this, assume that $P_{\mathbf{X}}$ on $\mathcal{X}^\infty$ is exchangeable, and the random variable B on $\mathcal{I}$ deconvolves

$\mathbf{X}$ so that $P_{\mathbf{X_{1:n}},B} = \big[\prod_{j=1}^{n} P_{X_0|B}\big] P_B$ for any n. Recall formulation (2.14):

$$\mathcal{D}_{\mathrm{OAE}}(P_{\mathbf{X}}, P_{\mathbf{Y}}; \mathbf{g}, P_{\mathbf{Z}}) = \inf_{Q_{\mathbf{Z}|\mathbf{X},B} \in \mathcal{Q}} \mathrm{E}_{P_{\mathbf{X},B}} \mathrm{E}_{Q_{\mathbf{Z}|\mathbf{X},B}} \mathrm{E}_{Q_{\mathbf{Y}|\mathbf{Z}}} d^p(X_0, Y_0).$$

where $P_{\mathbf{Y}} = \int Q_{\mathbf{Y}|\mathbf{Z}} dP_{\mathbf{Z}} = P_{\mathbf{Z}}\mathbf{g}^{-1}$. Again, $\mathcal{Q}$ is the set of all conditional distributions $Q_{\mathbf{Z}|\mathbf{X},B}$ such that the joint distribution $Q_{\mathbf{Z}|\mathbf{X},B} P_{\mathbf{X},B}$ of $(\mathbf{X}, \mathbf{Z}, B)$ has marginals $\big[\prod_{j=1}^{n} Q_{Z_0|X_0,B}\big] P_{\mathbf{X}_{1:n},B}$ and $P_{\mathbf{Z_{1:n}}}$ on $(\mathbf{X}_{1:n}, \mathbf{Z}_{1:n}, B)$ and $\mathbf{Z}_{1:n}$ for any n, respectively. Note that any $Q_{Z_0|X_0,B}$ fully parameterizes $\mathcal{Q}$ and under model (4.2), $Q_{Z_0|X_0,B} = Q_{B,E_0|X_0,B} = Q_{B|X_0,B} Q_{E_0|X_0,B}$, where $Q_{B|X_0,B}(\cdot|x_0, b)$ is the Dirac measure on b for any $b \in \mathcal{I}$. From the construction (4.2), for $b \in \mathcal{I}$ and $e_j \in \mathcal{V}$, $j = 1, ..., n$,

$$P_{\mathbf{Z_{1:n}}}((b, e_1), (b, e_2), ..., (b, e_n)) = \big[\prod_{j=1}^{n} P_{E_0|B}(e_j|b)\big] P_B(b)$$

for any n. But from (2.14), $\int \big[\prod_{j=1}^{n} Q_{Z_0|X_0,B}\big] dP_{\mathbf{X}_{1:n},B}$ can be formulated by

$$\begin{aligned}
&P_{\mathbf{Z_{1:n}}}((b, e_1), (b, e_2), ..., (b, e_n)) \\
&= \int_{\mathcal{X}^n \times \mathcal{I}} \big[\prod_{j=1}^{n} Q_{Z_0|X_0,B}((b', e_j)|x_j, b)\big] dP_{\mathbf{X}_{1:n},B}(x_1, \cdots, x_n, b') \\
&= \int_{\mathcal{I}} \int_{\mathcal{X}^n} \big[\prod_{j=1}^{n} Q_{Z_0|X_0,B}((b', e_j)|x_j, b)\big] dP_{\mathbf{X}_{1:n}|B}(x_1, ..., x_n|b')\, dP_B(b') \\
&= \int_{\mathcal{I}} \left[\prod_{j=1}^{n} \int_{\mathcal{X}} Q_{Z_0|X_0,B}((b', e_j)|x_j, b) dP_{X_0|B}(x_j|b')\right] dP_B(b') \\
&= \int_{\mathcal{I}} \left[\prod_{j=1}^{n} \int_{\mathcal{X}} Q_{B|X_0,B}(b'|x_j, b) Q_{E_0|X_0,B}(e_j|x_j, b) dP_{X_0|B}(x_j|b') P_{X_0|B}(x_j|b')\right] dP_B(b') \\
&= \left[\prod_{j=1}^{n} \int_{\mathcal{X}} Q_{E_0|X_0,B}(e_j|x_j, b) dP_{X_0|B}(x_j|b)\right] P_B(b)
\end{aligned}$$

for all n, since $Q_{B|X_0,B}$ is the Dirac measure on b. Thus

$$\int_{\mathcal{X}} Q_{E_0|X_0,B} dP_{X_0|B} = P_{E_0} \quad \text{a.s.} \tag{4.3}$$

and $Q_{E_0|X_0,B}$ fully parameterizes $\mathcal{Q}$ under the constraint (4.3) and that B satisfies $P_{\mathbf{X_{1:n}},B} = \left[\prod_{j=1}^{n} P_{X_0|B}\right] P_B$ for all n. The latter condition well-defines the conditional distributions $P_{B|\mathbf{X}}$ such that $P_{B|\mathbf{X}_{1:n}} P_{\mathbf{X}_{1:n}} = P_{\mathbf{X}_{1:n},B}$ for all n. Thus, $D_{\mathrm{OAE}}(P_{\mathbf{X}}, P_{\mathbf{Y}}; P_{\mathbf{Z}})$ has the formulation in terms of a pair of encoders $(Q_{E_0|X_0,B}, Q_{B|\mathbf{X}})$:

$$\begin{aligned}
&\mathcal{D}_{\mathrm{OAE}}(P_{\mathbf{X}}, P_{\mathbf{Y}}; P_{\mathbf{Z}}) \\
&= \inf_{Q_{\mathbf{Z}|\mathbf{X},B} \in \mathcal{Q}} \mathrm{E}_{P_{\mathbf{X},B}} \mathrm{E}_{Q_{\mathbf{Z}|\mathbf{X},B}} \mathrm{E}_{Q_{\mathbf{Y}|\mathbf{Z}}} d^p(X_0, Y_0), \\
&= \inf_{Q_{E_0|X_0,B} \in \mathcal{Q}_{E_0}} \mathrm{E}_{P_{\mathbf{X}}} \mathrm{E}_{Q_{B|\mathbf{X}}} \mathrm{E}_{Q_{E_0|X_0,B}} d^p(X_0, g(B, E_0)),
\end{aligned}$$

where $\mathcal{Q}_{E_0} = \{Q_{E_0|X_0,B} : \int_{\mathcal{X}} Q_{E_0|X_0,B} dP_{X_0|B} = P_{E_0}\}$.

To ease computation, we consider further relaxation. From $P_{E_0|B} = P_{E_0}$, we have

$$\begin{aligned}
P_{E_0} &= \int_{\mathcal{I}} P_{E_0} dP_B \\
&= \int_{\mathcal{I}} \int_{\mathcal{X}} Q_{E_0|X_0,B} dP_{X_0|B} dP_B \\
&= \int_{\mathcal{X}\times\mathcal{I}} Q_{E_0|X_0,B} dP_{X_0,B}.
\end{aligned}$$

The resulting version of the product-space OAE is then

$$\begin{aligned}
&\inf_{g} \inf_{Q_{B|\mathbf{X}}} \inf_{Q_{E_0|B,X_0}} \big[\mathrm{E}_{P_{\mathbf{X}}} \mathrm{E}_{Q_{B|\mathbf{X}}} \mathrm{E}_{Q_{E_0|B,X_0}} d^p(X_0, g(B, E_0)) \\
&\quad + \lambda_1 \mathcal{D}_B(P_B, \textstyle\int_{\mathcal{X}^\infty} Q_{B|\mathbf{X}} dP_{\mathbf{X}}) + \lambda_2 \mathcal{D}_{E_0}(P_{E_0}, \textstyle\int_{\mathcal{X}\times\mathcal{I}} Q_{E_0|X_0,B} dP_{X_0,B}),
\end{aligned} \tag{4.4}$$

for appropriate choices of the divergence measures $\mathcal{D}_B$ and $\mathcal{D}_{E_0}$.

4.3 Training product-space OAE

Alternating optimization Empirically, the following alternating optimization scheme is effective in training the product-space OAE, especially for challenging datasets like VGGFace2: 1) Fix the parameters of the "within-unit variation encoder" $Q_{E_0|B,X_0}$, and update the parameters of the "identity encoder" $Q_{B|\mathbf{X}}$ and decoder g until the infimand of problem (4.4) no longer changes. 2) Fix the parameters of the identity encoder and update the parameters of $Q_{E_0|B,X_0}$ and decoder g until the infimand of problem (4.4) no longer changes. 3) Repeat steps 1 and 2 until the parameters of the encoder pair $(Q_{B|\mathbf{X}}, Q_{E_0|B,X_0})$ and decoder g converge.

Similar to the random-intercept OAE in Section 3.1, the product-space OAE uses the $\mathcal{D}_{GAN}$ for $\mathcal{D}_{E_0}$ and MMD for $\mathcal{D}_B$. The identity encoder $Q_{B|X_0}$ takes only a single coordinate as its input and yields "identity variables" $\hat{b}^i_j \sim Q_{B|X_0}(\cdot|x^i_j)$ for each $j = 1, \ldots, m_i$. In order to obtain $\hat{b}^i \sim Q_{B|\mathbf{X}_{1:m_i}}(\cdot|\{x^i_j\}_{j=1}^{m_i})$, we aggregate $\hat{b}^i_j$'s so that $\hat{b}^i = \frac{1}{m_i}\sum_{j=1}^{m_i} \hat{b}^i_j$, as used in Lines 5 of Algorithm 2. We also sample $\hat{e}^i_j$ independently from the within-unit variation encoder $Q_{E_0|B,X_0}(\cdot|\hat{b}^i, x^i_j)$ given this $\hat{b}^i$ and the observations x^i_j of unit i, for $j = 1, \ldots, m_i$. This mean-field-like approximation to $Q_{B|\mathbf{X}}$ may harm the independence of the encode variables B^i and E^i_j. It is empirically observed that adding an additional penalty based on the Hilbert-Schmidt Independence Criterion (HSIC, Gretton et al., 2005; Lopez et al., 2018) greatly improves the training.

$$\begin{aligned}&\mathrm{HSIC}_{\iota,\vartheta}(Q_{B,E_0})\\ &= \left\| \mathrm{E}_{Q_{B,E_0}} \left(\left[\iota(\cdot, B) - \mathrm{E}_{P_B}\iota(\cdot, B)\right] \otimes \left[\vartheta(\cdot, E_0) - \mathrm{E}_{P_{E_0}}\vartheta(\cdot, E_0)\right] \right) \right\|^2_{\mathrm{HS}},\end{aligned}$$

for positive definite kernels $\iota : \mathcal{I} \times \mathcal{I} \to \mathbb{R}$ and $\vartheta : \mathcal{V} \times \mathcal{V} \to \mathbb{R}$ that respectively induce reproducing kernel Hilbert spaces $\mathcal{H}_\iota$ and $\mathcal{H}_\vartheta$; $\otimes$ denotes the tensor product and $\|C\|^2_{\mathrm{HS}}$ is the squared Hilbert-Schmidt norm (the sum of the squared singular values) of the cross-covariance operator C. This independence criterion is effective in training the product-space OAE, especially when dataset has a small number of samples per each unit. For example, VGGFace2 has 362.6 number of images per person on average, which is small compared to 10,000 number of images per digit in MNIST.

The resulting training algorithm is summarized as Algorithm 2.

Initialization. In practice, random variable B only deconvolves the output process $\mathbf{Y}$ but not necessarily the input $\mathbf{X}$. Ideally, the latent variable B should encode the "identity" of the samples of an observational unit and make the coordinates of input $\mathbf{X}$ conditionally i.i.d. Thus, encoder $Q_{B|\mathbf{X}}$ should be some smoothed version of a classifier (Olshen, 1974). Any sensible classifier of the training data can be fit and used as the "initial value" of $Q_{B|\mathbf{X}}$. For example, features from the last hidden layer of the pre-trained ResNet classifier (Cao et al., 2018) can be employed for VGGFace2.

Algorithm 2 Product-space OAE training

Input: Exchangeable sequences $(x_1^i, ..., x_{n_i}^i)$ for $i = 1, ..., L$

Output: Encoder pair $(Q_{B|X_0}, Q_{E_0|B,X_0})$ and decoder g

Require: P_B, P_{E_0}, regularization coefficients λ_1, λ_2 and λ_3, positive definite kernels ι, ϑ

1: **while** $Q_{B|X_0}$, $Q_{E_0|B,X_0}$, f, g not converge **do**

2: **while** $Q_{B|X_0}$, g not converged **do**

3: Sample units $i = 1, ..., n$ and sequence $(x_1^i, ..., x_m^i)$ for each unit i

4: Sample b^i from P_B and $(e_1^i, ..., e_m^i)$ from P_{E_0} for all $i = 1, ..., n$

5: Sample $\hat{b}_j^i \sim Q_{B|X_0}(\cdot|x_j^i)$ for each $j = 1, ..., m$ and aggregate $\hat{b}^i = \frac{1}{m}\sum_{j=1}^m \hat{b}_j^i$ for $i = 1, ..., n$.

6: Sample $(\hat{e}_1^i, ..., \hat{e}_m^i)$ from $Q_{E_0|B,X_0}$ given $\hat{b}^i$ and $(x_1^i, ..., x_m^i)$ for all $i = 1, ..., n$

7: Update $Q_{B|X_0}$ and g by descending:

$$\frac{1}{nm}\sum_{i=1}^n\sum_{j=1}^m d^p(x_j^i, g(\hat{b}^i, \hat{e}_j^i)) + \frac{\lambda_1}{n(n-1)}\sum_{i\neq l}\kappa(b^i, b^l) + \frac{\lambda_1}{n(n-1)}\sum_{i\neq l}\kappa(\hat{b}^i, \hat{b}^l)$$

$$-\frac{2\lambda_1}{n^2}\sum_{i,l}\kappa(b^i, \hat{b}^l) + \frac{\lambda_3}{(nm)^2}\sum_{i,j}^n\sum_{q,r}^m \iota(\hat{b}_q^i, \hat{b}_r^j)\vartheta(\hat{e}_q^i, \hat{e}_r^j)$$

$$+\frac{\lambda_3}{(nm)^4}\sum_{i,j,k,l}^n\sum_{q,r,v,w}^m \iota(\hat{b}_q^i, \hat{b}_r^j)\vartheta(\hat{e}_v^k, \hat{e}_w^l) - \frac{\lambda_3}{(nm)^3}\sum_{i,j,k}^n\sum_{q,r,v}^m \iota(\hat{b}_q^i, \hat{b}_r^j)\vartheta(\hat{e}_q^i, \hat{e}_v^k)$$

8: **end while**

9: **while** $Q_{E_0|B,X_0}$, f, g not converged **do**

10: Repeat 3 - 6

11: Update $Q_{E_0|B,X_0}$ and g by descending:

$$\frac{1}{nm}\sum_{i=1}^{n}\sum_{j=1}^{m} d^p(x_j^i, g(\hat{b}^i, \hat{e}_j^i)) - \frac{\lambda_2}{nm}\sum_{i=1}^{n}\sum_{j=1}^{m}\log f(\hat{e}_j^i)$$

$$+\frac{\lambda_3}{(nm)^2}\sum_{i,j}^{n}\sum_{q,r}^{m}\iota(\hat{b}_q^i, \hat{b}_r^j)\vartheta(\hat{e}_q^i, \hat{e}_r^j) + \frac{\lambda_3}{(nm)^4}\sum_{i,j,k,l}^{n}\sum_{q,r,v,w}^{m}\iota(\hat{b}_q^i, \hat{b}_r^j)\vartheta(\hat{e}_v^k, \hat{e}_w^l)$$

$$-\frac{\lambda_3}{(nm)^3}\sum_{i,j,k}^{n}\sum_{q,r,v}^{m}\iota(\hat{b}_q^i, \hat{b}_r^j)\vartheta(\hat{e}_q^i, \hat{e}_v^k)$$

12: Update f by ascending: $\sum_{i=1}^{n}\sum_{j=1}^{m}\log f(e_j^i) + \log(1 - f(\tilde{e}_j^i))$

13: **end while**

14: **end while**

4.4 Empirical results

The performance of the product-space OAE with the proposed training method was assessed for the three tasks discussed in Section 1.1. **Exemplar generation**: Given a few observations $(x_j^i)_{j=1}^m$ from a new unit i, sample the "identity variables" $(\hat{b}_j^i)_{j=1}^m$ from $Q_{B|X_0}(\cdot|x_j^i)$ to take an average $\hat{b}^i = \frac{1}{m}\sum_{j=1}^m \hat{b}_j^i$. Draw a "within-unit variation" e_j from P_{E_0}. An exemplar is the reconstruction $g(\hat{b}^i, e_j)$. If $m = 1$, this task is called one-shot exemplar generation. **Style transfer**: From observations $(x_l^k)_{l=1}^L$ of another unit $k \neq i$, sample $\hat{b}^k$ as in exemplar generation. Draw "within-unit variation" $\hat{e}_l^k$ from $Q_{E_0|B,X_0}(\cdot|\hat{b}^k, x_l^k)$. Then sequence $(g(\hat{b}^i, \hat{e}_l^k))_{l=1}^L$ transfers the style of unit k to i. If $m = 1$, this task is called one-shot style transfer. **Unit generation**: In order to generate a new unit not in the data, sample b^{new} from P_B and sequence (e_j^{new}) from P_{E_0} i.i.d. Then pass $((b^{\text{new}}, e_j^{\text{new}}))$ to the decoder g. In addition, the representation power of the "identity variables" can be considered by the **prototype image**: for unit i, compute $\hat{b}^i$ as in exemplar generation. Then $g(\hat{b}^i, \mu_{E_0})$ is the prototype image of unit i, where μ_{E_0} is the mean of E_0.

For all the experiments, $\mathcal{X} = \mathbb{R}^{d_\mathcal{X}}$ and $\mathcal{Z} = \mathbb{R}^{d_\mathcal{Z}}$ with Euclidean metric $d(x, x') = ||x - x'||_2$. The prior distribution $P_\mathbf{Z}$ of the latent variable $\mathbf{Z}$ follows model (4.2). The independent standard normal prior $P_{E_0} = \mathcal{N}(0, \mathbf{I}_{d_\mathcal{V}})$ and $P_B = \mathcal{N}(0, \mathbf{I}_{d_\mathcal{I}})$ were set over $\mathcal{I} = \mathbb{R}^{d_\mathcal{I}}$ and $\mathcal{V} = \mathbb{R}^{d_\mathcal{V}}$, respectively. The identity encoder $Q_{B|X_0}$ and the within-unit variation encoder $Q_{E_0|B,X_0}$ were also designed as Gaussian:

$$B^i|\{X_0 = x_j^i\} \overset{\text{iid}}{\sim} \mathcal{N}(\mu_B(x_j^i), \sigma_B^2(x_j^i)\mathbf{I}_{d_\mathcal{I}}),$$
$$E_j^i|\{B = b^i, X_0 = x_j^i\} \overset{\text{iid}}{\sim} \mathcal{N}(\mu_E(x_j^i, b^i), \sigma_E^2(x_j^i, b^i)\mathbf{I}_{d_\mathcal{V}}),$$

with the mean functions $\mu_B : \mathcal{X} \to \mathcal{I}$, $\mu_E : \mathcal{X} \times \mathcal{I} \to \mathcal{V}$ and the variance functions

$\sigma_B^2 : \mathcal{X} \to \mathbb{R}_{++}$, $\sigma_E^2 : \mathcal{X} \times \mathcal{I} \to \mathbb{R}_{++}$. For the VGGFace2 experiments, μ_B and σ_B^2 were initialized with a pre-trained classifier (Cao et al., 2018). The μ_E and σ_E were designed to share most of the network to prevent overfitting. The quality of generated sample was compared with WAE, in which observational units are not preserved. Within-unit sample generations was compared with the random-intercept OAE proposed in Chapter 3.

If all units are present in training data, the quality of samples were also compared with CAAE, by interpreting each unit as a class. Further implementation details are given in Section 4.6.

4.4.1 Imbalanced MNIST

For the MNIST data, randomly selected 40,357 images were used for training, and the rest were used for testing. In order to impose imbalance in the training data, 90% of the training images of digits of 1, 2, and 6 were removed.

Generated images and t-SNE maps of the within-unit variation in the latent space are shown in Figure 4.1. As a reference, samples from the prior distribution are plotted in translucent blue dots in panel B. For CAAE and the random-intercept OAE, the distribution of the encoded within-unit variation shows non-ideal clustering. In particular, the digit 1 (orange), which is a minority class, is distinctly clustered. A similar phenomenon can be observed for the digit 2 (green). This is an indication of training instability of the random-intercept OAE, leading to the poor quality in the prototype images of digits 2 and 6 in panel A. Because of the class imbalance, CAAE also shows poor quality in the prototype images of digit 1 and 2. In panel C, for each method, the first column carries the prototype images from one observation. The rest of columns correspond to newly generated variations of the prototypes. Dashed red lines represent the minority classes, i.e., the digits 1, 2, and 6. Product-space

	Accuracy	One-shot accuracy	SSIM	Sharpness
WAE	-	-	-	0.047(±0.003)
CAAE	0.860(±0.008)	-	0.244(±0.020)	0.041(±0.007)
Random-intercept OAE	0.919(±0.033)	0.873(±0.059)	0.292(±0.017)	0.025(±0.004)
Product-space OAE	0.939(±0.019)	0.878(±0.029)	0.263(±0.008)	0.032(±0.008)
Testset	0.994(±0.002)	-	0.229(±0.009)	0.075(±0.004)

Table 4.1: Imbalanced MNIST performance measures.

OAE shows the best performance in matching the distribution of prior and the within-unit variation, and quality of within-unit sample generation for minority class.

Table 4.1 reports several measures of reconstruction quality, averaged over 10 repetitions of 100 exemplar generations.[1] (One-shot) accuracy measures the classification accuracy of an MNIST-trained deep digit classifier for five (one) generated images per digit. Also, the structural similarity (SSIM) (Wang et al., 2004; Odena et al., 2017) and sharpness measured by using the Laplace filter (Tolstikhin et al., 2018) are provided to assess the per-image quality of the generated digits. Samples from CAAE and both OAEs indicate similar quality to the unconditionally generated ones from WAE. As for the recognizability of the generated digits, both OAEs report higher accuracy than CAAE. Noticeably, the (class-given generation) accuracy of CAAE is even lower than the one-shot accuracy of OAEs. Between the two OAEs, the product-space OAE outperforms by all metrics.

[1]Standard deviations are also provided in the parentheses.

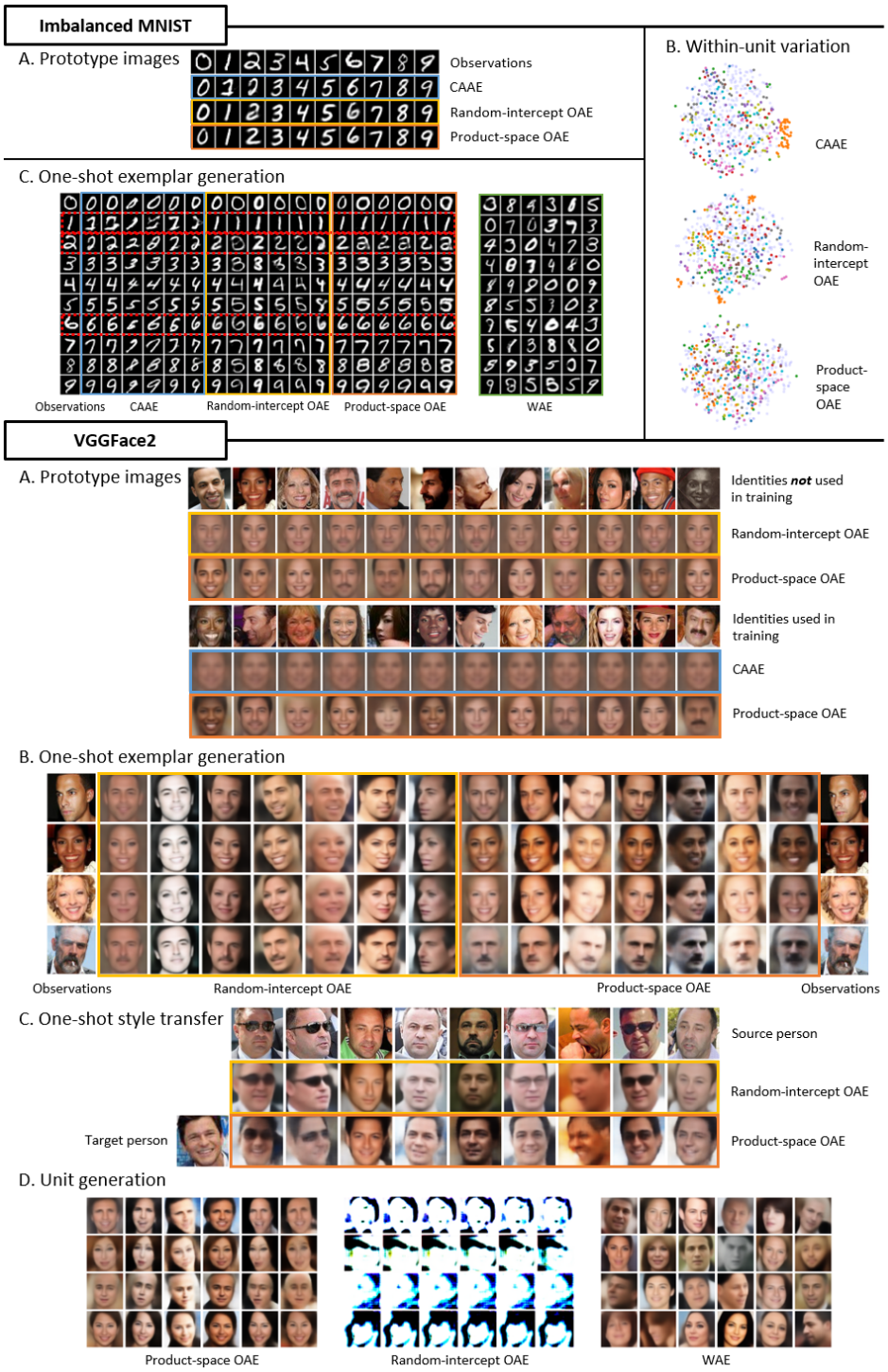


Figure 4.1: Sample generation from imbalanced MNIST and VGGFace2.

4.4.2 VGGFace2

For the VGGFace2 dataset, each of the face images was cropped and rescaled to a common size of 128 by 128 pixels. The training set consisted of 2,513,512 images from 8,631 randomly chosen people. The test set for people used during training consisted of the 628,378 images from those 8,631 people. Another test set is comprised of 169,396 images from 500 randomly chosen people not already included in the training set.

The generated images are given in Figure 4.1. Panel A shows that CAAE failed to preserve the identity of the people *used* in training. This is likely because of the large number of classes (identities) in the data and severe class imbalance. On the other hand, the product-space OAE successfully preserves the identities of the input images in the prototypes, regardles of whether or not the input identity is known to the model. In this task, the random-intercept OAE was not as good as the product-space OAE. The superior "identity-preservation" performance of the product-space OAE carries over to the tasks of one-shot exemplar generation (panel B) and one-shot style-transfer (panel C). Note that each row of panel B should look like the same person, and the last row of panel C should look like the target person, while copying the poses of the corresponding image above in the first row. Remarkably, the product-space OAE also succeeded in generating completely new identities (neither in the training set nor the test set) with shared pose variation (panel D). The random-intercept OAE failed, and class-conditional models like CAAE are not capable of this task. WAE may generate new identities, but are not able to create their systematic variations.

In Table 4.2, the quality of one-shot exemplar generation is quantified by the inception score (IS) (Salimans et al., 2016), the sharpness of the generative

	Identities used in training			Identities not used in training		
	IS	FID	Sharpness	IS	FID	Sharpness
WAE	–	-	-	2.125(±0.016)	106.250(±3.024)	0.001(±0.000)
CAAE	2.029(±0.010)	115.767(±2.796)	0.001(±0.000)	-	-	-
Random-intercept OAE	2.068(±0.011)	107.961(±2.371)	0.001(±0.000)	2.067(±0.020)	102.476(±3.363)	0.001(±0.000)
Product-space OAE	2.146(±0.104)	98.525(±2.487)	0.001(±0.000)	2.125(±0.118)	94.287(±2.323)	0.001(±0.000)
Testset	3.883(±0.146)	-	0.004(±0.001)	3.807(±0.164)	-	0.003(±0.001)

Table 4.2: VGGFace2 performance measures.

images, and the Frechet inception distance (FID) between the generated images distribution and the original image distributions (Heusel et al., 2017). These measures were averaged over 10 repetitions of 30 exemplar generations. For each unit, 300 samples were generated. All measures favor OAE over WAE and CAAE. For FID, the product-space OAE shows the best result, which implies the data generated from the latter is closer to the original data than the random-intercept OAE.

4.5 Discussion

To learn the correlation structure inherently residing in the exchangeable data, OAE applies the exchangeable model to the latent space. It is shown that OAE can provide disentangled representations, i.e., latent variables that are well-clustered by subjects. OAE has successfully demonstrated high performance in three types of identity-preserving generation tasks that have been advocated in assessing the quality of generative models, namely exemplar generation, style transfer, and unit generation. To the best of author's knowledge, this is the only framework to date that can perform all of the three tasks successfully. Also, OAE is robust to data imbalance compare to other conditional LVMs. This capability has potential applications in classification and recognition as well.

A natural next step is to go beyond the exchangeability assumption, in tasks like generating faces of an individual person with age variation. Exploring domain generalization would also be a promising future avenue. By considering the domain as an observational unit, we may be able to disentangle the domain effect within the latent space.

4.6 Appendix

This section provides the implementation details and additional experiments.

4.6.1 Implementation details

Conditional adversarial auto-encoders When all of the observational units were present in the training data, the quality of samples from the product-space OAE were compared with CAAE, by interpreting each unit as a class. For the CAAE, the conditional Gaussian latent variables were set:

$$Z_j^i | \{C^i = k\} \overset{\text{iid}}{\sim} \mathcal{N}(0, \mathbf{I}_{d_{\mathcal{Z}}}),$$

for $k = 1, \ldots, K$ where C is a given class label of X_0 and K is the number of subjects. For each class $k = 1, \ldots, K$, the encoder $Q_{Z_0|X_0,C}$ of CAAE was designed to be a Gaussian encoder:

$$Z_j^i | \{X_0 = x_j^i, C^i = k\} \sim \mathcal{N}\big(\mu_k(x^i, c^i), \sigma_k^2(x^i, c^i)\mathbf{I}_{d_{\mathcal{Z}}}\big),$$

where $\mu_k : \mathcal{X} \times \{1, \ldots, K\} \to \mathcal{Z}$, $\sigma_k^2 : \mathcal{X} \times \{1, \ldots, K\} \to \mathbb{R}_{++}$ are parameterized by a deep neural network. The decoder $g : \mathcal{Z} \times \{1, \ldots, K\} \to \mathcal{X}$ was also parameterized by a deep neural network.

Random-intercept OAE The implementations of the random-intercept OAE followed the recipe in Section 3.2. The prior distribution P_{Z_0} was a random intercept model with Gaussian noise:

$$Z_j^i | \{B^i = b^i\} \overset{\text{iid}}{\sim} \mathcal{N}(b^i, \mathbf{I}_{d_{\mathcal{Z}}}), \quad B^i \overset{\text{iid}}{\sim} \mathcal{N}(0, 100\mathbf{I}_{d_{\mathcal{Z}}}),$$

and the encoder pair $(Q_{Z_0|B,X_0}, Q_{B|X_0})$ was designed to be a random-intercept Gaussian encoder pair:

$$Z_j^i|\{B=\tilde{b}^i, X_0=x_j^i\} \overset{\text{iid}}{\sim} \mathcal{N}(\mu(x_j^i)+\tilde{b}^i, \sigma^2(x_j^i)\mathbf{I}), \quad B|\{X_0=x_j^i\} \overset{\text{iid}}{\sim} \mathcal{N}(\nu(x_j^i), \tau^2\mathbf{I}),$$

where the mean functions $\mu : \mathcal{X} \to \mathcal{Z}$, $\nu : \mathcal{X} \to \mathcal{Z}$, the variance function $\sigma^2 : \mathcal{X} \to \mathbb{R}_{++}$, and the decoder $g : \mathcal{Z} \to \mathcal{X}$ were parameterized by deep neural networks. The hyperparameter τ was kept small.

4.6.2 Architectures

For every convolutional layer used in the networks, padding and truncated normal initialization were applied.

Imbalanced MNIST Tables 4.3, 4.4, and 4.5 provide the details of the architecture of the product-space OAE. "Batch norm" indicates whether there was a batch normalization layer (Ioffe and Szegedy, 2015) included. A sigmoid activation was used to decode the image range transformed from [0,225] to [0,1]. The network architectures for CAAE, WAE, and the random-intercept OAE were almost the same as the product-space OAE, except for the output layers of the encoder pair and the input of the decoder; CAAE required additional input for the one-hot encoding of the label information for both encoder and decoder. The encoder-decoder architecture had 1M parameters, and the discriminator architecture had 17k parameters.

Layer	Operation	Filters	Kernel	Strides	Batch norm	Activation	Linked layer
1	Convolution	64	4x4	2x2	Yes	ReLU	Input
2	Convolution	64	4x4	1x1	Yes	ReLU	1
3	Convolution	128	4x4	2x2	Yes	ReLU	2
4	Convolution	128	4x4	1x1	Yes	ReLU	3
5	Dense	64	-	-	Yes	ReLU	4
6	Dense	32	-	-	Yes	ReLU	5
7	Dense	16	-	-	Yes	ReLU	6
μ_B	Dense	8	-	-	No	Linear	7
σ_B^2	Dense	8	-	-	No	Linear	7
8	Dense	32	-	-	Yes	ReLU	5
μ_E	Dense	8	-	-	No	Linear	8, $B\|X_0$
σ_E^2	Dense	8	-	-	No	Linear	8, $B\|X_0$
Output $(B\|X_0)$	Sample $B\|X_0$	-	-	-	-	-	μ_B, σ_B^2
Output $(E_0\|B, X_0)$	Sample $E_0\|B, X_0$	-	-	-	-	-	μ_E, σ_E^2

Table 4.3: MNIST encoder pair $(Q_{E_0|B,X_0}, Q_{B|X_0})$. $d_{\mathcal{I}} = 8$ and $d_{\mathcal{V}} = 8$

.

Layer	Operation	Filters	Kernel	Strides	Batch norm	Activation	Linked layer
1	Dense	7x7x128	-	-	No	ReLU	Input
2	Reshape to (7,7,128)	-	-	-	-	-	1
3	Transpose Convolution	64	4x4	2x2	Yes	ReLU	2
4	Transpose Convolution	32	4x4	2x2	Yes	ReLU	3
5	Transpose Convolution	16	4x4	1x1	Yes	ReLU	4
Output	Convolution	1	4x4	1x1	No	Sigmoid	5

Table 4.4: MNIST decoder g

Layer	Operation	Filters	Kernel	Strides	Batch norm	Activation	Linked layer
1	Dense	64	-	-	No	ReLU	Input
2	Dense	64	-	-	No	ReLU	1
3	Dense	64	-	-	No	ReLU	2
4	Dense	64	-	-	No	ReLU	3
5	Dense	64	-	-	No	ReLU	4
Output	Dense	1	-	-	No	Sigmoid	5

Table 4.5: MNIST discriminator f

VGGFace2 Tables 4.6 through 4.9 summarize the details of the product-space OAE architecture trained for the VGGFace2 data. For each image, the pixel value range was transformed from [0, 225] to [-1,1]. A hyperbolic tangent activation was used for the decoder output. For product-space OAE, features from the 2,048 dimensional last hidden layer of a pre-trained VGGFace2 classifier (Cao et al., 2018) were employed as input to the identity encoder $Q_{B|X_0}$ (Table 4.6). This pre-trained VGGFace2 classifier employed a ResNet-50-based architecture (He et al., 2016) with squeeze-and-excitation (SE) blocks (Hu et al., 2018). It was trained with a training set comprised of 8631 identities. Except for the input layer and the last layer, the architectures of the encoders and decoders from the CAAE, WAE, and the random-intercept OAE were primarily the same as the within-unit encoder and the decoder from product-space OAE, respectively; CAAE required additional input for the one-hot encoding of the label information for both the encoder and decoder. Aside from CAAE, the encoder-decoder architecture had 19M parameters. CAAE had an encoder-decoder architecture with a larger 24M parameters, mainly due to the added information pertaining to the 8,631 number of classes. The discriminator architectures had 855K parameters.

Layer	Operation	Filters	Kernel	Strides	Batch norm	Activation	Linked layer
1	Dense	384	-	-	Yes	ReLU	Input
2	Dense	256	-	-	Yes	ReLU	1
μ_B	Dense	64	-	-	No	Linear	2
σ_B^2	Dense	64	-	-	No	Linear	2
Output $(B\|X_0)$	Sample $B\|X_0$	-	-	-	-	-	μ_B, σ_B^2

Table 4.6: VGGFace2 identity encoder $Q_{B|X_0}$; $d_{\mathcal{I}} = 64$

Layer	Operation	Filters	Kernel	Strides	Batch norm	Activation	Linked layer
1	Convolution	64	5x5	2x2	Yes	ReLU	Input
2	Convolution	128	5x5	2x2	Yes	ReLU	1
3	Convolution	256	5x5	2x2	Yes	ReLU	2
4	Convolution	512	5x5	2x2	Yes	ReLU	3
5	Convolution	256	3x3	2x2	Yes	ReLU	4
6	Dense	128	-	-	Yes	ReLU	5
7	Dense	256	-	-	Yes	ReLU	6, $B\|X_0$
8	Dense	128	-	-	Yes	ReLU	6
μ_E	Dense	128	-	-	No	Linear	8
σ_E^2	Dense	128	-	-	No	Linear	8
Output $(E_0\|B, X_0)$	Sample $E_0\|B, X_0$	-	-	-	-	-	μ_E, σ_E^2

Table 4.7: VGGFace2 within-unit encoder $Q_{E|B,X_0}$; $d_{\mathcal{V}} = 128$

Layer	Operation	Filters	Kernel	Strides	Batch norm	Activation	Linked layer
1	Dense	8x8x512	-	-	Yes	ReLU	Input
2	Reshape to (8,8,512)	-	-	-	-	-	1
3	Transpose Convolution	256	5x5	2x2	Yes	ReLU	2
4	Transpose Convolution	128	5x5	2x2	Yes	ReLU	3
5	Transpose Convolution	64	5x5	2x2	Yes	ReLU	4
6	Transpose Convolution	32	5x5	2x2	Yes	ReLU	5
7	Convolution	32	5x5	1x1	Yes	ReLU	6
Output	Convolution	3	3x3	1x1	No	Hyperbolic tangent	7

Table 4.8: VGGFace2 decoder g

Layer	Operation	Filters	Kernel	Strides	Batch norm	Activation	Linked layer
1	Dense	512	-	-	No	ReLU	Input
2	Dense	512	-	-	No	ReLU	1
3	Dense	512	-	-	No	ReLU	2
4	Dense	512	-	-	No	ReLU	3
Output	Dense	1	-	-	No	Sigmoid	4

Table 4.9: VGGFace2 discriminator f

4.6.3 Training details

The Adam optimizer (Kingma and Ba, 2014) was used to train the model, with $\beta_1 = 0.9$ for updating the first moment estimate and $\beta_2 = 0.999$ for updating the second moment estimate.

Details of the imbalanced MNIST training All models were trained for 10,000 iterations with mini-batch of size 600 with no need of alternating optimization. The models were updated with the learning rates of 0.001 for the encoder-decoder pair and 0.0005 for the discriminator. Both learning rates were decayed by multiplying 1/1.0001 after every 100 iterations. $\lambda_1 = 1$, $\lambda_2 = 1$, and $\lambda_3 = 1$ were set. On average, 100 iterations took 6 seconds.

Details of the VGGFace2 training First, the model was trained with alternating optimization: 1) Fix the parameters of the within-unit variation encoder $Q_{E_0|B,X_0}$ and train the identity encoder $Q_{B|X_0}$ and decoder g for 10,000 iterations with $\lambda_1 = 100$, $\lambda_2 = 0$, and $\lambda_3 = 1000$; 2) Fix the parameters of the identity encoder $Q_{B|X_0}$ and train the identity encoder $Q_{E_0|B,X_0}$ and decoder g for 10,000 iterations with $\lambda_1 = 0$, $\lambda_2 = 100$, and $\lambda_3 = 1,000$. Steps 1 and 2 were repeated for 15 times, then the model was fine-tuned without alternating optimization for 30,000 iterations with $\lambda_1 = 100$, $\lambda_2 = 100$, and $\lambda_3 = 1,000$. For a total of 330,000 iterations, the mini-batches with a size of 600 were used for training. The learning rates were 0.001 for the encoder-decoder and 0.001 for the discriminator. On average, 100 iterations took 151 seconds.

Computing infrastructure For the imbalanced MNIST dataset, a single model was trained with 5 Intel(R) Xeon(R) CPU Silver 4114 @ 2.20GHz processors and one NVIDIA TITAN V GPUs which had 5120 CUDA cores, 640

tensor cores, and 12GB memory. For the VGGFace2 experiments, we trained a single model with 18 CPU processes and 4 NVIDIA TITAN V GPUs. All of the implementations were based on Python 3.6, Tensorflow 1.15.0 and Keras 2.3.1.

4.6.4 Additional figures from the VGGFace2 experiment

This section presents additional figures from the VGGFace2 experiment in Section 4.4.2. Figure 4.2 compares the quality of sample generation for people *not used in training*. Additional generated samples are shown in panels A and B. Panels C, D, and E show the t-SNE maps of the latent variables, identity variable, and the within-unit variation in the representation (latent) space, respectively. The product-space OAE shows the best quality in separating observational units in the representation space, while WAE could not provide meaningful separation in the representation. For the product-space OAE, the t-SNE map of the encoded identity (panel D, same person is plotted in same color) shows better clustering power than the random-intercept OAE. In panel E, the distribution of the encoded within-unit variation of the product-space OAE matches well with the reference samples from the prior distribution, which are plotted in translucent blue dots. Figure 4.2 shows additional generated images and the representations of the people who *were* used in training. Comparing Figure 4.2 with Figure 4.3, the quality of the within-unit face generation of the people not in the training set was similar with that of the people who were in the training set, which implies that both OAEs were well-generalized. The product-space OAE shows superior identity-preservation performance for both known and unknown people. Panel B also shows that CAAE failed to preserve the identity of the people *used in training*.

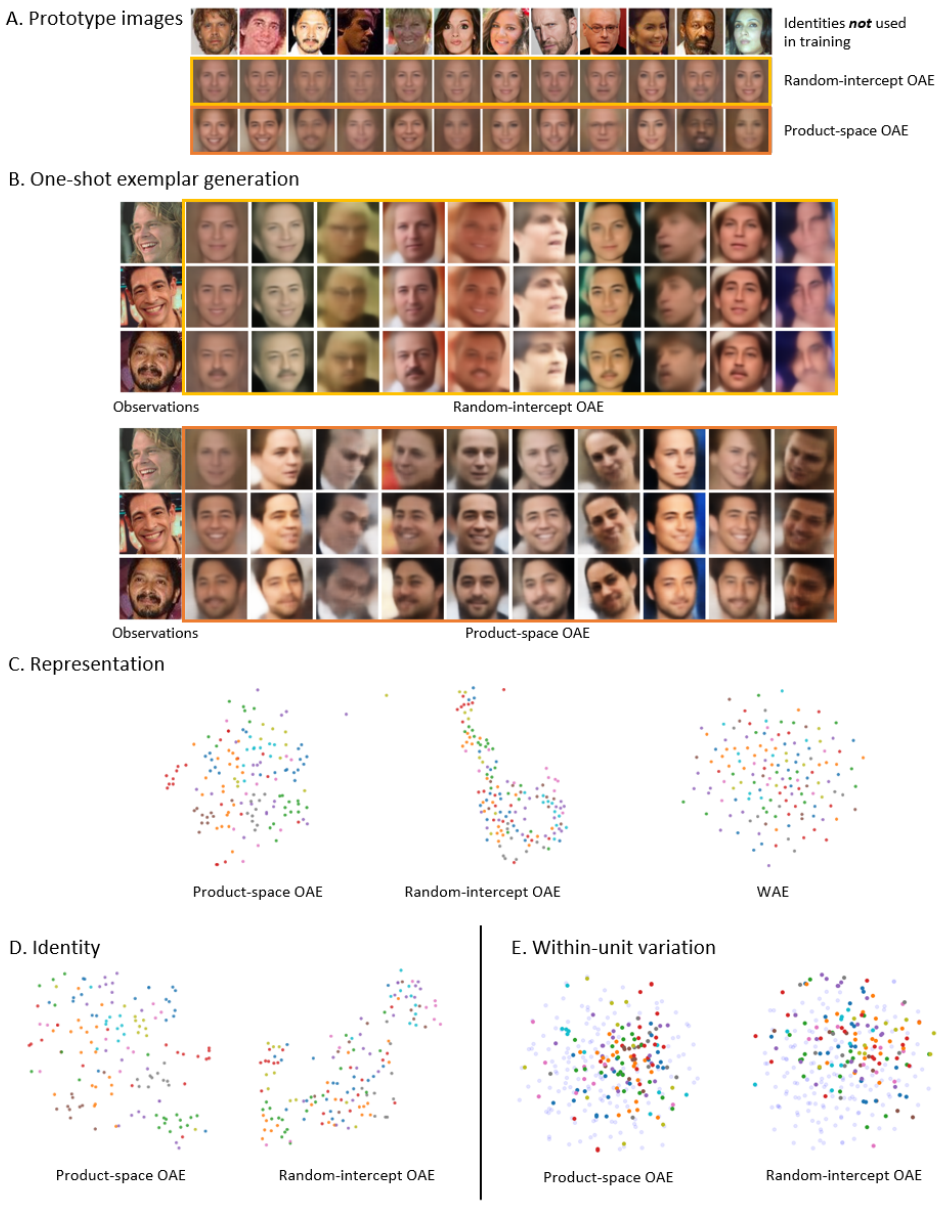


Figure 4.2: Additional sample generation from VGGFace2 (people not used in training)

Figure 4.3: Additional sample generation from VGGFace2 (people used in training)

Part II

Predictive Model for Hierarchically Correlated Data

Chapter 5

DeepBiome

For the second part, this thesis includes a study for learning a predictive model to capture a hierarchical correlation in microbiome taxonomic abundance data. In this study, the main contributions are developing the `DeepBiome` software, and demonstrating the ability of the `DeepBiome` software with the simulation studies. Section 5.1 is a brief introduction to the `DeepBiome` method. The `DeepBiome` software is described in Section 5.2. In Section 5.3, the ability of the developed software is demonstrated by the simulation studies. The `DeepBiome` method development and the application to the real-life data are also discussed in Jing Zhai's thesis (Zhai, 2019).

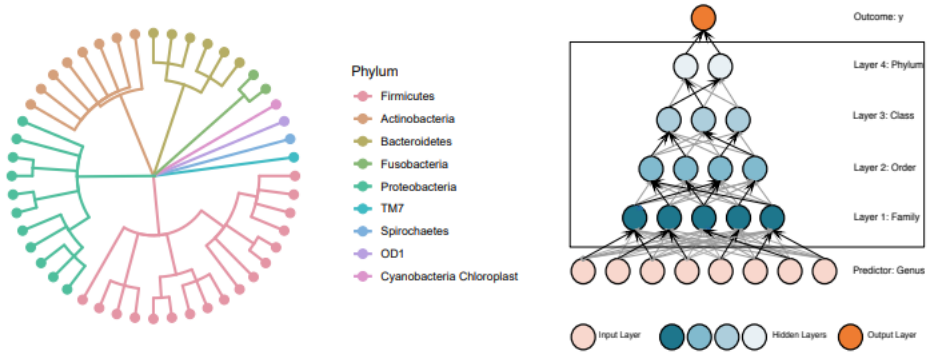
Microbiome data structure In the 16S ribosomal ribonucleic acid (RNA) sequencing, data are grouped into Operational Taxonomic Units (OTUs) according to a similarity threshold, e.g., 97%. These OTUs are then clustered at different phylogenetic depths to build a phylogenetic tree portraying their evolutionary relationships (Schloss and Handelsman, 2005; Turnbaugh et al.,

2007). Suppose we have p OTUs from a total of n microbiome samples and a phylogenetic tree that depicts the evolutionary relationship among microbes. Each OTU is a tip node on the phylogenetic tree, and each internal node is a taxonomic unit representing a common ancestor of its descendent taxa. In this work, we aggregate p OTUs to m genus level taxa as the basic analyzing units. However, the basic analyzing unit can also start at finerlevels. Let $\mathbf{x} = (\mathbf{x}_1, \ldots, \mathbf{x}_n)$, where $\mathbf{x}_i = (x_{i1}, \ldots, x_{im})^T$ indicate the abundance of m genera of the ith subject, be the input data and $\mathbf{y} = (y_1, \ldots, y_n)$ be the outcome of interest. Outcome variables can be continuous, binary, or categorical.

5.1 DeepBiome

`DeepBiome` is a DNN-based predictive model for capturing microbiome signals at different phylogenetic depths. By leveraging the phylogenetic information, `DeepBiome` relieves the heavy burden of tuning for the optimal deep learning architecture, avoids overfitting, and, more importantly, enables visualizing the path from microbiome counts to disease. This model is applicable to both regression and classification problems. It takes microbiome taxonomic abundance data as input.

Phylogeny-informed architecture `DeepBiome` prespecifies the network architecture according to the phylogenetic tree. The number of taxonomic levels decides the number of hidden layers, and the number of taxa at each taxonomic level decides the number of neurons in the corresponding layer. Figure 5.1 illustrates an example `DeepBiome` architecture. The input layer receives microbiome abundances as inputs. The information is then propagated through multiple layers of the `DeepBiome` network to the outcome of interest.



Phylogenetic Level	No. of Taxa
Genus	48
Family	40
Order	23
Class	17
Phylum	9

Hidden Layers	No. of Neuron
Input Layer	$m^{(0)}= 48$
Hidden Layer 1	$m^{(1)}=40$
Hidden Layer 2	$m^{(2)}= 23$
Hidden Layer 3	$m^{(3)}= 17$
Hidden Layer 4	$m^{(4)}= 9$

Figure 5.1: DeepBiome architecture

Phylogeny regularization via weight decay By regularizing the neural network architecture towards the phylogenetic structure, `DeepBiome` greatly reduces the number of parameters. Phylogeny regularization is achieved by weight decay, a popular technique (Mundie and Massengill, 1991; Krogh and Hertz, 1992; Gupta and Lam, 1998) to prevent overfitting and boost performances of DNNs (Zhang et al., 2019). `DeepBiome` incorporates the bacteria evolutionary relationship into a differential weight decay regularization matrix, thus generating an interpretable effect transfer network in modeling and analyzing microbiome data.

Suppose the phylogenetic level $l = 0, ..., L$. If taxa j in level l and k in level $l-1$ have ancestor-descendent relationship, the associations between the corresponding neurons are stronger, then we assume larger weight value w_{jk}^L. When taxa j in level l and k in level $l-1$ do not have this ancestral relationship, we assume w_{jk}^L to be a small value, i.e., weight decay. Thus, we construct a weight decay matrix $\{\omega_{jk}^l\}$ to regularize weights $\{w_{jk}^l\}$ of the neural network using evolutionary relationship carried by the phylogenetic tree. If nodes j and k are ancestor-descendent related, $\omega_{jk}^l = 1$; if not, ω_{jk}^l is a small value, e.g., 0.01.

The predictive model with phylogeny regularization via weight decay is described in Algorithm 3. Adam optimizer, an adaptive gradient algorithm (Kingma and Ba, 2014), is used to train `DeepBiome`.

Here, $m^{(l)}$ is the number of taxa in level l. $m^{(L)}$ is 1 for regression, and K for classification with K categories. $ReLU(a) := max(0, a)$ is the rectified linear unit (ReLU) activations.

Algorithm 3 Predictive model with phylogeny regularization via weight decay

Input: Microbiome abundances $\mathbf{x}^0 \in \mathbb{R}^n \times \mathbb{R}^{m^{(0)}}$

Output: Clinical outcome $\mathbf{y}$

Require: $\mathbf{w}^l \in \mathbb{R}^{m^{(l)}} \times \mathbb{R}^{m^{(l-1)}}$ and $\mathbf{b}^l \in \mathbb{R}^{m^{(l)}}$ for $l = 1, ..., L$.

1: **for** l=1,...,L-1 **do**

$$x_{ij}^l = ReLU(\sum_{k=1}^{m^{(l-1)}} \omega_{jk}^l w_{jk}^l x_{ik}^l + b_j^l)$$

for $i = 1, ..., n$, $j = 1, ..., m^{(l)}$ where

$$\begin{cases} \omega_{jk}^l = 1 \text{ when taxa } j \text{ of level } l \text{ and } k \text{ of level } l-1 \text{ have relationship} \\ \omega_{jk}^l = 0.01 \text{ when taxa } j \text{ of level } l \text{ and } k \text{ of level } l-1 \text{ has no relationship} \end{cases}$$

2: **end for**

3: Predict

$$\begin{aligned} \hat{y}_i &= \mathbf{w}^L \mathbf{x}_i^L + b^L \quad \text{for regression,} \\ \hat{\Pr}(y_i = c) &= \frac{e^{\mathbf{w}_c^L \mathbf{x}_i^L + b_c^L}}{\sum_{j=1}^K e^{\mathbf{w}_j^L \mathbf{x}_i^L + b_j^L}} \quad \text{for classification with } K \text{ categories} \end{aligned}$$

5.2 Software

`DeepBiome` is implemented in Python 3.6-based TensorFlow (Abadi et al., 2015) and Keras (Chollet et al., 2015) frameworks. Both GPUs and CPUs can be used in this frameworks. It is an open-source tool available at `https://github.com/Young-won/DeepBiome`. It can be built on Python 3.4, 3.5, and 3.6. Comprehensive documentation and tutorials are available at `https://young-won.github.io/deepbiome/`. This software has the following features:

- Adam optimization method with phylogeny regularization
- User-friendly interface for training, testing and taxa selection with huge data such as the the American Gut Project (AGP) (McDonald et al., 2018)
- Automated visualization tool that can show the selected taxa on the phylogenetic tree based on the trained weights to infer the microbiome-disease path

The software provides a user-friendly interface. From microbiome abundance data and associated phylogenetic tree dictionary, the reader can train the `DeepBiome` model with a single line – for example:

```
from deepbiome.deepbiome import deepbiome_train
test_eval, train_eval, network = deepbiome_train(log,
                                                 network_info,
                                                 path_info)
```

The phylogeny-informed architecture is automatically generated from the inputs, and the model is trained with phylogeny regularization via weight decay. After training, the test evaluations from k-fold cross validation and the trained

model is provided. DeepBiome software also provides functions for testing and taxa selection from the trained model.

Examples of the visualization of the bacteria-to-disease path detected by DeepBiome is shown in Figure 5.2. This figure is generated by the visualization tool of the DeepBiome software. The blue and red nodes indicate the negative and positive weights of the taxa, respectively. The size of the colored nodes represent the magnitudes of the weights. Black nodes represent non-selected taxa.

5.3 Simulation studies

The capability of the DeepBiome method and developed software is first demonstrated by simulation studies. The results show superior performance of DeepBiome over commonly used tools such as support vector machine (SVM), regression with ℓ_1 (lasso) or $\ell_1 + \ell_2$ (elastic net) penalties, DNN without tree regularization, and DNN with ℓ_1 penalty.

In real-life data analysis, a DeepBiome model trained using the developed software shows higher prediction performance and selects the taxa associated with the disease, matching knowledge from existing clinical research. As an example, application on the American gut project (AGP) using DeepBione software is described in Section 5.5.

Performance metrics We employ several statistical metrics to evaluate the performance of DeepBiome for its prediction, classification, and taxa selection performances. For a quantitative outcome, the primary metric is the mean squared error (MSE). Additionally, the Pearson correlation coefficient ρ of predicted $\hat{y}_i$ and true y_i are reported. For categorical outcomes, i.e., classification problems, we measure their performance using sensitivity (true positive rate

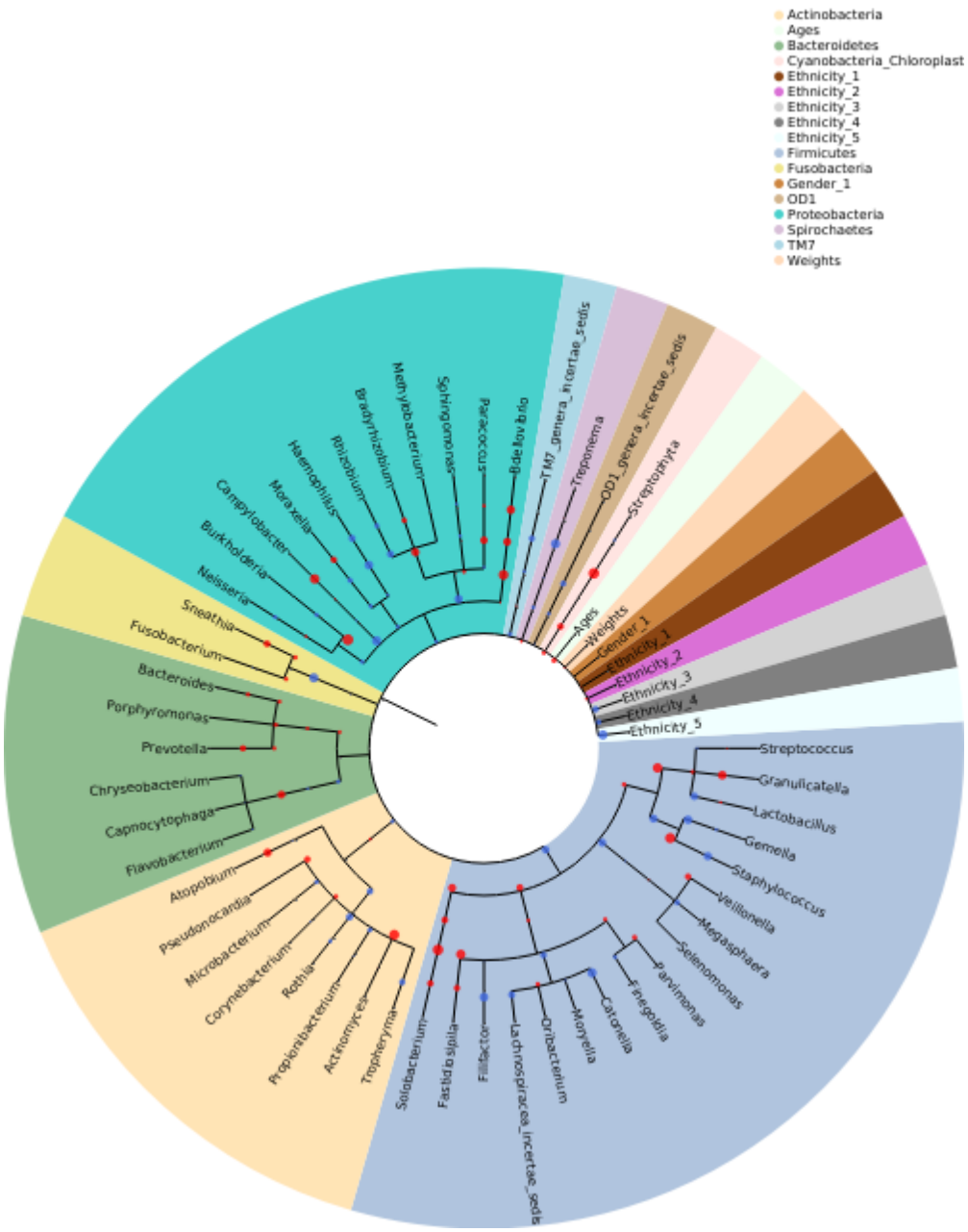


Figure 5.2: DeepBiome visualization tools

(TPR)), specificity, g-measure, accuracy (ACC), precision (PPV), and the $F1$ score:

$$\begin{aligned}
\text{Sensitivity} &= \frac{\text{TP}}{\text{TP+FN}}, \\
\text{Specificity} &= \frac{\text{TN}}{\text{TN+FP}}, \\
g\text{-Measure} &= (\text{Sensitivity} \times \text{Specificity})^{\frac{1}{2}}, \\
\text{ACC} &= \frac{\text{TP+TN}}{\text{TP+TN+FP+FN}}, \\
\text{PPV} &= \frac{\text{TP}}{\text{TP+FP}}, \\
F1 \text{ score} &= 2 \times \frac{\text{PPV} \times \text{TPR}}{\text{PPV+TPR}} = \frac{2\text{TP}}{2\text{TP+FP+FN}},
\end{aligned}$$

where TP is true positive (or recall), TN is true negative, FP is false positive, and FN is false negative. $F1$ score is the harmonic mean of precision and sensitivity. An $F1$ score reaches its best value at 1 when the prediction has perfect precision and recall and the worst value at 0. Note that $F1$ score does not take the true negative cout into account. We use the g-measure, which is the geometric mean of sensitivity and specificity, to assess the performance of a binary classifier. Same as the $F1$ score, a g-measure reaches its best value at 1 when the sensitivity and specificity are both perfect (1) while the worst at 0 if any of sensitivity and specificity is 0. We also report AUC (area under the receiver operating characteristics), which indicates the capability of a model to distinguish between classes. Sensitivity, specificity, g-measure, and ACC across all hidden layers are used to report the selection accuracies.

Simulation studies Extensive simulation studies are provided to demonstrate the ability of `DeepBiome` software. The performance of a `DeepBiome` model trained with the developed software is compared it with conventional methods in three different schemes: linear regression, binary, and multi-

categorical ($K \geq 3$) classification design. Throughout the simulation experiments, the sample size is $n = 1,000$, and samples are split into a training set (75%, $n_{\text{training}} = 750$) and a test set (25%, $n_{\text{test}} = 250$). Different proportions of the data split give qualitatively similar results (not shown). All of the results were obtained based on $1,000$ replicates. Simulation Scenario 1 covers continuous outcome models; simulation Scenario 2 examines for binary outcome cases; and simulation Scenario 3 considers the situation when outcome variable is categorical. Model robustness is evaluated in simulation Scenario 4, examining performance when tree structure is mis-specified and when sequencing abundances contain measurement errors. Details on the simulation is discussed in Section 5.5.

Scenario 1: Regression design

In this section, two simulation strategies were used. In strategy (1), microbiome taxa associated with outcome y are clustered at the phylum level. In strategy (2), the associated taxa are clustered at phylum and order levels .

We compare `DeepBiome` to linear regression, as well as penalized regression with ℓ_1 norm (Lasso), ℓ_2 norm (ridge), and $\ell_1 + \ell_2$ norm (elastic net) penalties. We also compare `DeepBiome` to conventional DNN and ℓ_1-regularized DNN. DNN and ℓ_1-DNN use the same number of hidden layers and neurons on each layer as `DeepBiome` without phylogenetics tree regularization.

Five-fold cross-validation is used to choose the tuning parameters for regularized linear regression models. For the deep learning models (i.e., DNN, ℓ_1-DNN, and `DeepBiome`), a holdout validation set is used for the early-stopping approach. Twenty percent of the training data was used for the holdout validation. Adam optimizer is used for training, with $\beta_1 = 0.9, \beta_2 = 0.999$, learning rate $lr = 0.01$, and mini-batch size of 50. The learning rate decayed for each

epoch with $lr_{epoch+1} = lr_{epoch} \frac{1}{1+0.0001}$.

Table 5.1 displays the predictive performance by two metrics, MSE and Pearson's correlation ρ, under a case that the outcome associated taxa are only clustering at one phylogenetic level (i.e., phylum). The outcome y predicted by `DeepBiome` has higher Pearson correlation and lower MSE on the test set than the regression methods and other deep learning models, which shows that `DeepBiome` has improved prediction performance. Overall, the penalized linear regression models, Lasso and elastic net, have slightly larger correlation coefficients on the test sets compared to linear regression and ridge regression. All deep learning models perform better than regression models, with lower MSE and higher ρ. `DeepBiome` performs the best among the examined deep learning models. Table 5.3 shows the prediction performance under a more complex case, where the outcome-associated taxa are clustered at different phylogenetic levels (phylum and order). It is obvious that all regression schemes perform poorly in this case; the correlation values ρ are only around 0.6. `DeepBiome` has over 80% reduction in MSE compared to regression based methods. The deep learning models DNN and ℓ_1-DNN improve ρ to 0.91 and 0.90 respectively. However, both models show hint of overfitting with lower testing performance. `DeepBiome` consistently achieves the best performances on the test set.

Identifying associated taxa at precise levels is critical for downstream biological validation. Figure 5.3, Tables 5.2 and 5.4 use the metrics sensitivity, specificity, g-measure and ACC to compare the selection performance of different methods. Regular regression methods do not discriminate associated taxa; therefore only the results of penalized regressions were included in Tables 5.2 and 5.4. The penalized regression schemes, Lasso and elastic net, can only select the taxa at one phylogenetic level. Here, we compute the performance metrics based on their phylogeny relationship. For example, if the genus *Prevotella* is se-

Method	Testing				Training			
	MSE		Correlation		MSE		Correlation	
	mean	sd	mean	sd	mean	sd	mean	sd
Linear regression	0.104	0.024	0.824	0.049	0.087	0.011	0.851	0.023
Ridge	0.104	0.022	0.824	0.049	0.09	0.012	0.851	0.023
Lasso	0.100	0.023	0.833	0.048	0.092	0.013	0.843	0.025
Elastic net	0.100	0.023	0.833	0.048	0.092	0.012	0.844	0.025
DNN	0.076	0.040	0.874	0.077	0.032	0.034	0.947	0.067
DNN+ℓ_1	0.075	0.040	0.875	0.073	0.034	0.039	0.945	0.068
DeepBiome	0.071	0.036	0.882	0.069	0.043	0.034	0.929	0.061

Table 5.1: Prediction performance under Scenario 1, strategy (1)

lected, we assume that its corresponding ancestor, the phylum *Bacteroidetes*, is also selected. In contrast, the selection performance of regularized deep learning models are based on the weights estimated at each hidden layer. `DeepBiome` offers outstanding performances not only in terms of sensitivity and specificity, but also g-measure and ACC (Figure 5.3, first row). Its g-measure ranges from 0.8 to 0.9, while those of Lasso regression (the second best method) ranges from 0.54 to 0.72. Interestingly, despite being the second best method regarding prediction (see Table 5.3), ℓ_1-DNN fails to identify the true microbiome taxa across all phylogenetic levels. Overall `DeepBiome` is a consistently efficient model under the regression design in both prediction and selection.

Scenario 2: Binary classification

We consider the case that outcome-associated taxa are clustered at a mixture of phylogenetic levels. For a binary outcome, we suppose

(1) the higher the abundance of blue node taxa, the higher the probability of y belong to the disease group;

(2) the higher the abundance of red node taxa, the higher the probability of

Method	PhyloTree	No. true taxa (total)	Sensitivity		Specificity		g-Measure		ACC	
			mean	sd	mean	sd	mean	sd	mean	sd
Lasso	Genus	31 (48)	0.380	0.136	0.812	0.15	0.536	0.083	0.533	0.064
	Family	23 (40)	0.474	0.150	0.812	0.150	0.602	0.086	0.618	0.065
	Order	9 (23)	0.637	0.169	0.783	0.169	0.688	0.092	0.726	0.084
	Class	7 (17)	0.739	0.161	0.730	0.197	0.715	0.105	0.734	0.099
Elastic net	Genus	31 (48)	0.389	0.138	0.803	0.158	0.540	0.075	0.536	0.063
	Family	23 (40)	0.484	0.149	0.803	0.158	0.605	0.078	0.620	0.063
	Order	9 (23)	0.646	0.159	0.774	0.174	0.691	0.088	0.724	0.088
	Class	7 (17)	0.750	0.149	0.720	0.201	0.717	0.107	0.733	0.104
DNN+ℓ_1	Genus	31 (48)	0.967	0.032	0.034	0.006	0.181	0.016	0.049	0.006
	Family	23 (40)	0.970	0.036	0.031	0.006	0.174	0.017	0.055	0.006
	Order	9 (23)	0.972	0.055	0.026	0.008	0.156	0.026	0.048	0.008
	Class	7 (17)	0.978	0.056	0.021	0.012	0.136	0.047	0.065	0.012
DeepBiome	Genus	31 (48)	0.954	0.042	0.669	0.087	0.797	0.053	0.673	0.085
	Family	23 (40)	0.967	0.037	0.828	0.062	0.894	0.037	0.832	0.060
	Order	9 (23)	0.970	0.058	0.855	0.057	0.910	0.042	0.858	0.056
	Class	7 (17)	0.983	0.050	0.835	0.063	0.905	0.043	0.842	0.060

Table 5.2: Taxa selection performance under Scenario 1, strategy (1)

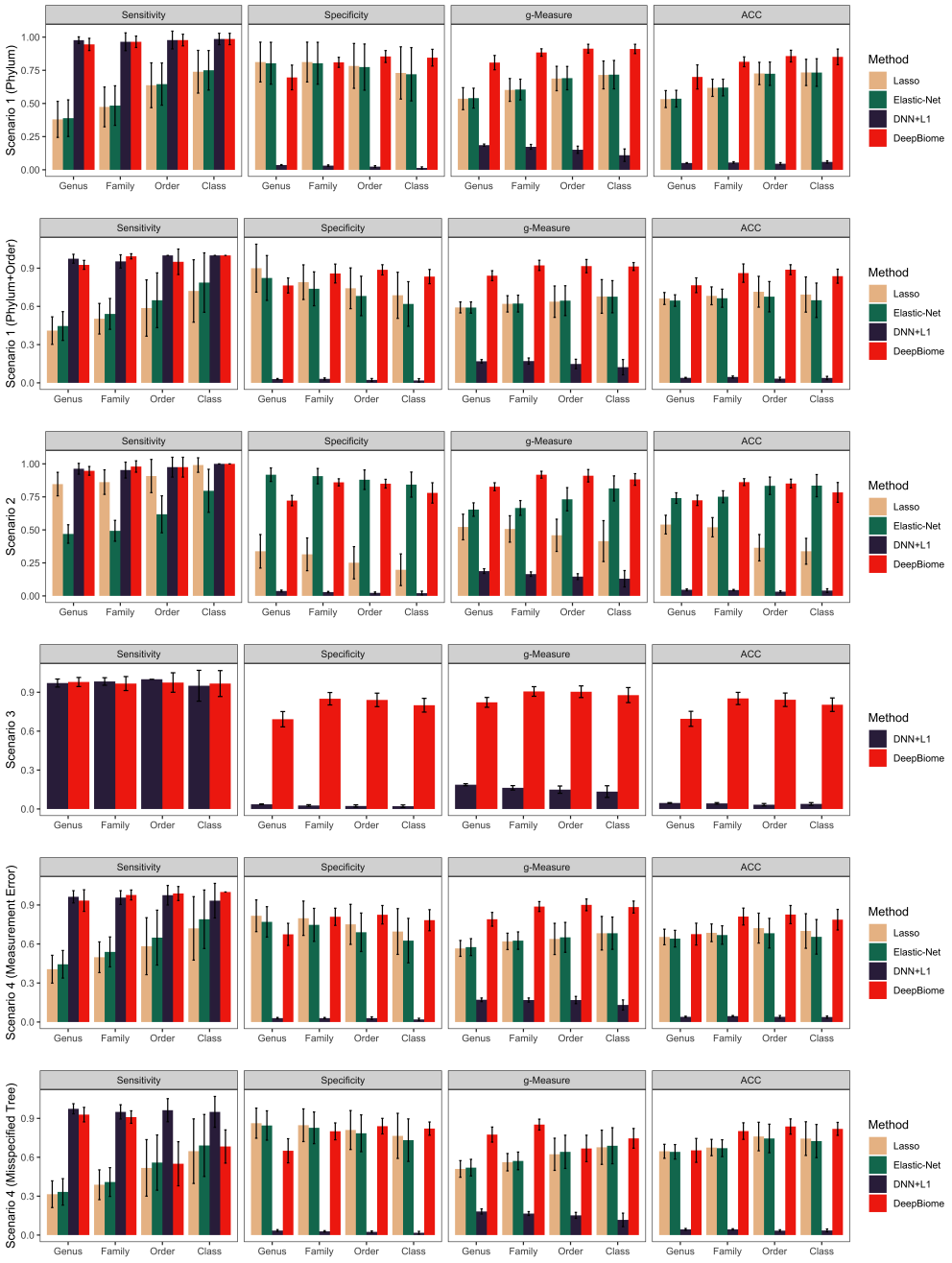


Figure 5.3: Taxa selection performance under 4 simulation schemes at each phylogenetic level

Method	Testing				Training			
	MSE		Correlation		MSE		Correlation	
	mean	sd	mean	sd	mean	sd	mean	sd
Linear regression	1.561	0.146	0.639	0.035	1.337	0.068	0.694	0.018
Lasso	1.479	0.115	0.662	0.034	1.411	0.075	0.678	0.020
Ridge	1.546	0.121	0.639	0.034	1.361	0.075	0.694	0.018
Elastic net	1.481	0.117	0.662	0.034	1.405	0.076	0.680	0.020
DNN	0.457	0.522	0.905	0.118	0.164	0.337	0.964	0.091
DNN+ℓ_1	0.456	0.516	0.904	0.122	0.176	0.362	0.963	0.085
DeepBiome	0.423	1.474	0.916	0.139	0.256	0.463	0.944	0.110

Table 5.3: Prediction performance under Scenario 1, strategy (2)

y belong to the healthy control group.

We compare `DeepBiome` to logistic regression, three penalized logistic regression models, and two conventional deep learning networks. The same learning rate, stopping criteria, and mini-batch size (100) are used for `DeepBiome`, DNN and ℓ_1-DNN. In Table 5.5, we present the metrics for evaluating the classification performance of binary outcome, including sensitivity, specificity, g-measure, ACC, and AUC. Logistic models have satisfactory sensitivity values, but other metrics are not competitive compared to `DeepBiome`. They tend to have many false positives which lead to poor specificity. In contrast, `DeepBiome` achieves the best classification performance with the highest specificity, g-measure, ACC, and AUC reaching 0.84, 0.87, 0.89 and 0.94, respectively. Figure 5.3 (second row) and Table 5.6 displays the performance of identifying the associated taxa. Although Lasso and ℓ_1-DNN show good sensitivity at some phylogenetic levels, g-measure and ACC are much worse compared to elastic net and `DeepBiome`. This suggests that Lasso and ℓ_1-DNN tend to select more taxa (false positive). Using the order level as an example, the g-measure value of `DeepBiome` is 0.91, while the ℓ_1-DNN is 0.15.

Method	PhyloTree	No. true taxa (total)	Sensitivity		Specificity		g-Measure		ACC	
			mean	sd	mean	sd	mean	sd	mean	sd
Lasso	Genus	19 (48)	0.410	0.108	0.900	0.188	0.592	0.043	0.662	0.047
	Family	15 (40)	0.503	0.120	0.790	0.136	0.62	0.064	0.683	0.070
	Order	4 (23)	0.587	0.221	0.742	0.160	0.637	0.123	0.715	0.121
	Class	3 (17)	0.721	0.245	0.687	0.181	0.678	0.132	0.693	0.138
Elastic net	Genus	19 (48)	0.446	0.113	0.823	0.176	0.591	0.044	0.646	0.045
	Family	15 (40)	0.541	0.121	0.738	0.133	0.622	0.066	0.664	0.071
	Order	4 (23)	0.648	0.215	0.682	0.155	0.645	0.117	0.676	0.119
	Class	3 (17)	0.787	0.233	0.619	0.175	0.676	0.126	0.648	0.135
DNN+ℓ_1	Genus	19 (48)	0.967	0.042	0.034	0.007	0.179	0.018	0.043	0.007
	Family	15 (40)	0.972	0.043	0.031	0.006	0.171	0.017	0.046	0.006
	Order	4 (23)	0.972	0.081	0.025	0.008	0.153	0.027	0.034	0.008
	Class	3 (17)	0.982	0.075	0.020	0.012	0.133	0.048	0.039	0.012
DeepBiome	Genus	19 (48)	0.941	0.058	0.686	0.086	0.801	0.054	0.688	0.085
	Family	15 (40)	0.966	0.047	0.820	0.062	0.889	0.040	0.822	0.061
	Order	4 (23)	0.976	0.077	0.835	0.066	0.901	0.054	0.837	0.066
	Class	3 (17)	0.978	0.087	0.784	0.083	0.873	0.064	0.787	0.081

Table 5.4: Taxa selection performance under Scenario 1, strategy (2)

Method	Testing										Training									
	Sensitivity		Specificity		g-Measure		ACC		AUC		Sensitivity		Specificity		g-Measure		ACC		AUC	
	mean	sd	mean	sd	mean	sd	mean	sd	mean	sd	mean	sd	mean	sd	mean	sd	mean	sd	mean	sd
Logistic	0.920	0.030	0.725	0.066	0.815	0.038	0.860	0.026	0.822	0.033	0.950	0.021	0.785	0.049	0.863	0.030	0.900	0.020	0.867	0.026
Lasso	0.965	0.016	0.583	0.086	0.747	0.056	0.848	0.027	0.774	0.041	0.973	0.006	0.619	0.081	0.774	0.053	0.866	0.023	0.796	0.040
Ridge	0.957	0.018	0.461	0.077	0.661	0.055	0.805	0.027	0.709	0.037	0.970	0.007	0.522	0.062	0.711	0.043	0.835	0.016	0.746	0.030
ElasticNet	0.998	0.004	0.022	0.020	0.121	0.082	0.699	0.030	0.510	0.010	0.998	0.002	0.022	0.013	0.140	0.046	0.702	0.015	0.510	0.006
DNN	0.887	0.049	0.725	0.148	0.788	0.133	0.837	0.038	0.897	0.039	0.932	0.032	0.869	0.158	0.887	0.146	0.913	0.043	0.958	0.033
DNN+ℓ_1	0.887	0.048	0.727	0.146	0.790	0.129	0.838	0.038	0.898	0.036	0.931	0.031	0.868	0.154	0.887	0.141	0.912	0.042	0.958	0.030
DeepBiome	0.918	0.042	0.835	0.111	0.870	0.093	0.892	0.044	0.941	0.051	0.952	0.033	0.922	0.106	0.932	0.094	0.943	0.041	0.974	0.048

Table 5.5: Prediction performance under Scenario 2

Method	PhyloTree	No. true taxa (total)	Sensitivity		Specificity		g-Measure		ACC	
			mean	sd	mean	sd	mean	sd	mean	sd
Lasso	Genus	19 (48)	0.847	0.089	0.338	0.127	0.522	0.097	0.540	0.071
	Family	15 (40)	0.862	0.093	0.315	0.124	0.507	0.100	0.520	0.073
	Order	4 (23)	0.908	0.126	0.251	0.122	0.459	0.122	0.365	0.100
	Class	3 (17)	0.991	0.055	0.198	0.120	0.415	0.155	0.338	0.098
Elastic-Net	Genus	19 (48)	0.469	0.070	0.918	0.051	0.654	0.051	0.741	0.040
	Family	15 (40)	0.493	0.079	0.907	0.058	0.666	0.056	0.751	0.045
	Order	4 (23)	0.618	0.140	0.880	0.074	0.732	0.088	0.834	0.066
	Class	3 (17)	0.796	0.163	0.843	0.095	0.814	0.095	0.835	0.084
DNN+ℓ_1	Genus	19 (48)	0.970	0.040	0.036	0.005	0.187	0.013	0.045	0.005
	Family	15 (40)	0.969	0.045	0.029	0.006	0.167	0.017	0.045	0.006
	Order	4 (23)	0.977	0.074	0.024	0.008	0.152	0.026	0.034	0.008
	Class	3 (17)	0.979	0.082	0.020	0.012	0.130	0.048	0.039	0.011
DeepBiome	Genus	19 (48)	0.964	0.045	0.721	0.068	0.832	0.042	0.724	0.067
	Family	15 (40)	0.969	0.045	0.865	0.048	0.915	0.034	0.867	0.047
	Order	4 (23)	0.974	0.081	0.851	0.057	0.909	0.053	0.852	0.057
	Class	3 (17)	0.978	0.083	0.796	0.077	0.880	0.059	0.800	0.075

Table 5.6: Taxa selection performance under Scenario 2

Scenario 3: Multicategory classification

We simulate multi-category outcomes, e.g., the severity of illness, which may be categorized as "mild", "moderate", or "severe". Consistent with previous simulations, we assume that the blue node taxa contribute to the "severe" group, red node taxa contribute to the "mild" group, and part of the gray node taxa contribute to the neutral "moderate" group.

Table 5.7 presents the evaluation metrics for `DeepBiome`, DNN, ℓ_1-DNN, and the support vector machine (SVM) with different kernels. For SVM, both linear and non-linear kernels such as the radial and polynomial kernels are included. The default parameter setting is used for training the SVMs. The same learning rate, stopping criteria, and mini-batch size (200) are used for `DeepBiome`, DNN and ℓ_1-DNN. Among the SVMs, the linear SVM has the highest accuracy and AUC, while the SVM with radial kernel yields better recall and $F1$ score. However, all SVMs exhibited inferior performance compared to deep learning models: `DeepBiome` exhibits the highest values on all evaluation metrics, with an AUC of 0.9; AUCs of DNN and ℓ_1-DNN are around 0.86. The $F1$ score of `DeepBiome` is 0.711, which is 14% higher than the second best, ℓ_1-DNN. We find that `DeepBiome` offers the best and most balanced performance with precision and recall, which are 0.72 and 0.71, respectively. Since SVM models cannot perform selection on the microbiome taxa, we only compare `DeepBiome` to ℓ_1-DNN in (Figure 5.3, fourth row and Table 5.8). `DeepBiome` surpasses ℓ_1-DNN in all of the evaluation metrics at all phylogenetic levels. For instance, the g-measure of ℓ_1-DNN in selecting genus level taxa is only 0.187 while that of `DeepBiome` is 0.816.

Method	Testing										Training									
	ACC		Precision		Recall		$F1$		AUC		ACC		Precision		Recall		$F1$		AUC	
	mean	sd	mean	sd	mean	sd	mean	sd	mean	sd	mean	sd	mean	sd	mean	sd	mean	sd	mean	sd
SVM-Linear	0.667	0.032	0.525	0.060	0.493	0.027	0.508	0.039	0.667	0.027	0.694	0.017	0.571	0.034	0.520	0.019	0.544	0.024	0.679	0.018
SVM-Radial	0.659	0.033	0.527	0.092	0.506	0.024	0.514	0.050	0.665	0.027	0.821	0.012	0.807	0.035	0.663	0.017	0.728	0.020	0.809	0.015
SVM-Polynomial	0.576	0.034	0.443	0.090	0.394	0.026	0.414	0.048	0.566	0.030	0.767	0.014	0.837	0.012	0.622	0.025	0.714	0.019	0.742	0.026
DNN	0.752	0.045	0.620	0.136	0.599	0.105	0.599	0.115	0.856	0.041	0.854	0.063	0.741	0.188	0.718	0.149	0.718	0.166	0.941	0.044
DNN+ℓ_1	0.760	0.042	0.668	0.139	0.612	0.090	0.618	0.104	0.863	0.043	0.864	0.058	0.794	0.160	0.731	0.130	0.739	0.146	0.948	0.037
DeepBiome	0.815	0.066	0.720	0.151	0.714	0.115	0.711	0.135	0.900	0.080	0.880	0.072	0.804	0.170	0.793	0.135	0.791	0.157	0.942	0.084

Table 5.7: Prediction performance under Scenario 3

Method	PhyloTree	No. true taxa (total)	Sensitivity		Specificity		g-Measure		ACC	
			mean	sd	mean	sd	mean	sd	mean	sd
DNN+ℓ_1	Genus	19 (48)	0.966	0.041	0.036	0.005	0.187	0.012	0.045	0.005
	Family	15 (40)	0.974	0.041	0.028	0.006	0.165	0.017	0.044	0.005
	Order	4 (23)	0.977	0.074	0.024	0.008	0.151	0.026	0.034	0.008
	Class	3 (17)	0.981	0.078	0.019	0.011	0.126	0.049	0.038	0.011
DeepBiome	Genus	19 (48)	0.967	0.041	0.690	0.073	0.816	0.045	0.693	0.072
	Family	15 (40)	0.972	0.043	0.859	0.046	0.913	0.032	0.861	0.046
	Order	4 (23)	0.971	0.082	0.854	0.051	0.910	0.049	0.856	0.050
	Class	3 (17)	0.981	0.078	0.808	0.063	0.889	0.052	0.812	0.062

Table 5.8: Taxa selection performance under Scenario 3

Scenario 4: Robustness under tree mis-specification and measurement errors of microbiome abundance

To examine the robustness of `DeepBiome`, we consider two sources of model mis-specifications,

(1) Abundance data containing measurement errors at genus levels. We assume that 10% of the associated genus reads are mis-classified to one randomly selected genus from the same phylum. The microbiome abundance data with measurement errors is then used for training models.

(2) The phylogenetic tree for training models is mis-specified.

- At the class level, the genera that belong to *Clostridia* and *Flavobacteria* are mis-classified to *Bacilli* and *Bacteroidia*.
- At the order level, the genera that belong to *Coriobacteriales* and *Flavobacteriales* are mis-classified to *Actinomycetales* and *Bacteroidales*.

The same learning rate, stopping criteria, and mini-batch size (100) are used for `DeepBiome`, DNN and ℓ_1-DNN. Tables 5.9, 5.10, and Figure 5.3 present the results with measurement errors. Tables 5.11 and 5.12 show the results when using a mis-specified phylogenetic tree. Like Scenario 1, we compare `DeepBiome` to linear regression, penalized regression, conventional DNN, and ℓ_1-regularized DNN. When the model is trained using data with measurement errors (case 1), performance of `DeepBiome` and ℓ_1-DNN drops, i.e., higher MSE and lower Pearson's ρ, compared to Scenario 1 using data without errors (Table 5.9; see also Table 5.3). `DeepBiome` has the best prediction performance among all methods. The average Pearson's ρ of `DeepBiome` is 0.95, while those of DNN and ℓ_1-DNN are 0.87 and 0.91, respectively. Table 5.11 displays the predic-

Method	Testing				Training			
	MSE		Correlation		MSE		Correlation	
	mean	sd	mean	sd	mean	sd	mean	sd
Linear Regression	1.569	0.154	0.639	0.036	1.336	0.066	0.694	0.018
Ridge	1.551	0.128	0.639	0.036	1.358	0.073	0.694	0.018
Lasso	1.488	0.119	0.661	0.034	1.408	0.073	0.679	0.020
Elastic-net	1.490	0.121	0.660	0.034	1.402	0.075	0.681	0.019
DNN	0.619	0.682	0.873	0.137	0.188	0.317	0.961	0.068
DNN+ℓ_1	0.445	0.351	0.909	0.081	0.129	0.234	0.974	0.050
DeepBiome	0.243	0.400	0.950	0.087	0.117	0.244	0.976	0.052

Table 5.9: Prediction performance under Scenario 4, measurment errors

tive performance under case 2 (mis-specified phylogenetic tree). `DeepBiome` outperforms other methods in both MSE and Pearson's ρ, demonstrating its robustness to tree mis-specifications. Figure 5.3 (5th and 6th row), Tables 5.10 and 5.12 show the ability of identifying associated microbiome taxa. When the abundance data contain measurement errors, the sensitivity decreases in both penalized regression and deep learning methods. For example, the specificity of `DeepBiome` at genus level is 0.67 (compared to 0.95 in Table 5.4), leading to slight decreases of the g-measure from 0.84 to 0.80. `DeepBiome` tends to select less associated taxa when input abundance data contain measurement errors. For the second case, when the phylogenetic tree has taxonomic classification errors, the Lasso, elastic net, and ℓ_1-DNN have similar performances compared to Scenario 1 (Figure 5.3, 6th row). Even if `DeepBiome` used a wrong tree structure to guide the model, it still maintains a decent performance with g-measure of 0.80 at the finest level (Figure 5.3, 6th row).

5.4 Discussion

`DeepBiome`, a phylogenetic tree-regularized deep learning model, is proposed for both prediction and classification tasks. The capability of the developed

Method	PhyloTree	No. true taxa (total)	Sensitivity		Specificity		g-Measure		ACC	
			mean	sd	mean	sd	mean	sd	mean	sd
Lasso	Genus	19 (48)	0.406	0.107	0.817	0.122	0.566	0.061	0.654	0.060
	Family	15 (40)	0.498	0.118	0.798	0.132	0.620	0.063	0.686	0.068
	Order	4 (23)	0.583	0.219	0.751	0.154	0.639	0.120	0.722	0.115
	Class	3 (17)	0.720	0.244	0.696	0.176	0.683	0.129	0.700	0.133
Elastic-Net	Genus	19 (48)	0.444	0.106	0.770	0.117	0.576	0.065	0.641	0.064
	Family	15 (40)	0.539	0.115	0.747	0.127	0.626	0.067	0.669	0.071
	Order	4 (23)	0.649	0.211	0.690	0.148	0.651	0.115	0.683	0.114
	Class	3 (17)	0.790	0.225	0.626	0.171	0.683	0.124	0.655	0.133
DNN+ℓ_1	Genus	19 (48)	0.970	0.040	0.034	0.006	0.180	0.017	0.043	0.006
	Family	15 (40)	0.969	0.044	0.030	0.006	0.170	0.019	0.046	0.006
	Order	4 (23)	0.975	0.079	0.025	0.008	0.152	0.027	0.034	0.008
	Class	3 (17)	0.982	0.075	0.021	0.012	0.135	0.046	0.040	0.012
DeepBiome	Genus	19 (48)	0.949	0.057	0.673	0.081	0.797	0.051	0.675	0.080
	Family	15 (40)	0.968	0.046	0.809	0.063	0.884	0.041	0.812	0.062
	Order	4 (23)	0.971	0.084	0.826	0.069	0.894	0.057	0.828	0.068
	Class	3 (17)	0.979	0.084	0.781	0.085	0.872	0.064	0.785	0.083

Table 5.10: Taxa selection performance under Scenario 4, measurment errors

Method	Testing				Training			
	MSE		Correlation		MSE		Correlation	
	mean	sd	mean	sd	mean	sd	mean	sd
Linear Regression	0.872	0.163	0.683	0.046	0.737	0.074	0.726	0.027
Lasso	0.826	0.144	0.706	0.047	0.780	0.080	0.710	0.030
Ridge	0.866	0.138	0.683	0.046	0.752	0.081	0.726	0.027
Elastic-net	0.826	0.144	0.706	0.047	0.779	0.079	0.711	0.028
DNN	0.437	0.208	0.849	0.077	0.167	0.166	0.944	0.058
DNN+ℓ_1	0.434	0.214	0.850	0.080	0.166	0.171	0.944	0.065
DeepBiome	0.316	0.261	0.892	0.094	0.195	0.207	0.933	0.075

Table 5.11: Prediction performance under Scenario 4, mis-specified phylogenetic tree

`DeepBiome` software is demonstrated with comprehensive simulation experiments. For regression tasks, the results suggest that, compared to sparse regression and other deep learning models, `DeepBiome` has a competitive performance, particularly when outcome-associated microbiome taxa are clustered at different phylogenetic levels. `DeepBiome` also excels in complex classification tasks with higher accuracy and AUC. More importantly, `DeepBiome` enables an intuitive visualization of the microbiome-phenotype association network.

The limitations of `DeepBiome` include the possibility of violation of the following assumptions: (1) microbiome classified in the same cluster have similar effects to outcome of interests, and (2) phylogenetic tree structure translates to effects aggregation structure. Extension of `DeepBiome` to accomodate longitudinal microbiome data is also needed for many studies with repeated measures of microbiome.

Method	PhyloTree	No. true taxa (total)	Sensitivity		Specificity		g-Measure		ACC	
			mean	sd	mean	sd	mean	sd	mean	sd
Lasso	Genus	19 (48)	0.315	0.103	0.863	0.116	0.510	0.064	0.646	0.054
	Family	15 (40)	0.388	0.115	0.847	0.126	0.562	0.067	0.675	0.063
	Order	4 (23)	0.518	0.218	0.810	0.150	0.623	0.124	0.760	0.110
	Class	3 (17)	0.647	0.249	0.765	0.174	0.677	0.132	0.744	0.130
Elastic-Net	Genus	19 (48)	0.333	0.102	0.845	0.113	0.520	0.064	0.642	0.056
	Family	15 (40)	0.409	0.111	0.827	0.122	0.572	0.068	0.670	0.065
	Order	4 (23)	0.559	0.213	0.785	0.142	0.642	0.128	0.745	0.110
	Class	3 (17)	0.691	0.239	0.732	0.164	0.689	0.139	0.725	0.128
DNN+ℓ_1	Genus	19 (48)	0.968	0.042	0.033	0.007	0.179	0.019	0.043	0.007
	Family	15 (40)	0.905	0.043	0.072	0.006	0.256	0.012	0.086	0.006
	Order	4 (23)	0.490	0.051	0.118	0.008	0.239	0.017	0.121	0.007
	Class	3 (17)	0.652	0.071	0.073	0.011	0.216	0.023	0.084	0.011
DeepBiome	Genus	19 (48)	0.952	0.050	0.669	0.077	0.796	0.048	0.672	0.076
	Family	15 (40)	0.905	0.044	0.833	0.063	0.867	0.038	0.834	0.062
	Order	4 (23)	0.486	0.059	0.851	0.058	0.641	0.053	0.847	0.058
	Class	3 (17)	0.655	0.062	0.806	0.074	0.725	0.055	0.803	0.072

Table 5.12: Taxa selection performance under Scenario 4, mis-specified phylogenetic tree

5.5 Appendix

5.5.1 Implementation details for Section 5.3

Sample generation Generation of microbiome abundance data was described in detail by Zhai et al. (2018, 2019). Briefly, to generate an $n \times p$ OTU count matrix, a dirichlet multinomial (DM) distribution was used with the mean proportion vector and the dispersion parameter estimated from a real pulmonary microbiome dataset, which contains $p = 2964$ OTUs and a phylogenetic tree (Twigg III et al., 2016). 2964 OTUs were aggregated as 48 genus. Based on a real lung microbiome dataset, the microbiome data is summarized at genus ($l = 0$), family ($l = 1$), order ($l = 2$), class ($l = 3$), and phylum ($l = 4$) level. The number of nodes are $m^{(0)} = 48$, $m^{(1)} = 40$, $m^{(2)} = 23$, $m^{(3)} = 17$, and $m^{(4)} = 9$, respectively.

In order to generate outcome $\mathbf{y}$, a forward propagation approach described below was used.

1. For level l, construct the weight matrix $\mathbf{w}^l \in \mathbb{R}^{m^{(l)} \times m^{(l-1)}}$ to propagate $\mathbf{x}^{l-1}$ to the lth hidden layer by

$$\mathbf{h}^l = \mathbf{w}^l \mathbf{x}^{l-1} + \mathbf{b}^l.$$

   The bias vector $\mathbf{b}^l \in \mathbb{R}^{m^{(l)}}$ follows a standard normal distribution $\mathcal{N}(0, \sigma_e^2)$ with $\sigma_e^2 = 4$. Suppose we have node j at the level l and k at the level $l-1$, then

$$w_{jk}^l \sim \begin{cases} \text{Uniform}(-0.5, 1) & \text{associated with output} \\ \mathcal{N}(0, 0.01) & \text{not associated with output.} \end{cases} \tag{5.1}$$

2. Multiply the w_{jk}^l by a small value $\omega_{jk}^l = 0.01$, if taxa j at the level l is not a direct ancestor of taxa k at level $l-1$; otherwise, w_{jk}^l stays the same.

3. Activate the neurons using ReLU, $\mathbf{x}^l = ReLU(\mathbf{w}^l\mathbf{x}^{l-1} + \mathbf{b}^l.)$.

4. Repeating step 1 - 3 for $l = 1, ..., 4$.

5. Repeating step 1 - 2 for $l = 5$, and simulate the continuous or categorical output layer $\mathbf{y}$ as follow

$$\begin{aligned} \hat{\mathbf{y}} &= \mathbf{w}^5\mathbf{x}^4 + \mathbf{b}^5 \\ \hat{\Pr}(y_i = c) &= \frac{e^{\mathbf{w}_c^5\mathbf{x}_c^5 + b_c^5}}{\sum_{j=1}^{K}(e^{\mathbf{w}_j^5\mathbf{x}_j^5 + b_j^5})} \end{aligned}$$

   where $K = 2$ for binary classification and $K \geq 3$ for multicategory classification.

Computing infrastructure All simulations are performed using a workstation equipped with Intel(R) Xeon(R) CPU E5-2650 v4 processor with 24 cores @ 2.20GHz and one NVIDIA GeForce GTX TITAN X GPU with 3072 CUDA cores @ 1 GHz and 12GB memory. `DeepBiome` required 290 ± 69 seconds to fully train the network for one replicate with 1000 samples, 50 mini-batches and 5000 epochs. For the same data, DNN took 282 ± 67 second and ℓ_1-DNN took 282 ± 67 second. `DeepBiome` and all other deep learning approaches took less than 0.004 seconds for prediction.

5.5.2 Real-world data analysis

In real-life data analysis, the developed software shows superior prediction performance and selects the taxa associated with the disease, which matched existing clinical literature. Analyses of the American gut project (AGP) and a lung microbiome study using `DeepBiome` software are discussed in Jing Zhai's thesis (Zhai, 2019). In this section, type 2 diabetes prediction with the American Gut Project is described as one example of real-world applications.

Application to the American gut project The American gut project (AGP), launched in 2012, is an open platform for citizen microbiome research (McDonald et al., 2018). Microbiome sample in AGP were collected from stools using dry swabs. For this work, microbiome samples with less than 10,000 sequence reads and genera with abundance less than 2% are excluded. Samples without metadata or missing demographic information, i.e., age, gender, and ethnicity, were also excluded.

Use of the `DeepBiome` software can greatly facilitate model training on enormous datasets like this. In the application of type 2 diabetes prediction, `DeepBiome` model trained with the developed software shows higher prediction performance than other deep learning models, and the selected taxa from `DeepBiome` also relate to the disease as expected from previous clinical studies.

Type 2 diabetes Type 2 diabetes (T2D) is a metabolic disorder with a combination of risk factors such as family history, lifestyle, and genetic factors. In the past decade, multiple studies have indicated that the risk of developing T2D may also involve factors from gut microbiome (Larsen et al., 2010; Musso et al., 2011; Qin et al., 2012). The performance of `DeepBiome` is compared with logistic regression, logistic regression with ridge, Lasso, and elastic net penalization, DNN, and $\ell 1$-DNN. Subjects who reported T2D diagnosed by medical professionals (doctor or physician assistant) were included as T2D cases ($n = 154$). Randomly select 154 subjects without T2D form a control group. This analysis used their demographic information, age, gender, and ethnicity, 373 genus level taxa, and a phylogenetic tree to classify T2D and to also select associated microbiome taxa. Table 5.13 shows the performance of classifying T2D using on 5-fold cross-validation. Although elastic net and Lasso regression have the highest sensitivity, their low specificities suggest that these methods are inclined to

predict healthy subjects as T2D (false positive). DNN and ℓ1-DNN show signs of over-training. `DeepBiome` performs the best among all methods with highest g-measure, accuracy, and AUC, which are 0.653, 0.641, and 0.655, respectively.

Figure 5.4 summarizes the taxa selected by `DeepBiome`, using the 85th percentile of the weight coefficient estimated at each phylogenetic level averaged over 5-fold cross validation. This figure is generated with the visualization tool within the `DeepBiome` software. The blue and red nodes indicate taxa with negative and positive weights, respectively. The size of colored nodes represent the magnitudes of the weights. Black nodes represent non-selected taxa. In total, `DeepBiome` selected 56 genera, 15 families, 8 orders, and 6 classes. Among these taxa, 33 genera, 7 families, 2 orders, and 1 class are positively associated with T2D, indicating that the higher the abundance of these taxa, the higher the probability of subjects having T2D. Compared with the analysis carried out by Qin et al. (2012), `DeepBiome` also selected T2D enriched genus *Alistipes* and *Lachnospira*, and healthy control enriched genus *Haemophilus* and family *Erysipelotrichaceae*. In an analysis of a Denmark T2D cohort, Larsen et al. (2010) pointed out that T2D is associated with gut microbiome dysbiosis, e.g. the proportion of *Clostridia* in diabetics is significantly lower than that in controls, and class *Betaproteobacteria* is highly enriched in subjects with diabetes. However, `DeepBiome` suggests that class *Betaproteobacteria* is negatively associated with T2D and that a group of genera positively associated with T2D are from class *Clostridia*, which also disagrees with Larsen et al. (2010). Indeed, results from Larsen et al. (2010) was not consistent in other cohorts with different ethnicity (Sircana et al., 2018; Qin et al., 2012; Karlsson et al., 2013). These results suggest that the contribution of gut microbiome to T2D may be different across ethnicity groups and environmental factors. Besides T2D, the selected *Haemophilus* from AGP has been demonstrated to be associated with

Method	Testing										Training									
	Sensitivity		Specificity		g-Measure		ACC		AUC		Sensitivity		Specificity		g-Measure		ACC		AUC	
	mean	sd	mean	sd	mean	sd	mean	sd	mean	sd	mean	sd	mean	sd	mean	sd	mean	sd	mean	sd
Logistic	0.499	0.117	0.414	0.074	0.448	0.047	0.464	0.045	0.513	0.065	0.952	0.087	0.937	0.127	0.944	0.107	0.944	0.107	0.945	0.107
Lasso	0.712	0.423	0.386	0.416	0.238	0.326	0.531	0.116	0.549	0.072	0.726	0.420	0.365	0.410	0.264	0.282	0.549	0.052	0.545	0.062
ridge	0.369	0.507	0.669	0.453	0.160	0.219	0.477	0.077	0.519	0.033	0.398	0.546	0.691	0.424	0.190	0.261	0.555	0.062	0.545	0.061
ElasticNet	0.956	0.071	0.067	0.113	0.146	0.213	0.501	0.087	0.511	0.021	0.956	0.022	0.067	0.164	0.202	0.258	0.544	0.060	0.542	0.072
DNN	0.584	0.037	0.501	0.044	0.540	0.036	0.575	0.029	0.604	0.028	0.915	0.013	0.899	0.016	0.907	0.013	0.915	0.013	0.941	0.014
DNN+ℓ_1	0.578	0.052	0.555	0.080	0.566	0.061	0.577	0.050	0.604	0.052	0.920	0.021	0.907	0.028	0.913	0.023	0.919	0.022	0.942	0.020
DeepBiome	0.653	0.057	0.659	0.082	0.653	0.030	0.641	0.167	0.655	0.079	0.643	0.026	0.656	0.057	0.649	0.021	0.629	0.134	0.647	0.067

Table 5.13: Performance of predicting type 2 diabetes

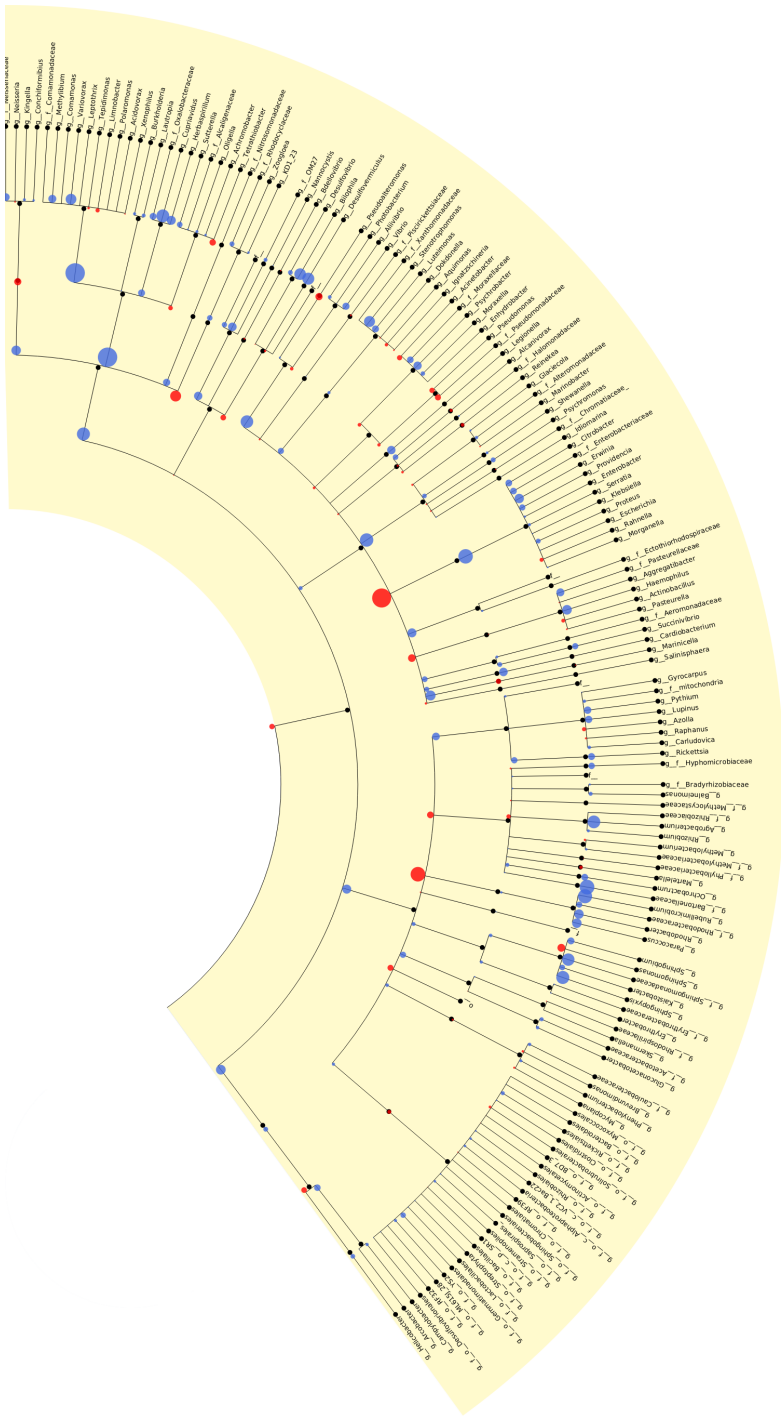


Figure 5.4: T2D associated microbiome taxa from phylum *Proteobacteria* selected by `DeepBiome`

prediabetes (Zhang et al., 2013).

Chapter 6

Conclusion

Correlation structures in modern machine learning tasks deserve more attention. In this dissertation, it is demonstrated via two deep learning models that frameworks considering the structures possessed in data can alleviate robustness, generalizability and explainability of models. In the first part, a representation learning model implementing the nested structure of the data is considered to conduct correlated data generation. In the second part, a DNN-based predictive model leveraging the phylogenetic information is proposed in order to analyze the path from microbiome counts to disease.

The nested structure of the first part originates from data collection, in which grouped observation units exist. The main contribution of this part is to introduce an optimal transport distance between stationary and *correlated* random processes to seek a latent space representation of the observed sequences. By viewing the nested data as a collection of i.i.d. observations of exchangeable random processes, the key attraction of the Ornstein auto-encoders is their ability to handle an unknown number of observational units and to generate samples

within a unit with high quality. In exchangeable sequence generation, OAEs have successfully demonstrated high performance in the three types of tasks commonly advocated in assessing the quality of generative models, namely: exemplar generation, style transfer, and unit generation. Importantly, OAEs are robust to data imbalance and can generate new variations of unknown, out-of-training-set observation units. They have achieved impressive performance in discriminating individuals from the VGGFace2 data and digits from highly imbalanced MNIST data in the representation space.

The second part calls for the need of reflecting hierarchical correlation structures that often arise in biological datasets directly into the model. In the development of the software for `DeepBiome`, a phylogenetic tree-regularized deep learning model that can be used for both prediction and classification tasks. The model regularizes the neural network structure towards the phylogenetic structure inherent in the microbiome data through weight decay. It is shown that using this approach greatly reduces the number of parameters, avoids overfitting, and allows visualization of the pathway from microbiome counts to phenotypes. The capabilities of the developed software have been demonstrated with comprehensive simulation experiments. The `DeepBiome` model trained with this software shows better generalizability than other deep learning models. More importantly, the incorporation of correlation structure enhances the explainablity of the microbiome-phenotype association network with intuitive visualization. In real-life data analysis, my software has shown the ability to train a high-performance model and select disease-related taxa confirmed in the existing clinical literature.

I hope that this work calls attention to the correlation structure of the data commonly observed in many real-world learning tasks.

Bibliography

Abadi, M., Agarwal, A., Barham, P., Brevdo, E., Chen, Z., Citro, C., Corrado, G. S., Davis, A., Dean, J., Devin, M., Ghemawat, S., Goodfellow, I., Harp, A., Irving, G., Isard, M., Jia, Y., Jozefowicz, R., Kaiser, L., Kudlur, M., Levenberg, J., Mané, D., Monga, R., Moore, S., Murray, D., Olah, C., Schuster, M., Shlens, J., Steiner, B., Sutskever, I., Talwar, K., Tucker, P., Vanhoucke, V., Vasudevan, V., Viégas, F., Vinyals, O., Warden, P., Wattenberg, M., Wicke, M., Yu, Y., and Zheng, X. (2015). TensorFlow: Large-scale machine learning on heterogeneous systems. Software available from tensorflow.org.

Arjovsky, M., Chintala, S., and Bottou, L. (2017). Wasserstein generative adversarial networks. In *Proceedings of the 34th International Conference on Machine Learning*, volume 70, pages 214–223.

Bousquet, O., Gelly, S., Tolstikhin, I., Simon-Gabriel, C.-J., and Schoelkopf, B. (2017). From optimal transport to generative modeling: The VEGAN cookbook. *arXiv preprint arXiv:1705.07642*.

Cao, Q., Shen, L., Xie, W., Parkhi, O. M., and Zisserman, A. (2018). VGGFace2: A dataset for recognising faces across pose and age. In *13th IEEE*

International Conference on Automatic Face Gesture Recognition (FG 2018), pages 67–74.

Chen, X., Duan, Y., Houthooft, R., Schulman, J., Sutskever, I., and Abbeel, P. (2016). Infogan: Interpretable representation learning by information maximizing generative adversarial nets. In *Advances in neural information processing systems*, pages 2172–2180.

Choi, Y. and Won, J.-H. (2019). Ornstein auto-encoders. In *Proceedings of the 28th International Joint Conference on Artificial Intelligence*, pages 2172–2178. AAAI Press.

Chollet, F. et al. (2015). Keras. `https://keras.io`.

Diggle, P., Diggle, P. J., Heagerty, P., Heagerty, P. J., Liang, K.-Y., and Zeger, S. (2002). *Analysis of longitudinal data*. Oxford University Press.

Dundar, M., Krishnapuram, B., Bi, J., and Rao, R. B. (2007). Learning classifiers when the training data is not IID. In *Proceedings of the 20th International Joint Conference on Artifical Intelligence*, pages 756–761. Morgan Kaufmann Publishers Inc.

Fitzmaurice, G. M., Laird, N. M., and Ware, J. H. (2012). *Applied longitudinal analysis*, volume 998. John Wiley & Sons.

Franzosa, E. A., Sirota-Madi, A., Avila-Pacheco, J., Fornelos, N., Haiser, H. J., Reinker, S., Vatanen, T., Hall, A. B., Mallick, H., McIver, L. J., et al. (2019). Gut microbiome structure and metabolic activity in inflammatory bowel disease. *Nature Microbiology*, 4(2):293.

Garcia, T. P., Müller, S., Carroll, R. J., and Walzem, R. L. (2013). Identification of important regressor groups, subgroups and individuals via regularization

methods: Application to gut microbiome data. *Bioinformatics*, 30(6):831–837.

Gilbert, J. A., Quinn, R. A., Debelius, J., Xu, Z. Z., Morton, J., Garg, N., Jansson, J. K., Dorrestein, P. C., and Knight, R. (2016). Microbiome-wide association studies link dynamic microbial consortia to disease. *Nature*, 535(7610):94.

Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., and Bengio, Y. (2014). Generative adversarial nets. In *Advances in Neural Information Processing Systems 27*, pages 2672–2680. Curran Associates, Inc.

Gray, R. M., Neuhoff, D. L., and Shields, P. C. (1975). A generalization of Ornstein's $\bar{d}$ distance with applications to information theory. *The Annals of Probability*, pages 315–328.

Gretton, A., Borgwardt, K. M., Rasch, M. J., Schölkopf, B., and Smola, A. (2012). A kernel two-sample test. *Journal of Machine Learning Research*, 13(Mar):723–773.

Gretton, A., Bousquet, O., Smola, A., and Schölkopf, B. (2005). Measuring statistical dependence with hilbert-schmidt norms. In *International conference on algorithmic learning theory*, pages 63–77. Springer.

Gupta, A. and Lam, S. M. (1998). Weight decay backpropagation for noisy data. *Neural Networks*, 11(6):1127–1138.

He, K., Zhang, X., Ren, S., and Sun, J. (2016). Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778.

Heusel, M., Ramsauer, H., Unterthiner, T., Nessler, B., and Hochreiter, S. (2017). GANs trained by a two time-scale update rule converge to a local Nash equilibrium. In *Advances in Neural Information Processing Systems 30*, pages 6626–6637. Curran Associates, Inc.

Hu, J., Shen, L., and Sun, G. (2018). Squeeze-and-excitation networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 7132–7141.

Ioffe, S. and Szegedy, C. (2015). Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *Proceedings of the 32nd International Conference on Machine Learning*, volume 37, pages 448–456. PMLR.

Karlsson, F. H., Tremaroli, V., Nookaew, I., Bergström, G., Behre, C. J., Fagerberg, B., Nielsen, J., and Bäckhed, F. (2013). Gut metagenome in european women with normal, impaired and diabetic glucose control. *Nature*, 498(7452):99.

Kingma, D. and Ba, J. (2014). Adam: A method for stochastic optimization. *International Conference on Learning Representations*.

Krogh, A. and Hertz, J. A. (1992). A simple weight decay can improve generalization. In *Advances in Neural Information Processing Systems*, pages 950–957.

Lake, B. M., Salakhutdinov, R., and Tenenbaum, J. B. (2019). The omniglot challenge: A 3-year progress report. *Current Opinion in Behavioral Sciences*, 29:97–104.

Larsen, N., Vogensen, F. K., Van Den Berg, F. W., Nielsen, D. S., Andreasen,

A. S., Pedersen, B. K., Al-Soud, W. A., Sørensen, S. J., Hansen, L. H., and Jakobsen, M. (2010). Gut microbiota in human adults with type 2 diabetes differs from non-diabetic adults. *PloS One*, 5(2):e9085.

Lopez, R., Regier, J., Jordan, M. I., and Yosef, N. (2018). Information constraints on auto-encoding variational bayes. In *Advances in Neural Information Processing Systems*, pages 6114–6125.

Maaten, L. v. d. and Hinton, G. (2008). Visualizing data using t-sne. *Journal of machine learning research*, 9(Nov):2579–2605.

Makhzani, A., Shlens, J., Jaitly, N., and Goodfellow, I. (2016). Adversarial autoencoders. In *International Conference on Learning Representations*.

McDonald, D., Hyde, E., Debelius, J. W., Morton, J. T., Gonzalez, A., Ackermann, G., Aksenov, A. A., Behsaz, B., Brennan, C., and Chen, Y. (2018). American gut: An open platform for citizen science microbiome research. *MSystems*, 3(3):e00031–18.

Mundie, D. B. and Massengill, L. W. (1991). Weight decay and resolution effects in feedforward artificial neural networks. *IEEE Transactions on Neural Networks*, 2(1):168–170.

Musso, G., Gambino, R., and Cassader, M. (2011). Interactions between gut microbiota and host metabolism predisposing to obesity and diabetes. *Annual Review of Medicine*, 62:361–380.

Ni, J., Shen, T.-C. D., Chen, E. Z., Bittinger, K., Bailey, A., Roggiani, M., Sirota-Madi, A., Friedman, E. S., Chau, L., Lin, A., et al. (2017). A role for bacterial urease in gut dysbiosis and crohn's disease. *Science translational medicine*, 9(416).

Odena, A., Olah, C., and Shlens, J. (2017). Conditional image synthesis with auxiliary classifier GANs. In *Proceedings of the 34th International Conference on Machine Learning - Volume 70*, pages 2642–2651. JMLR.org.

Olshen, R. (1974). A note on exchangeable sequences. *Zeitschrift für Wahrscheinlichkeitstheorie und Verwandte Gebiete*, 28(4):317–321.

Ornstein, D. S. (1973). An application of ergodic theory to probability theory. *The Annals of Probability*, 1(1):43–58.

Parkhi, O. M., Vedaldi, A., and Zisserman, A. (2015). Deep face recognition. In Xie, X., Jones, M. W., and Tam, G. K. L., editors, *Proceedings of the British Machine Vision Conference 2015, BMVC 2015, Swansea, UK, September 7-10, 2015*, pages 41.1–41.12. BMVA Press.

Qin, J., Li, Y., Cai, Z., Li, S., Zhu, J., Zhang, F., Liang, S., Zhang, W., Guan, Y., and Shen, D. (2012). A metagenome-wide association study of gut microbiota in type 2 diabetes. *Nature*, 490(7418):55.

Reiman, D., Metwally, A., and Dai, Y. (2017). Using convolutional neural networks to explore the microbiome. In *Engineering in Medicine and Biology Society (EMBC), 2017 39th Annual International Conference of the IEEE*, pages 4269–4272. IEEE.

Rubenstein, P. K., Schölkopf, B., and Tolstikhin, I. (2018a). Learning disentangled representations with Wasserstein auto-encoders. In *Workshop at the 6th International Conference on Learning Representations (ICLR)*.

Rubenstein, P. K., Schölkopf, B., and Tolstikhin, I. (2018b). Wasserstein auto-encoders: Latent dimensionality and random encoders. In *Workshop at the 6th International Conference on Learning Representations (ICLR)*.

Salimans, T., Goodfellow, I., Zaremba, W., Cheung, V., Radford, A., and Chen, X. (2016). Improved techniques for training gans. In *Advances in neural information processing systems*, pages 2234–2242.

Sartor, R. B. (2008). Microbial influences in inflammatory bowel diseases. *Gastroenterology*, 134(2):577–594.

Schloss, P. D. and Handelsman, J. (2005). Introducing dotur, a computer program for defining operational taxonomic units and estimating species richness. *Appl. Environ. Microbiol.*, 71(3):1501–1506.

Sircana, A., Framarin, L., Leone, N., Berrutti, M., Castellino, F., Parente, R., De Michieli, F., Paschetta, E., and Musso, G. (2018). Altered gut microbiota in type 2 diabetes: Just a coincidence? *Current Diabetes Reports*, 18(10):98.

Tolstikhin, I., Bousquet, O., Gelly, S., and Schoelkopf, B. (2018). Wasserstein auto-encoders. In *International Conference on Learning Representations*.

Turnbaugh, P. J., Ley, R. E., Hamady, M., Fraser-Liggett, C. M., Knight, R., and Gordon, J. I. (2007). The human microbiome project. *Nature*, 449(7164):804.

Twigg III, H. L., Knox, K. S., Zhou, J., Crothers, K. A., Nelson, D. E., Toh, E., Day, R. B., Lin, H., Gao, X., Dong, Q., et al. (2016). Effect of advanced HIV infection on the respiratory microbiome. *American Journal of Respiratory and Critical Care Medicine*, 194(2):226–235.

Villani, C. (2008). *Optimal Transport: Old and new*, volume 338. Springer Science & Business Media.

Wang, Z., Bovik, A. C., Sheikh, H. R., and Simoncelli, E. P. (2004). Image

quality assessment: From error visibility to structural similarity. *IEEE Transactions on Image Processing*, 13(4):600–612.

Xiao, J., Chen, L., Johnson, S., Zhang, X., and Chen, J. C. (2018). Predictive modeling of microbiome data using a phylogeny-regularized generalized linear mixed model. *Frontiers in Microbiology*, 9:1391.

Zhai, J. (2019). Advances in microbiome analysis: From the variance component model to deep learning.

Zhai, J., Kim, J., Knox, K. S., Twigg III, H. L., Zhou, H., and Zhou, J. J. (2018). Variance component selection with applications to microbiome taxonomic data. *Frontiers in Microbiology*, 9:509.

Zhai, J., Knox, K., Twigg III, H. L., Zhou, H., and Zhou, J. J. (2019). Exact variance component tests for longitudinal microbiome studies. *Genetic Epidemiology*, 43(3):250–262.

Zhang, G., Wang, C., Xu, B., and Grosse, R. (2019). Three mechanisms of weight decay regularization. In *International Conference on Learning Representations*.

Zhang, X., Shen, D., Fang, Z., Jie, Z., Qiu, X., Zhang, C., Chen, Y., and Ji, L. (2013). Human gut microbiota changes reveal the progression of glucose intolerance. *PloS One*, 8(8):e71108.

초록

자료의 구조를 알고 있는 경우, 이 구조를 활용한 프레임워크는 심층 학습에서 마주하는 강건성, 일반화 및 설명가능성 등의 중요한 이슈를 해결하는 데 도움을 줄 수 있다. 본 학위 논문에서는 구조적 자료를 활용한 심층 학습 방법을 두 가지 문제에 대해 다룬다. 첫 번째 문제는 중첩 구조 자료를 생성할 수 있는 표현 학습 모형의 개발이다. 본 연구에서는 상관성이 있는 자료를 위한 표현 학습 모형 Ornstein auto-encoder (OAE)를 제안한다. 많은 실제 자료는 그룹화된 측정에서 얻어지므로 중첩 구조를 가진다. 예를 들어, VGGFace2 자료는 한 사람당 평균 300개의 이미지로 구성된 자료이다. 이러한 자료는 정상 확률 과정의 i.i.d. 샘플로 구성된 것으로 볼 수 있다. 이를 통해, 두 정상 확률 과정 사이의 최적 수송 거리 (optimal transport distance, Orstein's d-bar distance)를 이용하는 OAE 방법을 제안한다. OAE 방법은 훈련에 사용되지 않은 관측 유닛에 대해서도 해당 유닛의 새로운 이미지를 생성할 수 있다는 점에서 기존의 조건부 모형과 구별되는 고유한 특징을 가진다. 이는 자료의 구조를 활용한 프레임 워크로 심층 신경망 모형의 일반화 성능을 향상시킬 수 있음을 보여준다. 또한, 자료가 교환 가능한 수열 (exchangeable sequence)인 경우, OAE는 훈련 가능한 알고리즘을 제공한다. OAE 방법은 생성 모형의 성능을 나타내는 전형 생성(exemplar generation), 스타일 이전 (style transfer), 관측 유닛 생성 (unit generation) 문제에서 모두 높은 성능을 보여준다. 또한 불균형 자료에 대해서도 소수 집단에 속하는 유닛의 이미지 생성에 기존의 조건부 방법보다 강건한 결과를 보여준다.

본 학위 논문은 또한 미생물의 분류별 개수 자료 (microbiome taxonomic abundance data)의 계층적인 상관 구조를 포착할 수 있는 예측 모형 개발에 대한 내용을 담고 있다. 미생물 개수 자료는 많은 질병을 예측할 수 있는 지표이지만, 계통 발생학 관점에서 계층적인 상관 구조를 가지고 있어 이를 반영한 분석이 필요하다.

DeepBiome은 심층 학습 기반의 예측 모형으로 계통 발생 정보를 활용해 심층 신경망의 과적합을 막고, 질병과 미생물 개수 자료 간의 관계를 설명한다. 훈련된 모형은 일반화 및 설명 가능성 면에서 기존 심층 신경망보다 좋은 성능을 보여준다. 본 논문은 이 연구에서 DeepBiome 소프트웨어 개발에 대한 내용을 담고 있다. 개발한 소프트웨어의 성능은 시뮬레이션 실험을 통해 확인한다. 회귀 문제와 분류 문제에서, 예측 성능 및 질병과 관련된 미생물 분류 선택 모두 기존의 희소 회귀 방법과 심층 학습 방법보다 DeepBiome이 우수한 성능을 보이는 것을 확인할 수 있다. 또한 DeepBiome 소프트웨어는 질병과 미생물 개수 자료 간의 관계에 대해 설명 가능한 심층 신경망의 시각화 자료를 제공한다.

주요어: 심층 학습, 표현 학습, 중첩 구조 자료, 얼굴 이미지 생성, 조건부 생성 모형, 미생물군유전체, 계통발생학

학번: 2016-30094