Variational Autoencoders for Discovering Influential Latent Factors

by

William Hu

B.S. Computer Science and Engineering
Massachusetts Institute of Technology (2022)

Submitted to the Department of Electrical Engineering and Computer Science
in partial fulfillment of the requirements for the degree of

Master of Engineering in Electrical Engineering and Computer Science

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

September 2022

Author .....
Department of Electrical Engineering and Computer Science
August 5, 2022

Certified by .....
Onkar Bhardwaj
AI Engineer, MIT-IBM Watson AI Lab
Thesis Supervisor

Certified by .....
Audé Oliva
Senior Research Scientist
Thesis Supervisor

Accepted by .....
Katrina LaCurts
Chair, Master of Engineering Thesis Committee

Variational Autoencoders for Discovering Influential Latent Factors

by

William Hu

Submitted to the Department of Electrical Engineering and Computer Science
on August 5, 2022, in partial fulfillment of the
requirements for the degree of
Master of Engineering in Electrical Engineering and Computer Science

Abstract

Generative modeling is increasingly being used to simulate or generate new unseen data instances by means of modeling the statistical distribution of data. Generative modeling falls under the broad area of representation learning, which aims to discover representations required for detecting features, classification, and other ways of understanding data. In this vein, variational autoencoders (VAEs) and their variants are one technique of generative modeling (and therefore representation learning) using variational inference under the assumption that the underlying data distribution is composed of a few latent random variables. For example, a VAE (or some other generative learning model) might learn that an image of a person can be generated from the hair color, face shape, and background color. By decomposing the data into latent factors, we could generate and explore new unseen data, which would enable us to investigate how certain data looks like in different environments. However, VAEs are not perfect, and the trained latent factors trained could potentially contain redundant information. In this thesis, we propose to apply VAEs as an unsupervised technique (i.e., in the absence of any external metadata) to investigate the extent to which we can discover a disentangled representation of tabular data and use these factors to generate new data.

Thesis Supervisor: Onkar Bhardwaj
Title: AI Engineer, MIT-IBM Watson AI Lab

Thesis Supervisor: Audé Oliva
Title: Senior Research Scientist

Acknowledgments

First, my deepest gratitude goes to Onkar Bhardwaj, my mentor at MIT IBM Watson AI Lab. Onkar has consistently provided me with constructive feedback, and his constant encouragement has really helped me throughout times when I felt lost. Onkar, thank you so much for your continued motivation, advice, and confidence that you have shown me throughout my internship at MIT IBM Watson AI Lab. I wish you the absolute best in your future endeavors and at IBM.

I would also like to thank Lee Martie for his help in getting me set up for my internship at MIT IBM Watson AI Lab and for organizing our weekly Demo Days. You were absolutely instrumental in introducing me to MIT IBM Watson AI Lab, and your guidance in the beginning was crucial in enabling me to set sail on my project. The Demo Days were a great way for me (and others!) to demonstrate what I had worked on, and they greatly motivated me to constantly work hard. In addition, I extend my heartiest congratulations to you on the birth of your new son. I am incredibly excited for you, and I wish you all the best as you navigate this chapter of your life as a new father.

I would like to thank Akash Srivastava for his guidance and knowledge, especially throughout our weekly meetings. As one of the authors of the β-TCVAE paper [3], you have consistently guided Onkar and me, among many others, to help us understand the technical details of concepts that we read about and helped clear up misunderstandings or gaps in knowledge that we might have.

I would also like to thank Kristjan Greenewald for his suggestions throughout our weekly meetings. Throughout our discussions, you provided different perspectives and suggestions pointed us in directions that we might not have thought of otherwise.

I have been very fortunate to have a supportive group of friends throughout my process. I would especially like to thank Spencer Compton, who also participated in the MIT 6-A program through MIT IBM Watson AI Lab, for his help when I had any questions about his own experience in the program.

Finally, I would like to thank my parents for their help, support, and sacrifice; I would

not be here if it were not for them.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

The raw data used in machine learning tasks in many companies is often unstructured and unlabeled, making them unusable in raw form. Acquiring labels for the data can often be a slow, time-consuming task that takes up significant resources. Therefore, it is often helpful for companies to be able to extract information such as features from unlabeled data so they can perform tasks such as prediction or classification. However, it is often hard to know what information should be extracted.

The rest of the chapter is organized as follows: we begin by introducing a broader categorization of machine learning models as discriminative and generative models. Then we briefly describe how generative models can be used to discover useful features to model the data. Later on, we discuss autoencoders and in particular variational autoencoders, a widely used variant of generative models which is of special interest for this thesis.

1.1 Discriminative vs Generative Modeling

Among the many ways to categorize different machine learning models, one such categorization is between discriminative and generative models. Discriminative models are typically supervised learning techniques which learn to distinguish between different classes of data by modeling the conditional distribution of class labels given data features. While discrim-

inative models are the most popular variety of machine learning models, they are limited because they may not understand the true distribution of the data, as they may need to extract only a few high-level features from the data to distinguish between the classes. For example, for a discriminative model learning to distinguish between horses and zebras, learning to recognize stripes might suffice. Generative models, on the other hand, aim to generate new unseen instances of the data by trying to understand the underlying distribution of the data. There exist several different kinds of generative modeling techniques, such as Bayesian networks, Generative Adversarial Networks (GANs) and variational autoencoders (VAEs). We now give a quick tour of some work in generative modeling before giving an overview of VAEs.

The field of generative modeling attempts to model the true distribution of the input data. Popular techniques of generative modeling include Bayesian networks, generative adversarial networks (GANs), and variational auto-encoders (VAEs). Bayesian networks [6] attempt to model the distribution of the data as a probabilistic graphical model with the nodes of a Directed Acyclic Graph (DAG) representing different features of the data and the edges encoding the conditional independence stuctures between the features. The overall distribution of the data is then computed as the product of various conditional distributions in the DAG. While Bayesian Networks often operate on explicit features of the data, techniques such as GANs and VAEs implicitly learn the distribution of the raw data.

To achieve this, GANs [5] exploit adversarial techniques where a generator module generates data instances by randomly sampling from a low-dimensional latent space, followed by a discriminator module that evaluates whether or not they are likely to belong in the training dataset. The aim of the generator is to learn to generate data instances which could pass the evaluation by the discriminator.

Among these, VAEs and their variants are of special interest to us. The fundamental assumption behind VAEs is that the data can be modeled as a nonlinear transformation of a multivariate distribution of a few Gaussian-distributed random variables, typically with much lower dimensionality than the number of features in the data. To achieve the desired

properties in the latent space, VAEs use variational inference techniques from statistical inference towards defining the loss function of the optimization process.

VAEs make use of the multivariate distribution of latent factors discovered in the process of optimization to generate new instances of data. There have been recent developments in research utilizing external metadata to guide the latent space to discover new *causal* interventions. We investigate the extent to which we can generate disentangled representations of different types of data by applying VAEs and related techniques, mainly in the financial domain. These representations can then potentially be used to simulate new unseen data regimes and to study system behavior under these regimes.

1.2 Autoencoders

An autoencoder is a neural network which learns efficient encodings and decodings of unlabeled data. For input x in input space X, the autoencoder attempts to learn an encoding $e : X \to Z$ and a decoding $d : Z \to X$, where Z is the latent space. It attempts to minimize the reconstruction loss $L(x, \hat{x})$, where $\hat{x} = d(e(x))$ is the reconstructed input. The latent vector z serves as a machine-understandable representation.

1.2.1 Variational Autoencoders

While GANs do not impose any structure on the latent space, VAEs [8] assume that the latent variables are multivariate Gaussian distributed. VAEs map the given input data to the latent space via an encoder module, which is then mapped to the input data via a decoder module. VAEs optimize an objective that combines the deviation of the latent variable distribution from the target multivariate Gaussian distribution with the quality of reconstructed data after passing input data through encoder and decoder. Generating a new data instance then corresponds to sampling from the latent space and passing it through the decoder module.

While the encoding $e : X \to Z$ and decoding $d : Z \to X$ are deterministic in traditional

autoencoders, variational autoencoders are nondeterministic. In fact, the statistical motivation behind variational autoencoders (VAEs) is different. In variational autoencoders, we assume that there is a hidden variable $z \sim p(z) = \mathcal{N}(0, I)$ that generates an observation x from a distribution $p(x|z)$. Given an observation x, we would like to infer the hidden value z based on x, i.e. compute $p(z|x)$. Bayes's rule gives us

$$p(z|x) = \frac{p(x|z)p(z)}{p(x)}$$

However,

$$p(x) = \int_z p(x|z)p(z)dz$$

is often intractable. Therefore, we use variational inference: we can approximate $p(z|x)$ with another distribution $q(z|x)$ (which is usually a Gaussian distribution) and, at the same time, attempt to minimize $\mathrm{KL}\left(q(z|x) \parallel p(z|x)\right)$.

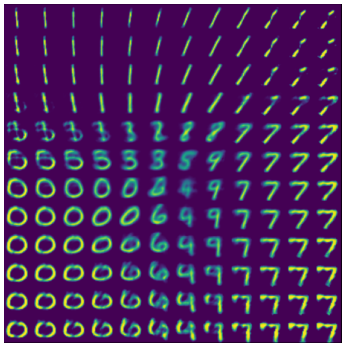
A VAE minimizes this by maximizing the ELBO (evidence lower bound) quantity

$$\mathcal{L}_{VAE} = \mathbb{E}_{q(z|x)}[\log p(x|z)] - \mathrm{KL}\left(q(z|x) \parallel p(z)\right)$$

The ELBO quantity is named as such because it is a lower bound of $\log p(x|z)$, i.e.

$$\log p(x|z) \geq \mathcal{L}_{VAE}$$

The effect of adding the KL-divergence term (which is also known as the *auxiliary loss term*) is significant. We analyze this by comparing VAEs to traditional autoencoders. In traditional autoencoders, the conditions for the latent space (i.e. Z-representation) are simple: values in the latent space should be easily decodable back into the original input. This, however, may cause the latent space Z to be disjoint. VAEs, however, address this issue by adding the auxiliary loss term by attempting to make the distribution $p(z|x)$ as close as possible to the same distribution, no matter the value of the input x. As a result, the auxiliary loss acts as a regularizing force and encourages the distribution to find a proper



(a) VAE-generated latent space

(b) GAN-generated latent space

Figure 1-1: VAE and GAN generated 2D latent spaces for MNIST dataset

and describable encoding for the input, which makes the latent space encodings a compact region for every single input type.

This property of VAEs is best illustrated in Figure 1-1, which includes the generated encodings for the digits using the MNIST dataset [4]. Compared to the GAN-generated latent space, the VAE-generated latent space has a single contiguous subregion for each individual digit, as opposed to the haphazard distribution of digits with a GAN-generated latent space.

1.3 Our Approach

In [2] and [11], the authors use external metadata (such as known drugs) to interpret or guide the structure of the latent space of generative models. In comparison to this, we intend to use VAEs and related generative models as unsupervised learning techniques in the absence of any external metadata to investigate the extent to which the latent space can reveal influential latent factors. These factors can then be used to simulate different data regimes by changing the values of associated latent variables.

The MIT IBM Watson AI Lab has been specifically investigating the relationship between associating disentangled representations with causal factors, and it is conjectured that a fully unsupervised learning approach has its limitations; it might not be able to find all causal factors. Our project attempts to understand a baseline and quantify those limitations for

tabular data.

We propose to use MSVAE, which we introduce in Section 2.4, to understand the extent to which unsupervised learning could be applied to discover influential latent factors in the tabular data and generate unseen data environments. For our experiments, we use financial data obtained from [18] to describe our findings. Specifically, we propose to do the following:

1. Design and implement MSVAE using PyTorch for an image dataset and a tabular dataset (Chapter 3).

2. Generate unseen data using the MSVAE, and for different types of data, see how varying the parameters of the MSVAE changes the data generated (Chapter 4).

3. Implement and evaluate metrics to understand the quality of representations learned by the MSVAE for a tabular dataset. Using the metrics, understand the extent to which we were able to learn a disentangled representation of tabular data (Chapter 5).

Chapter 2

Background

We discuss existing work that attempts to learn representations of different types of data using autoencoders. According to the history in [15], some of the early uses of autoencoders for representation learning were for unsupervised pre-training for neural networks. However, as described in the following sections, we use representation learning for different purposes. There are several variants of autoencoders, such as the compositional perturbation autoencoder (CPA) [11] and variational autoencoders (VAEs) [8]. Below we discuss some recent work to use autoencoders and VAEs for discovering influential latent factors under different domains.

2.1 A Computational Method for Repurposing Drugs for COVID-19

In [2], the authors propose a computational platform for repurposing drugs for SARS-CoV-2 by applying generative modeling (autoencoders) in conjunction with causal networks. In this research, the authors employ an overparameterized autoencoder to discover potential interventions as the drugs that correspond to a movement in the latent space opposite to the movement caused by the SARS-CoV-2 in the affected cell-types. [2] makes use of the Connectivity Map (CMap) database [17] from the Broad Institute, which contains gene

expressions of human cell types that have been given doses of certain drugs. [2] embeds the CMap data together with the SARS-CoV-2 expression data for signature matching and trains an autoencoder that minimizes reconstruction error on CMap data. These potential drugs are then passed through a causal framework for further validation and refinement.

The effect of a drug is often specific to a cell type. Therefore, a standard approach of for computing drug signatures may not be sufficient. To determine robust drug signatures, the authors embed the CMap data together with the SARS-CoV-2 expression data for signature matching, and they train an autoencoder that minimizes reconstruction error on CMap data. This establishes drug signatures in the vein of the CMap database.

Mathematically speaking, suppose that cell types C_1 and C_2 both have an entry for perturbation (drug) D. Then, the approach will work if the effect of the drug D is aligned in both C_1 and C_2 (i.e. within the latent space, $C_1 - C_1' = C_2 - C_2'$), where C_1' and C_2' denote the autoencoder's embeddings of cells C_1 and C_2 after the perturbation of drug D.

2.2 Compositional Perturbation Autoencoder (CPA)

The compositional perturbation autoencoder (CPA) [11] is an interpretable method that analyzes and predicts perturbation responses given a combination of conditions. Most importantly, the latent space of neural networks is not linear; otherwise, the observed gene expression could simply be factored into distinct perturbation components. On the other hand, the CPA models a nonlinear superposition latent space.

Given a dataset of different cell type, drug, and dosage combinations and the results of such perturbations, the CPA model produces interpretable embeddings for those cell types and predicts a cell's gene expression for unseen drug-dosage combinations. The CPA decomposes a cell's gene expression into three embeddings: the basal state, perturbation (e.g. dose and time), and covariates (e.g. cell type).

The CPA disentangles the basal state from the perturbation and covariate embeddings in a GAN-like manner — the CPA's encoder acts like a generator against a corresponding discriminator classifier. The encoder attempts to disentangle the basal state, after which the

discriminator cannot predict the perturbation or covariate values. The three types of learned embeddings are then integrated into a single embedding, and a neural network decoder recovers the gene expression vector.

The most useful aspect of the CPA comes at evaluation time: after the disentangled embeddings are calculated, one can simply substitute in a different perturbation embedding to determine a counterfactual, namely the gene expression of a cell if it had been given a different treatment.

[11] attempts to demonstrate three different scenarios of the application of the CPA: (i) diverse doses, (ii) drug combinations, and (iii) variation of time instead of dose.

2.3 β-VAE and β-TCVAE

As described in Section 1.2, a VAE randomly generates an encoding z of an input x from an input space X from a (usually Gaussian) distribution $q(z|x)$. The objective it attempts to maximize is $\mathbb{E}_{q(z|x)}[\log p(x|z)] - \mathrm{KL}\left(q(z|x) \parallel p(z)\right)$.

The first term is the expected reconstruction loss with posterior $q(z|x)$, and the second (KL divergence) term attempts to match the posterior $q(z|x)$ as close to the distribution $p(z) = \mathrm{N}(0, \mathbf{I})$. Because the dimensions of the distribution $p(z)$ are independent from each other, the KL-divergence term also attempts to move the distribution $q(z|x)$ towards a disentangled representation. From this, we derive the β-VAE framework and then the β-TCVAE framework.

2.3.1 β-VAE

The β-VAE framework is a modification of the VAE objective:

$$\mathcal{L}_{\beta-VAE} = \mathbb{E}_{q(z|x)}[\log p(x|z)] - \beta\ \mathrm{KL}\left(q(z|x) \parallel p(z)\right)$$

where $\beta > 1$, which results in an even more disentangled representation z than a normal VAE. However, higher values of β often result in a tradeoff between the reconstruction quality

and learning a disentangled representation.

This idea is best illustrated with an example, which we do in the scatterplot Figure 2-1. In the setup, we train different β-VAEs to encode the MNIST dataset [4] into a two-dimensional latent space for different values of β.

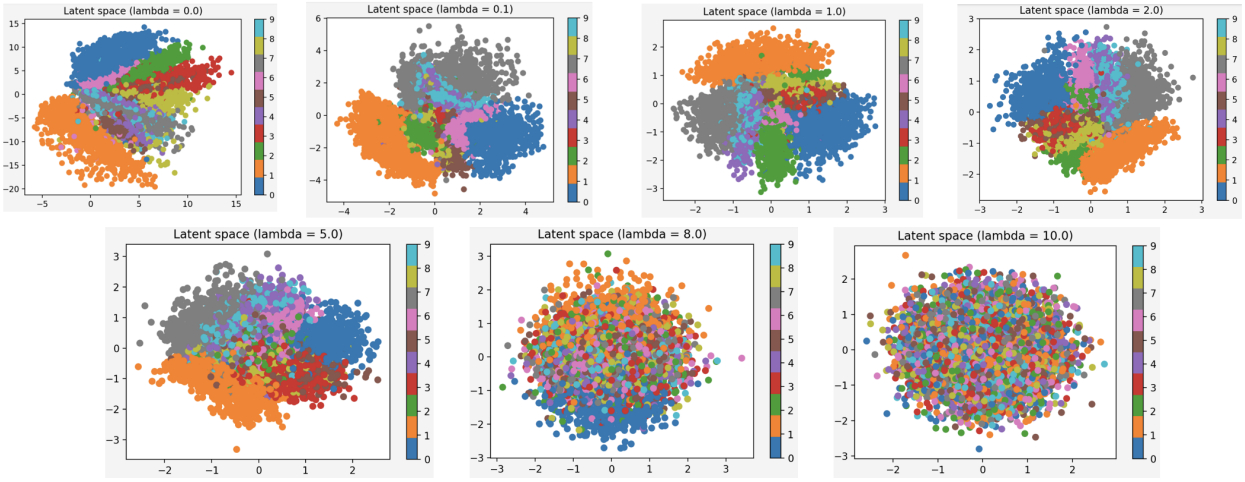


Figure 2-1: Samples from 2D latent spaces for MNIST Dataset (Note that the diagrams are titled with "lambda", but it is equivalent to β)

In Figure 2-1, for the lowest values of β, the latent space sorts each individual digit out into distinct contiguous groups, each with their own centers. The highest values of β result in a more randomly scattered plot for each individual digit; however, the points collectively form a circular shape around the origin that is more compact than the plots for the lowest values of β.

2.3.2 β-TCVAE

The β-TCVAE [3] further explores the term $\mathrm{KL}\left(q(z|x) \parallel p(z)\right)$ and looks for sources of disentanglement within it, leading to an even newer β-TCVAE objective.

Given a dataset $\{x_i\}_{i=1}^{N}$ of data, the β-VAE objective for a single data point x_i is

$$\mathcal{L}_{\beta-VAE} = \mathbb{E}_{q(z|x_i)}[\log p(x_i|z)] - \beta \ \mathrm{KL}\left(q(z|x_i) \parallel p(z)\right)$$

For the entire dataset $\{x_i\}_{i=1}^N$, we define the β-VAE objective as

$$\mathcal{L}_{\beta-VAE} = \frac{1}{N}\sum_{i=1}^{N}\left(\mathbb{E}_{q(z|x_i)}[\log p(x_i|z)] - \beta\ \mathrm{KL}\left(q(z|x_i)\parallel p(z)\right)\right)$$

The authors of [3] then introduce a new setup by adding a new random variable on $\{1, 2, \ldots, N\}$ with distribution $p(n) = \frac{1}{N}$ (i.e. $p(n)$ is uniform over the indices). As stated in [3], letting $q(z, n) = q(z|x_n)p(n)$, $q(z) = \sum_{i=1}^{N} q(z, i)$, and letting L be the dimension of the latent variable z, this enables us to decompose the average (or "expected value", in another sense) of the KL-divergence term $\mathrm{KL}\left(q(z|x_i)\parallel p(z)\right)$:

$$\mathbb{E}_{p(n)}\left[\mathrm{KL}\left(q(z|x_n)\parallel p(z)\right)\right] = \mathrm{KL}\left(q(z,n)\parallel q(z)p(n)\right) + \underbrace{\mathrm{KL}\left(q(z)\parallel\prod_{i=1}^{L}q(z_i)\right)}_{\text{Total Correlation (TC)}} + \sum_{i=1}^{L}\mathrm{KL}\left(q(z_i)\parallel p(z_i)\right)$$

The Total Correlation (TC) term is of particular importance to our new VAE framework. Being a multidimensional analog of mutual information, it pushes our model to find statistically independent latent factors. Based off this idea, the β-TCVAE framework weights the TC term with a weight $\beta > 1$, which inclines it towards a more disentangled representation. As the authors claim, it is the TC term in the decomposition that makes β-TCVAE work. Therefore, our β-TCVAE loss function (for an entire dataset) becomes

$$\begin{aligned}\mathcal{L}_{\beta-TCVAE} = &\left(\frac{1}{N}\sum_{i=1}^{N}\left(\mathbb{E}_{q(z|x_i)}[\log p(x_i|z)]\right)\right)\\ &-\left(\mathrm{KL}\left(q(z,n)\parallel q(z)p(n)\right) + \beta\ \mathrm{KL}\left(q(z)\parallel\prod_{i=1}^{L}q(z_i)\right) + \sum_{i=1}^{L}\mathrm{KL}\left(q(z_i)\parallel p(z_i)\right)\right)\end{aligned}$$

Nevertheless, as with β-VAE, the β-TCVAE framework still results in a tradeoff between reconstruction quality and learning a disentangled representation.

2.4 MSVAE

There have been many variations of VAEs in the literature. (See [8] for a (incomplete) list of some examples.) A growing line of research is to use VAEs to learn a disentangled representation of data by modeling the data as a result of independent latent factors (see [3] for some examples). One particular variant of interest to us is the Multi-Stage Variational Autoencoder (MSVAE), introduced in [16]. MSVAE assumes that the data is composed of disentangled factors (i.e., independent latent random variables) and correlated factors that can be learned after separating the contribution of disentangled factors. Thus, MSVAE improves data reconstruction by resolving the tradeoff between reconstruction quality and the disentangled representation (DR).

Disentangled representation is a loosely-defined term that has no formal definition but is generally understood as implying that manipulating only one semantic factor will cause only one semantically meaningful aspect of an input to change. More formally, DR assumes that the latent variables h in an autoencoder can be partitioned into independent components c and correlated components z, i.e. DR assumes that an observation x is generated from low-dimensional factors $h = (c, z)$.

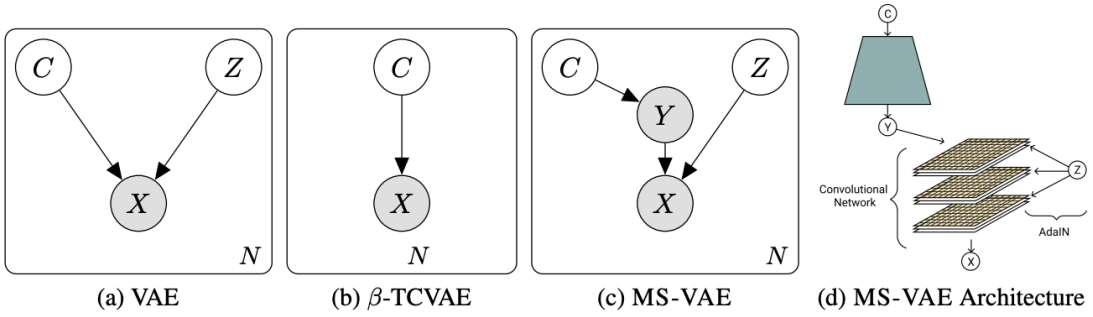


Figure 2-2: The MSVAE architecture, as described in [16]

The MSVAE [16] first learns a disentangled latent representation C with a β-TCVAE [3], an existing DR learning method. Then, the MSVAE attempts to learn the correlated components Z to avoid the tradeoff between reconstruction quality and DR learning. It trains a DSVAE (D-separated VAE), an architecture similar to the DCGAN architecture

[14], on the disentangled β-TCVAE output y and a VAE for the difference $x - y$ between the input to learn a latent representation z of the correlated factors missing from the β-TCVAE, which enables us to learn a final output $\hat{x}$.

Chapter 3

MSVAE: Design and Implementation

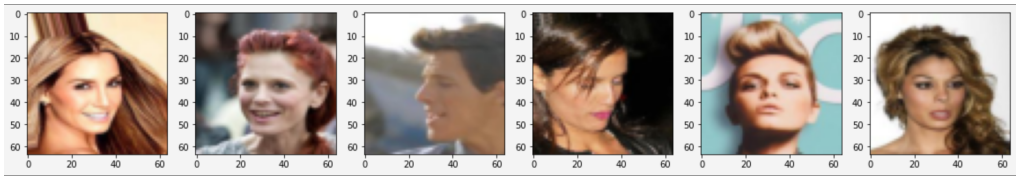
We design and implement a Python library for the MSVAE (Figure 2-2), which we will ultimately use to generate and explore unseen data in Chapter 4. We first run our library on image data and then extend it to non-image data. By doing so, we extend the MSVAE beyond the image settings as described in [16] into a new realm of tabular data. However, due to the single-dimensional nature of tabular data, we accordingly adjust the architecture for its corresponding MSVAE. Our code is based on [1], and some of it can be found in Appendix C.

3.1 β-TCVAE: celebA

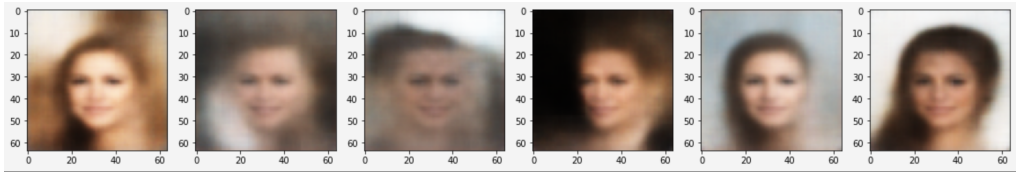
The first step of the MSVAE is a β-TCVAE (β-Total Correlation Variational Autoencoder), as shown in Figure 2-2. As mentioned in Section 2.4, the β-TCVAE learns the disentangled latent representation c, which is generated from a latent space with a smaller dimension than the original data. Figure 3-1 visualizes sample images from the celebA dataset [9], a dataset with color images of celebrity faces, and their reconstructions from a β-TCVAE. (More reconstructions are in Appendix A in Figure A-1.) The reconstructed images reflect the small number of latent dimensions, as they strip down each image into a few essential features. Most notably, all the images in the reconstruction have the same template face,

which indicates that the presence of a face itself is a latent factor. Another characteristic of this concept is that in each reconstruction, the background is compressed down to only a few colors.

The β-TCVAE architecture is based on [1], with alternating 2D convolutional layers and ReLU layers in both the encoder and the decoder. The encoder finds the μ and the $\log(\sigma^2)$ parameters of the corresponding Gaussian distribution, from which a sample image is generated and put through the decoder.



(a) 5 original images from celebA dataset



(b) β-TCVAE reconstructions

Figure 3-1: Sample celebA dataset [9] images and their reconstructions

3.1.1 Comparison with β-VAE

We show that our β-TCVAE approach is better at finding disentangled factors than the β-VAE.

The β-VAE, unlike the β-TCVAE, is not targeted towards disentangled representations. This lack of targeting is reflected in the loss function, which simply weights the entire KL-divergence term, instead of surgically weighting the total correlation term. Therefore, the representations it learns are quite different and less specific to the particular images in the dataset.

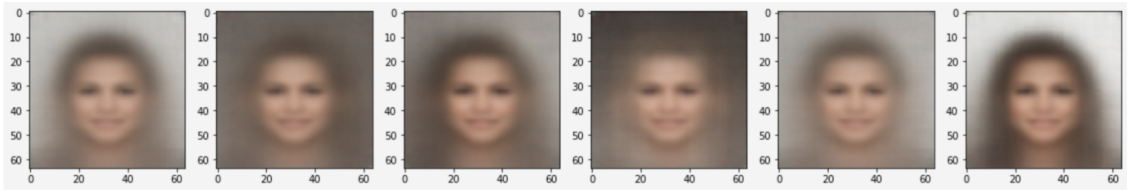


Figure 3-2: β-VAE reconstructions of images in Figure 4-1a

Compared to Figure 4-1a, the reconstructions in Figure 3-2 show less variation. In particular, the background colors in the former roughly match the background colors in the original images, but not in the latter. This causes some of the reconstructions to look similar even if the original images are not very similar; for example, this occurs with the first and fifth images in Figure 3-2.

Nevertheless, β-VAE outputs have significant similarities with the β-TCVAE outputs. For instance, the faces in the output images with both architectures are template (or generic) faces. In addition, both of these architectures show little color variation in the backgrounds of their image outputs.

3.2 β-TCVAE: Tabular Data

LIMIT_BAL	SEX	EDUCATION	MARRIAGE	AGE	PAY_0	PAY_2	PAY_3	PAY_4	PAY_5	...	BILL_AMT4
20000	2	2	1	24	2	2	-1	-1	-2	...	0
120000	2	2	2	26	-1	2	0	0	0	...	3272
90000	2	2	2	34	0	0	0	0	0	...	14331
50000	2	2	1	37	0	0	0	0	0	...	28314
50000	1	2	1	57	-1	0	-1	0	0	...	20940
...	...	...	...	...	...	...	...	...	...	...	...
220000	1	3	1	39	0	0	0	0	0	...	88004
150000	1	3	2	43	-1	-1	-1	-1	0	...	8979
30000	1	2	2	37	4	3	2	-1	0	...	20878
80000	1	3	1	41	1	-1	0	0	0	...	52774
50000	1	2	1	46	0	0	0	0	0	...	36535

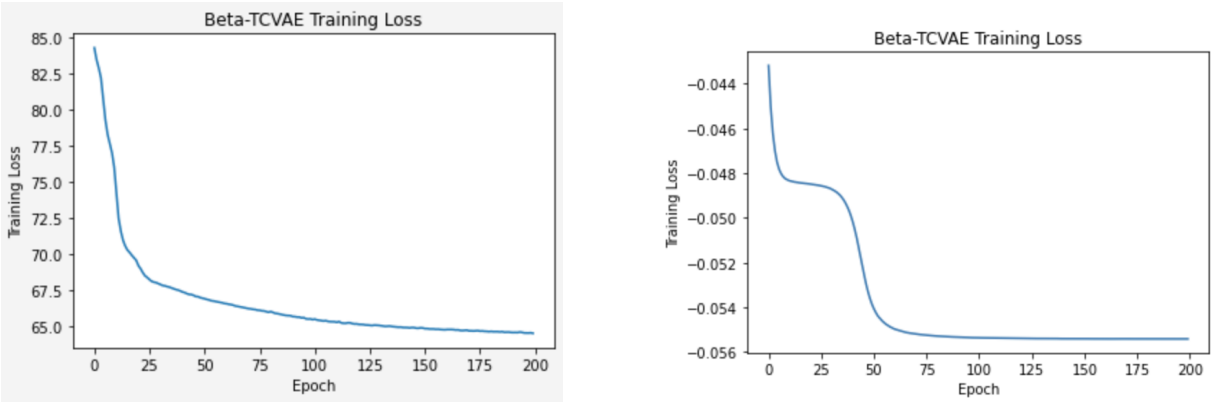
Figure 3-3: Sample from the Taiwan credit score dataset [18]

As it currently stands, the β-TCVAE library only supports image datasets. We extend the usage of our MSVAE to non-image data, with the Taiwan credit score dataset in [18]. As

shown in Figure 3-3, the dataset is tabular in nature. Some columns, such as "BILL_AMT4", contain numerical data, while other columns, such as "SEX", contain categorical data.

Our tabular data uses a different architecture for the β-TCVAE encoder and decoder from the celebA dataset, since image data is multidimensional in nature while financial data is one-dimensional. Therefore, the encoder and decoder use linear layers for financial data, as opposed to convolutional layers for image data. The code for these architectures can be found in Listings C.1 and C.2 in Appendix C. The training loss function, however, is similar; it is the same cross-entropy loss function as in [3]. To ensure valid inputs to our loss function, however, we perform min-max normalization so that the data values are between 0 and 1.

3.3 Loss Functions



(a) Training Loss: celebA dataset

(b) Training Loss: Taiwan credit score dataset

Figure 3-4: Comparison of β-TCVAE Training Loss

We use the loss function $\mathcal{L}_{\beta-TCVAE}$ as described in Section 2.3.2. By surgically weighting the the total correlation (TC) term by β, we attempt to also make the resulting latent representation more disentangled.

The difference in network architecture between the β-TCVAE for the celebA dataset (as described in Section 3.1) and the Taiwan credit score dataset (as described in Section 3.2) contributes to differences in the progress of the loss function throughout each epoch.

As shown in Figure 3-4, the loss function for image data is entirely convex for the celebA dataset. On the other hand, the β-TCVAE training loss for the Taiwan credit score dataset [18] plateaus temporarily and then decreases once again. This temporary plateauing behavior during training often makes it difficult to tell whether we have trained for enough epochs; the loss function graph for the first 25 epochs would be entirely convex, but it does not reflect the fact that the plateau at the end is temporary. Therefore, when training a β-TCVAE for tabular data, we take extra caution to be confident that we have trained for enough epochs.

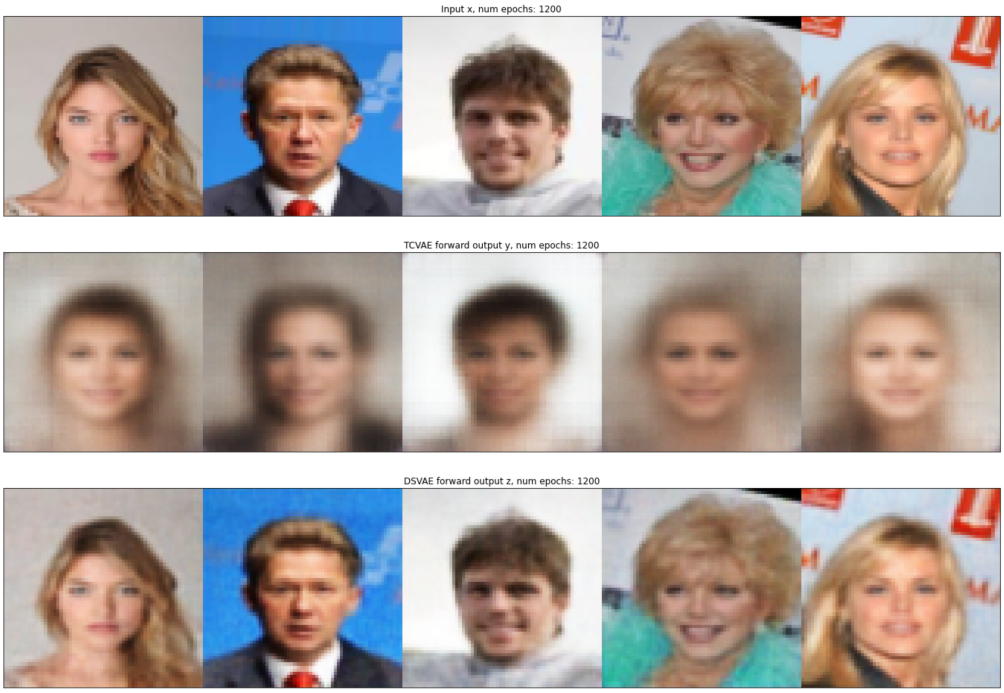
3.4 MSVAE

As stated in Section 2.4, the β-TCVAE produces an incomplete reconstruction of an input, because it only takes the independent factors c into account. The MSVAE improves the reconstruction by incorporating the correlated factors z to reconstruct the original image $h = (c, z)$.

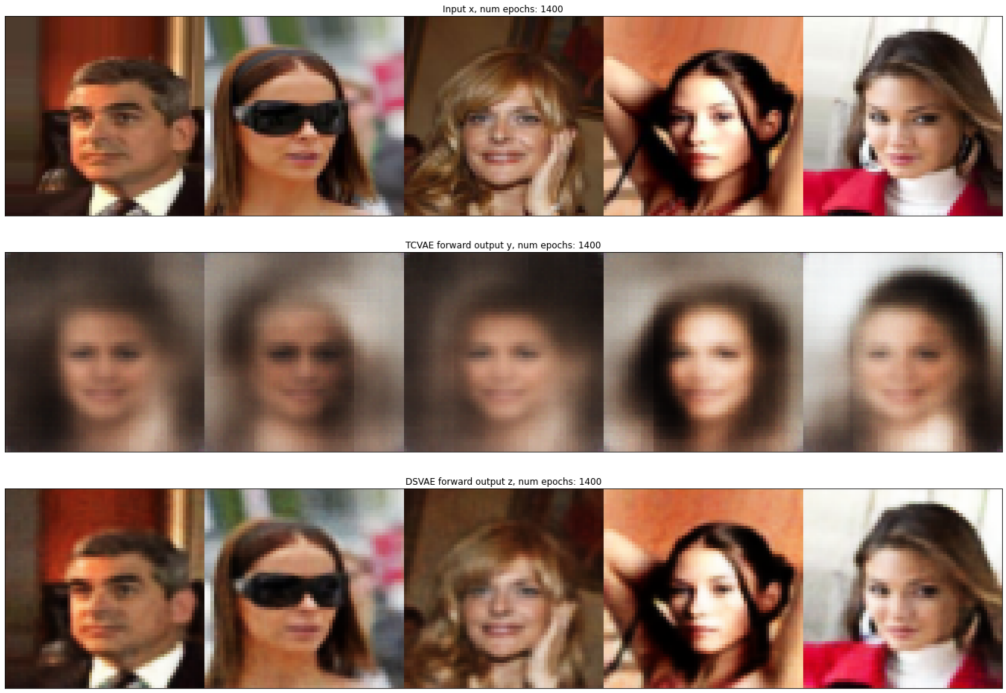
As with every VAE, the MSVAE consists of two portions: the encoder and the decoder. For an image X_i whose β-TCVAE encoded version is Y_i, the MSVAE takes in two inputs: the encoding of independent factors constructed from $c = X_i$ and the encoding of dependent factors constructed from $z = X_i - Y_i$.

Before training the MSVAE, we train an encoder for the dependent factors z so that we can generate inputs. To do this, we take an image x and its β-TCVAE reconstruction y, and we then train the encoder on the residual $x - y$. To train the MSVAE, we train a decoder on y and the *encoded* residual z.

For image data, the decoder and encoder both have convolutional layers, as with the β-TCVAE portion. As shown in Figure 3-5, the MSVAE is able to restore the images mostly to their normal quality (although the resulting images are slightly more grainy than the original images).



(a) MSVAE reconstructions after 1200 training epochs



(b) MSVAE reconstructions after 1400 training epochs

Figure 3-5: Sample MSVAE reconstructions. Each diagram has, in top-to-bottom order: Original image, β-TCVAE reconstruction, MSVAE-reconstructed image

3.5 Conclusion

Our MSVAE library provides the basis for extending the β-TCVAE and MSVAE can be extended to tabular data. Therefore, this MSVAE library is a proof of concept that we can, in fact, use the MSVAE to find a disentangled representation of tabular data. We can use the latent space to generate new, unseen financial data as we were able to do with image data.

As we will see, however, the difference between the types of data results in a noticeable difference in analysis of the β-TCVAE and MSVAE. The loss function plots in Figure 3-5 provide us with a glimpse of how the differences between image data and tabular data manifest themselves in their respective MSVAEs. In particular, the temporary plateaus in the loss function plot for the tabular data MSVAE show that training a β-TCVAE on tabular data can be rather tricky. Indeed, as we will see soon, the analysis of β-TCVAE for tabular data is not entirely straightforward and quite nuanced; the loss function's trend is only the beginning of it all.

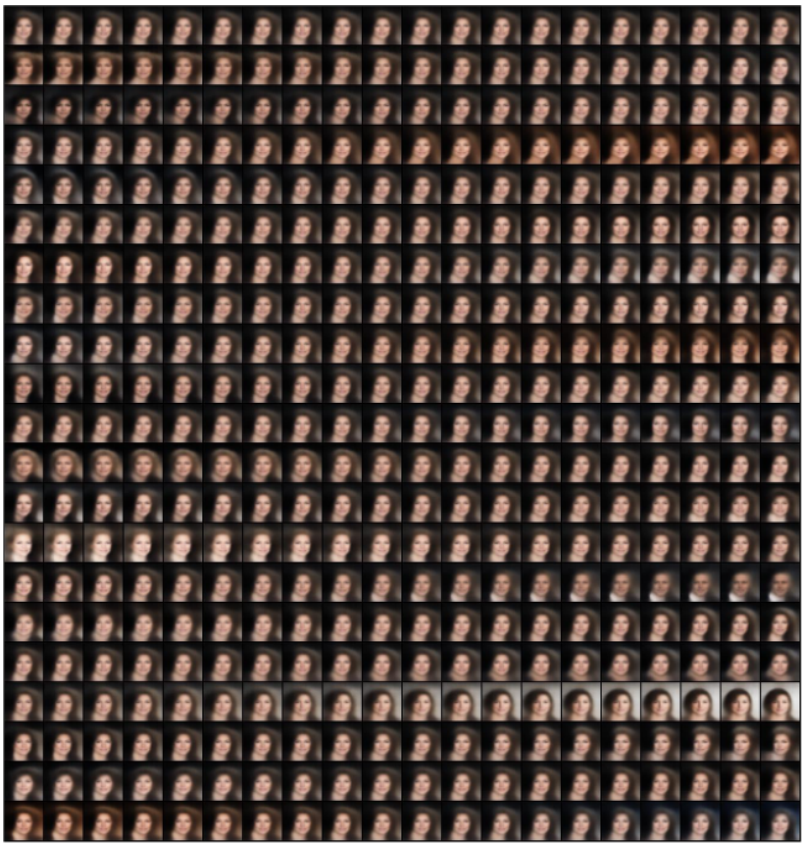
Chapter 4

MSVAE: Generating Unseen Data

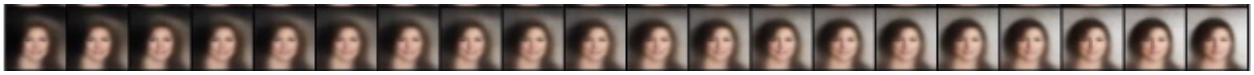
We use the MSVAE architecture implemented in Chapter 3 to generate unseen data. For both the image celebA dataset [9] and the tabular Taiwan credit score dataset [18], we tune different parameters to generate our resulting unseen data. In Sections 4.1 and 4.2, we traverse the latent space of a β-TCVAE to generate unseen data. While the reconstructions of image data are much easier to interpret than for other types of data, we study how to interpret the reconstructions with non-image data. We use different ways to visualize our unseen data for the two datasets and to compare the generated unseen data for different values of the parameters. Finally, in Section 4.3, we investigate inputting elements of different images into the MSVAE.

4.1 Latent Traversals: celebA

Because of the small number of latent factors, we can visualize how changing each single latent factor in the encoding of an image will change the output of the decoded version of the image. For instance, Figure 4-1a visualizes the different images that result in varying the different latent dimensions from a β-TCVAE with a 20-dimensional latent space. Figure 4-1b displays the 17th row of the gallery, which clearly corresponds to the latent dimension that represents background color brightness.



(a) Gallery of changing different latent factors for image reconstruction



(b) 17th Row in Figure 4-1a, the corresponding latent dimension is the background color brightness.

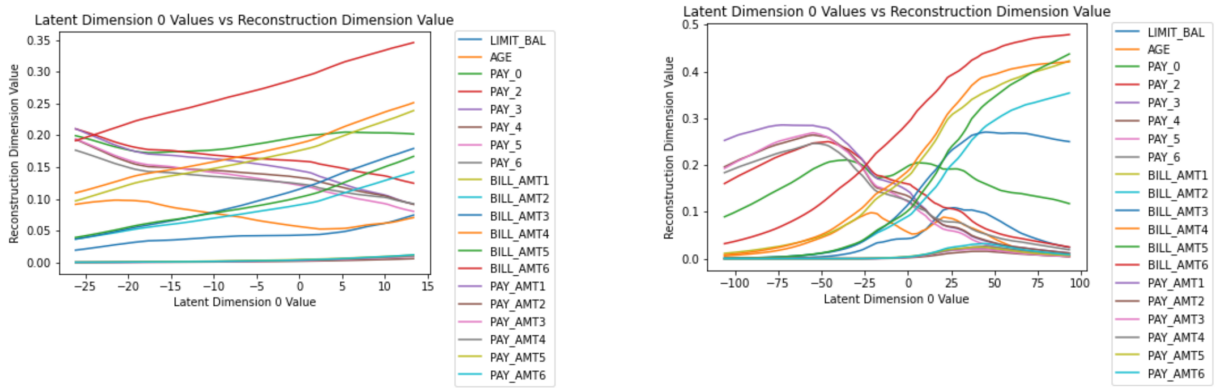
Figure 4-1: Visualization of Changing Different Latent Factors for a 20D Latent Space

The 17th row is not the only row whose feature is empirically describable. For example, we can qualitatively describe the 4th row as the hair color's latent dimension; the hair color changes from black to brown. While the 4th and 17th rows in 4-1a can be empirically described, some other latent dimensions do not have such a straightforward, qualitative description. For example, through visual inspection, the 16th row seems to vary less from left to right, and does not have a clear purpose. To resolve this, we could potentially incorporate some metadata into the β-TCVAE, which includes characteristics such as eye color and hairstyle. Then, we could clamp some latent factors to the known metadata. However, in the absence of any external metadata, it may still be quite difficult to precisely

describe each of the latent factors that the β-TCVAE trains.

4.2 Latent Traversals: Tabular Data

As with the celebA dataset, we visualize latent traversals with the Taiwan credit score dataset. More specifically, we use a line plot, as opposed to a gallery (e.g. Figure 4-1a), to visualize how changes in the latent factors affect each attribute in the reconstruction. We train a β-TCVAE with $\beta = 2$, 5 latent dimensions, and learning rate 10^{-4}. Then, we take a single data point from the Taiwan credit score dataset and put it through the encoder to get its latent representation. We then vary a single latent dimension (dimension 0, in our case). In Figure 4-2, for each column in the dataset, we plot how the reconstruction value varies according to the value of that latent dimension.



(a) Smaller Scale Plot

(b) Large Scale Plot

Figure 4-2: Traversals for Latent Dimension 0 for Taiwan Credit Score Dataset [18] on different scales. The legend shows the data's attributes (i.e. the column names of the original table's data).

The trends in the reconstruction attribute values look quite different when we view them on different scales. On a smaller scale (Figure 4-2a), each attribute has an almost linear trend (with a slight bend). Yet on a larger scale, as shown in Figure 4-2b), there are significant differences between the shapes of the curves for the different attributes. For instance, from left to right, the "PAY_3" curve has an slight upward trend and then curves downward,

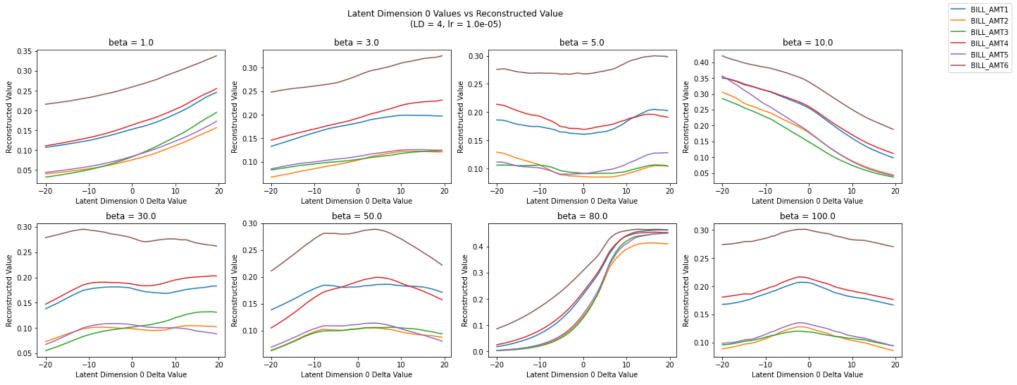
but the "PAY_2" curve has a consistently increasing trend (although the curve is steeper in the middle than at the ends). This indicates that the trends of the reconstruction values are generally consistent on a smaller scale, but the larger scale plot tends to become less predictable — the trends can completely reverse.

4.2.1 β Analysis

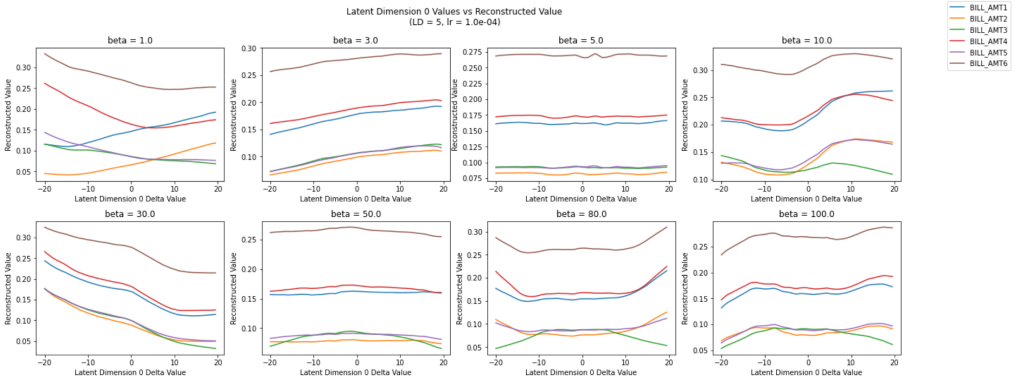
As the β parameter increases in our β-TCVAE, we expect its encoder's latent factors to be more statistically independent. In turn, this should also change the reconstruction values once those latent values are passed through the decoder again. Therefore we investigate the trend of the reconstruction value with respect to β. In Figure 4-3, we show different plots for a single data point and a single latent dimension, but with varying values of β rather than the latent dimension value. We also show these different plots for two different combinations of the following: number of dimensions of the latent space (LD) and learning rate (lr) when training the β-TCVAE. More plots can be found in Figures A-2 and A-3 in Appendix A.

As β increases, the plot of latent dimension vs reconstruction value becomes horizontal. At the same time, the transition to a more horizontal plot is not gradual; in Figure 4-3a, the plot has an upward slope for $\beta = 80$. Nevertheless, by the time β becomes large enough (i.e. $\beta = 100$), the plots are fairly close to horizontal. Nevertheless, in the LD $= 5$, lr $= 10^{-4}$ plots, the trends start becoming horizontal by $\beta = 50$.

With β higher, the β-TCVAE's encoder's objective shifts from constructing a decodable, reconstructible latent representation to learning more disentangled representation. At larger β values, changing the value of a latent dimension has a smaller effect on the decoded representations. In other words, even though the value of a latent dimension may be changed, the decoder will still learn the same representation. As a result, even different raw data points may result in the same β-TCVAE reconstruction.



(a) LD = 4, lr = 10^{-5}



(b) LD = 5, lr = 10^{-4}

Figure 4-3: Plots for Latent Dimension Value vs Reconstruction Value. The legend corresponds to the individual column names. More plots can be found in Appendix A, Figures A-2 and A-3.

4.3 MSVAE Modifications

The two-input nature of the MSVAE enables us to explore the possibility of inputting elements from different images into the MSVAE. While the MSVAE as described above takes in inputs $c = X_i$ and $z = X_i - Y_i$ for a single image X_i, we can incorporate elements of two different images into our final reconstruction. This would enable us to visualize an image under a different environment, i.e. a different set of background (or, in our case, correlated) factors.

4.3.1 MSVAE Modification: Mixup

While the MSVAE produces an output based on two encoded inputs generated from the same image, we do a slight modification where the MSVAE inputs are based on encoded inputs generated from two different images. For MSVAE input $c = X_i$, we investigate what occurs when we let $z = X_j - Y_j$ for some different j (which we call the $(X_i, X_j - Y_j)$ variant — other variants are named similarly).

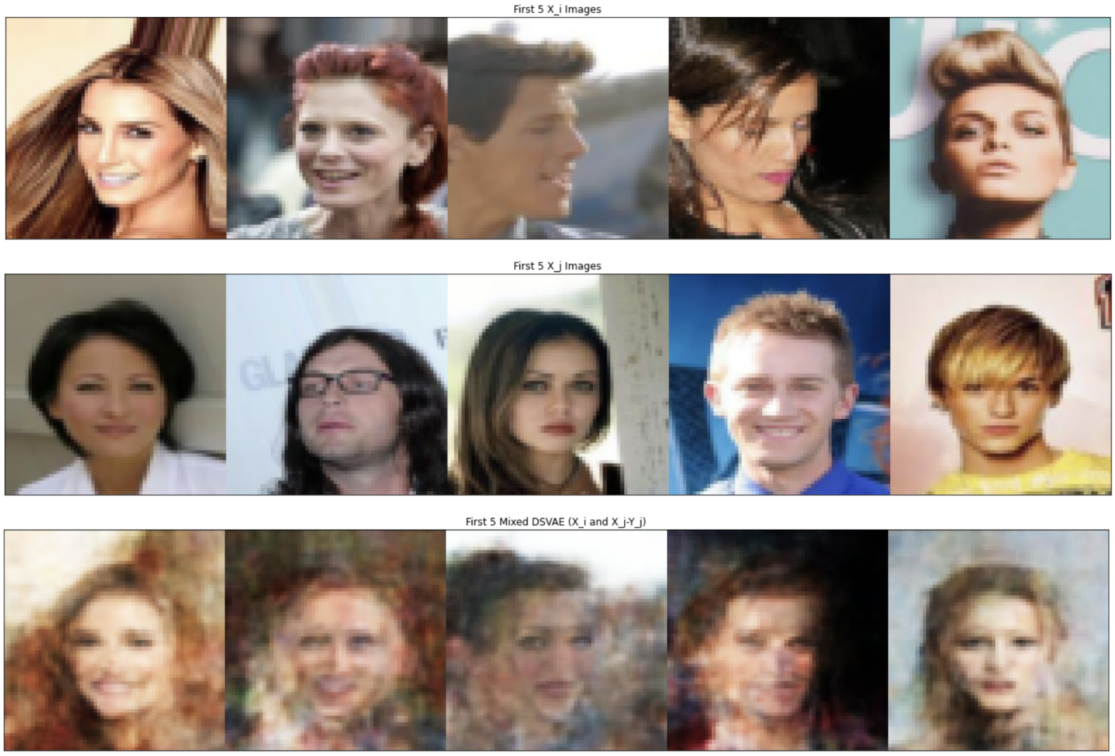


Figure 4-4: MSVAE mixed reconstructions for $(X_i, X_j - Y_j)$ variant. The rows, from top to bottom, contain: X_i images, whose β-TCVAE reconstruction is inputted into the MSVAE; X_j images, whose encoded residuals are inputted into the MSVAE; and the MSVAE reconstructions themselves.

We can empirically observe that the reconstructed images are closer to the X_i images. More technically speaking, the content comes from the X_i images, and the style of the X_j images is transferred to the reconstruction. However, the type of style transfer that occurs in the MSVAE-reconstructed images are different from image pair to image pair. For instance, the third image does a "face transplant" from the X_j image to the X_i image, since the

X_j image's face is visible in the reconstruction; on the other hand, the fifth image does a hairstyle transplant. This variation in results demonstrates that the nature of the difference in nature of the reconstructions between the three mixup variants is largely unpredictable.

4.3.2 MSVAE Modification: More Mixups

We now investigate more variants of our MSVAE mixup paradigm. While we keep c fixed, we try different variations for our z parameter. We then see the resulting images that follow.

For the X_i and X_j images in Figure 4-4, here are the mixups for the (X_i, X_j) variant:

Figure 4-5: MSVAE mixed reconstructions (X_i, X_j), where X_i are the images in the first row of and X_j are the images in the second row of Figure 4-4.

and the mixups for the $(X_i, X_j - Y_i)$ variant:

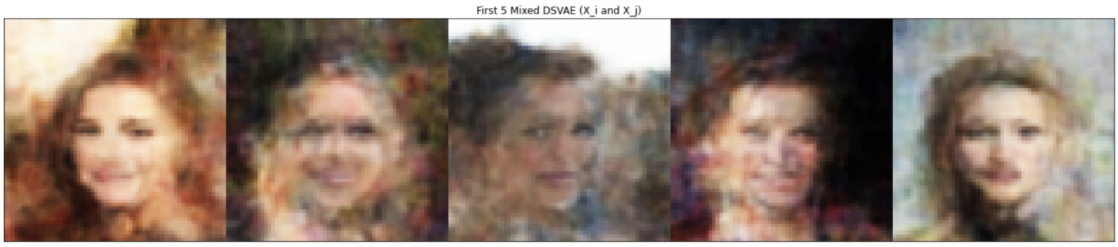


Figure 4-6: MSVAE mixed reconstructions $(X_i, X_j - Y_i)$, where X_i are the images in the first row of and X_j are the images in the second row of Figure 4-4.

Through empirical observation, it appears images in Figures 4-5 and 4-6 look mostly similar to the mixup image reconstructions for the $(X_i, X_j - Y_j)$ variant in Figure 4-4. Even the style transfers are the same as the $(X_i, X_j - Y_j)$ variant. For example, in the third image, the "face transplant" from the X_j image to the X_i image occurs in all three mixup variants.

However, there are some subtle differences. For instance, for the leftmost image pair in Figure 4-4, the leftmost image appear to transplant the closed mouth from the X_j image into the (X_i, X_j) and $(X_i, X_j - Y_i)$ variants but not for the $(X_i, X_j - Y_j)$ variant. Nevertheless, these differences are specific to the two images that are being mixed; for example, the "mouth transplant" is specific to the leftmost image, and does not seem to occur in the other images. In this sense, the similarities between the variants still overshadow these slight differences.

4.4 Conclusion

For the β-TCVAE, the most important way of generating unseen data is to vary a single latent dimension after passing data through an encoder, and then decode it accordingly. To a certain extent, we are able to get the β-TCVAE to learn generative factors of the celebA images, such as hair color and background color. However, in the absence of any external metadata, not all the latent factors we learned are necessarily useful. Therefore, incorporating external metadata could potentially improve our results.

Our current paradigm for the celebA dataset does not work quite so well for the Taiwan credit score dataset, since the data cannot be so easily visualized. Therefore, we use line plots instead of a gallery to visualize trends in each of the columns as we vary a latent dimension. Unlike the celebA dataset, however, the latent dimensions of the corresponding β-TCVAE are not easily described quantitatively. However, we do believe that there are subtle explanations that underlie how these latent factors influence the data; perhaps some domain knowledge of finance would enable us to find such an explanation.

The image nature of the celebA dataset through our MSVAE modification is convenient in another way: we can combine it with the two-input nature of the MSVAE to generate more unseen data. By combining inputting elements of two different images into the inputs of the MSVAE, we can visualize how one image looks under a different environment, such as the "face transplants" as described in Section 4.3. Through this, we were able to see how the different mixup types still generated mostly similar images.

We could also extend the MSVAE modification paradigm to tabular financial data. For

example, a financial dataset A might reflect a time of economic growth; however, we might be interested in how the data looks like under a recession. To see the data, we might take a secondary financial dataset B that reflects a recession and replace the z input with a value based on our the dataset.

Chapter 5

Disentanglement and Statistical Metrics

Given a dataset, the main purpose of the β-TCVAE is to find a disentangled representation of the dataset without supervision. However, the representation that the β-TCVAE learns is unlikely to be perfectly disentangled; there will likely be redundant information between different the different latent dimensions learned. In addition, finding how well a given β-TCVAE satisfies our goal of disentanglement is not a straightforward task. For example, the process of computing a certain disentanglement metric may not be properly defined for different datasets. We investigate the FactorVAE [7] and the Mutual Information Gap (MIG) [3] disentanglement metrics and evaluate them on our two datasets: the image celebA dataset [9] and the tabular Taiwan credit score dataset [18].

5.1 FactorVAE Disentanglement Metric

The FactorVAE disentanglement metric [7] attempts to determine how effectively a β-TCVAE disentangles data by considering the individual latent dimensions; given a table of data, it measures to what extent a change in one dimension of the representation corresponds to a change in exactly one factor of variation.

To evaluate the extent to which data is disentangled, we use the FactorVAE disentanglement metric based on the one shown in Figure 2 in [7]. To calculate the FactorVAE

disentanglement metric, we train and evaluate a majority-vote classifier; the accuracy in these two steps gives us our disentanglement metric. We do 1000 runs of the following process: the classifier chooses a random latent dimension to fix at a constant value and then simply inputs the encoded version of the data through the β-TCVAE's decoder and then through the encoder again. We then calculate the dimension with the lowest variance in the resulting output. The accuracy is then calculated as the proportion of the 1000 runs whose lowest-variance dimension matches the fixed latent dimension. The FactorVAE metric is intuitive; if the fixed latent dimension matches the lowest-variance dimension, it signifies that the fixed latent dimension has low influence from the other latent dimensions as it is passed in through the β-TCVAE's decoder and encoder.

5.1.1 Setup

We run the FactorVAE disentanglement metric for the Taiwan credit score dataset [18] and the celebA dataset [9]. For each dataset, we evaluate the FactorVAE disentanglement metric for different values of the following: number of dimensions of the latent space (LD) and learning rate (lr) when training the β-TCVAE. We keep the parameter $\beta = 2$ for each β-TCVAE variant.

Our code can be found in Appendix C, Listing C.3. It is based on, but not exactly identical to, the code in [10]. For each dataset, we describe the setup for the training of the β-TCVAE, present our results, and then analyze them briefly.

5.1.2 FactorVAE Metric: celebA Dataset

As shown in Table 5.1, the FactorVAE disentanglement metric is able to perform well on the celebA dataset given a well-selected learning rate for each LD parameter. For each LD value of 10, 20, and 25, smaller learning rates result in higher train and eval accuracy figures. In particular, we are able to attain a perfect FactorVAE disentanglement metric values for LD values of 10 and 20.

Larger values of LD also result in lower FactorVAE metric values. Perfect disentanglement

LD	lr	Train Accuracy	Eval Accuracy
10	10^{-3}	0.939	0.940
10	10^{-4}	1.000	1.000
20	10^{-3}	0.271	0.274
20	10^{-4}	1.000	1.000
25	10^{-3}	0.239	0.234
25	10^{-4}	0.505	0.501

Table 5.1: FactorVAE Disentanglement Metric (Second Variant) for celebA Dataset [9] (1000 runs). Latent Space Dimensions (abbreviated LD), and Learning Rates (abbreviated lr)

can become harder to attain for larger values of LD, since a higher latent dimension gives more room for the latent dimensions to contain redundant information. For example, for lr $= 10^{-4}$, train and evaluation accuracy both drop precipitously for LD $= 20$ to LD $= 25$ from 1 to about 0.5. This indicates that for all combinations of values of the other parameters, there is a certain latent dimension threshold after which the FactorVAE metric value is no longer a perfect 1.

5.1.3 FactorVAE Metric: Taiwan Credit Score Dataset

LD	lr	Train Accuracy	Eval Accuracy
5	10^{-3}	0.457	0.427
5	10^{-4}	0.801	0.811
5	10^{-5}	0.591	0.578
10	10^{-3}	0.333	0.276
10	10^{-4}	0.293	0.307
10	10^{-5}	0.148	0.126
20	10^{-3}	0.156	0.119
20	10^{-4}	0.116	0.089
20	10^{-5}	0.113	0.110

Table 5.2: FactorVAE Disentanglement Metric (Second Variant) for β-TCVAE ($\beta = 2$) trained on the Taiwan Credit Score Dataset [18] (1000 runs). Latent Space Dimensions (abbreviated LD), and Learning Rates (abbreviated lr)

By the results shown in Table 5.2, the Taiwan Credit score dataset is, generally speaking, harder to train a β-TCVAE that achieves high disentanglement. Because the values are

rather low for LD = 10 and LD = 20, we attempt to determine how the metric performs for LD = 5, a smaller value than usual.

The FactorVAE metric results are higher for LD = 5 than other LD values. The value lr = 10^{-4} results in an especially high value for the FactorVAE disentanglement metric, which indicates that strong disentanglement is still possible even with tabular data. Although the other lr values do not result in FactorVAE disentanglement metric values that are as high, they are still higher than all FactorVAE disentanglement metric values for the other values of LD. These results for LD = 5 indicates that an especially small latent space is appropriate to describe generative factors for the Taiwan credit score dataset.

5.1.4 FactorVAE Metric: β Analysis on Taiwan Credit Score Dataset

LD	**lr**	β	**Train Accuracy**	**Eval Accuracy**
4	10^{-4}	2	0.531	0.538
4	10^{-4}	4	0.519	0.513
4	10^{-4}	10	0.420	0.397
4	10^{-4}	20	0.497	0.468
10	10^{-4}	2	0.191	0.215
10	10^{-4}	4	0.214	0.198
10	10^{-4}	10	0.241	0.214
10	10^{-4}	20	0.227	0.182
10	10^{-5}	2	0.419	0.381
10	10^{-5}	4	0.401	0.412
10	10^{-5}	10	0.410	0.418
10	10^{-5}	20	0.403	0.398

Table 5.3: FactorVAE Disentanglement Metric for Taiwan Credit Score Dataset [9] (1000 runs), with β varied

We do an extra analysis of the FactorVAE metric of a β-TCVAE with regards to the β parameter for the β-TCVAE loss function, and we present a selection of our results in Table 5.3. A full table of results is in Table B.1 in Appendix B.

The FactorVAE metric is, surprisingly, low variance with respect to β. This indicates that weighting the total correlation (TC) term, as explained in Section 2.3.2, has a smaller

effect than expected on disentanglement. Instead, the metric is largely determined by the LD and lr values, rather than β. For instance, there is a precipitous drop from the LD = 4 metric values to the LD = 10 metric values, but not within each individual (LD, lr) combination. This precipitous drop reflects the idea that a smaller latent dimension is more suitable for building a disentangled representation of the Taiwan credit score dataset; β is not an important factor in determining such disentanglement. With a higher latent dimension, the latent representation is more likely to be redundant, causing the sharp drop in the FactorVAE metric that even higher values of β are unable to overcome.

5.1.5 Comparison: celebA and Taiwan Credit Score Dataset

The FactorVAE disentanglement metric results with the Taiwan Credit Score dataset particularly underscores the difficulty in disentanglement as LD gets larger. The drop in train and evaluation accuracy is quite precipitous in the Taiwan Credit Score dataset, especially compared to the celebA dataset. It is still important to note that the FactorVAE disentanglement metric drops with a higher LD value for the celebA dataset, but the drop is not as significant. Due to the small dimension of each individual data point of the Taiwan Credit Score dataset (especially compared to the celebA images), the FactorVAE disentanglement metric would infer that the data reflects a smaller number of generative factors. This makes the majority-vote classifier more sensitive to the latent dimension, causing the precipitous drop in the FactorVAE disentanglement metric. However, this sensitivity provides us with a silver lining: the range of possible latent dimensions for a smaller-dimensional dataset is much narrower, which makes the corresponding viable LD range smaller.

5.2 Mutual Information Gap

In this section, we explore the Mutual Information Gap (MIG) disentanglement metric from [3]. We create a MIG library which computes the MIG disentanglement metric for both tabular data and image data. To calculate the MIG metric, we assume that the original

data is generated from a distribution $q(z_j, v_k)$, with z_j representing latent variables and v_k representing ground truth factors. We use this joint distribution to calculate the mutual information $I_n(z_j; v_k)$. We then determine how well the axes of the latent variables are aligned with respect to ground truth factors. For each ground truth factor v_k, we calculate the difference between the top two mutual information values. The MIG metric is a weighted average of these differences over all ground truth factors, which results in a value from 0 to 1. (A more detailed description of the weighted average mechanism is in [3].)

The MIG metric formulation addresses two important characteristics of disentangled representations. First, if latent variable axes are not aligned well with respect to ground truth factors, the latent variables can contain information about multiple ground truth factors. Second, for any ground truth factor, there should only be one latent variables that contains information about it.

Finally, it is important to point out the key difference between the MIG metric and the FactorVAE metric: the MIG metric depends on how well the data can be disentangled based on ground truth factors, while the FactorVAE simply measures the amount of disentanglement of a dataset without respect to other factors.

5.2.1 Setup

As with the FactorVAE metric, we run the MIG disentanglement metric on the celebA dataset [9] and Taiwan credit score dataset [18]. We evaluate the MIG disentanglement metric on different values of three different factors: number of latent space dimensions (LD), learning rate when training the β-TCVAE (lr), and the β parameter of the β-TCVAE. For each dataset, we describe the dataset-specific setup for the training of the β-TCVAE, present our results, and then analyze them briefly.

5.2.2 Analysis: Taiwan Credit Score Dataset

For the Taiwan credit score dataset, we use the categorical variables (SEX, EDUCATION, MARRIAGE, AGE) in the data table as our ground truth factors and evaluate the MIG

based on the assumption that all other factors are generated based off of those categorical variables. We vary the number of latent dimensions (LD) and the learning rate (lr) of our trained β-TCVAE. We calculate the MIG score, which is calculated over all categorical variables with respect to all latent dimensions, and we plot those MIG scores in Figure 5-1. Because of the large volume of results, we display our MIG results here as a line plot. (An actual table of results can be found in Table B.1 in Appendix B.)

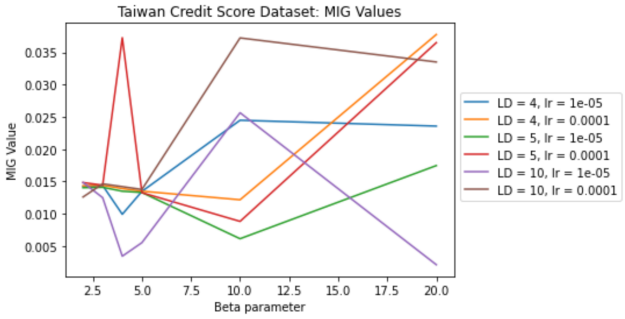
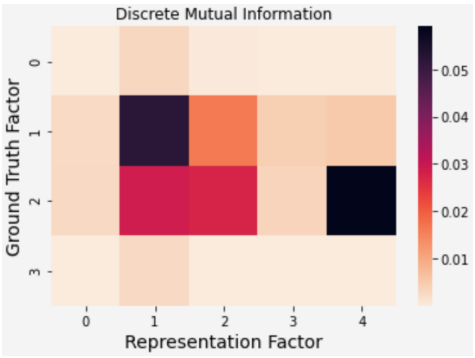
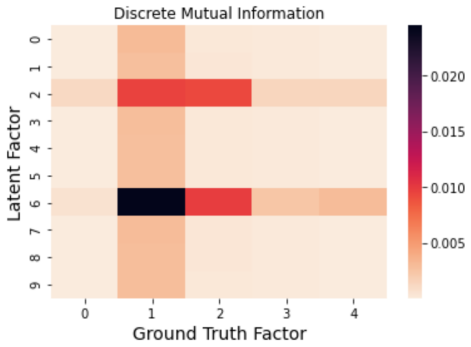


Figure 5-1: MIG values for the Taiwan dataset. For different (LD, lr) pairs, we plot the values for $\beta = 2$, 3, 4, 5, 10, and 20.



(a) LD = 4, lr = 10^{-5}, $\beta = 2$



(b) LD = 10, lr = 10^{-5}, $\beta = 4$

Figure 5-2: Mutual Information (MI) heatmap between ground truth factors and latent factors (labeled "representation factors"). The LD parameter corresponds to the number of latent factors.

In general, all of our MIG metric values are an order of magnitude lower than the Factor-VAE metric values. While the MIG values are close to each other for the smaller values of β, it becomes apparent that by $\beta = 20$ that for every learning rate, the MIG metric is higher when the LD value is lower. This signifies that if there are too many latent dimensions, the

trained β-TCVAE will cause two latent dimensions to store redundant information. This reaffirms that redundancy among latent dimensions decreases MIG. At the same time, even with a low number of latent dimensions, the categorical variables still poorly align with the true ground truth factors, causing the MIG values to be low.

To examine why the MIG values are so low, we analyze the mutual information (MI) values $I_n(z_j; v_k)$ between the latent variables and the ground truth factors. In Figure 5-2, we plot a heatmap for the MI values for two different settings: LD = 4, lr = 10^{-5}, $\beta = 2$; LD = 10, lr = 10^{-5}, $\beta = 4$. More MI heatmaps can be found in Appendix B in Figure B-1. With a higher number of latent dimensions, there are two ground truth factors that appear to have much higher mutual information with the latent factors. This implies that the calculation of the mutual information is largely dominated by these two ground truth factors.

Nevertheless, the most important conclusion to make of all is the obvious one: the categorical variables do not have generative influence on the numerical variables (e.g. the "BILL_AMT" and the "PAY" values). However, this still leaves the possibility open of analyzing the MIG metric under different circumstances: a dataset with observable potential generative factors. In this vein, we analyze the MIG metric for the celebA dataset next.

5.3 Analyzing Reconstructions for Tabular Data

In this section, we statistically analyze the β-TCVAE reconstructions of the tabular data in the Taiwan credit score dataset [18] and compare them to the original dataset. We vary the following parameters: LD (latent space dimension), lr (learning rate), and β.

5.3.1 Empirical Analysis: β-TCVAE Reconstruction Distribution

We first empirically analyze and compare the β-TCVAE reconstructions of the Taiwan credit score dataset to the original dataset.

As the β parameter for our β-TCVAE is varied, we expect the reconstructions to change as well. When β is varied, the Total Correlation (TC) term is weighted heavier. Therefore,

we expect the β-TCVAE's latent representations to become more statistically independent from each other. To analyze whether this is the case, we first find an empirical way of looking at statistical independence: we do so by examining the distribution of the β-TCVAE reconstruction values in the "BILL_AMTi" columns for $i = 1, 2, 3, 4$ in the Taiwan credit score dataset. We also compare these distributions to the original distribution, which is displayed in Figure 5-3.

We plot the distribution of these values in Figure 5-4 for different β parameters on β-TCVAEs, trained on LD $= 4$ and lr $= 10^{-5}$. For small values of β, the "BILL_AMT" columns appear to be skewed to the right, like the original dataset in Figure 5-3. As β is increased, the distributions of the "BILL_AMT" variables become more normally distributed. This shift towards a normal distribution indicates that each "BILL_AMT" column is becoming more independent, and hence disentangled, in the reconstructed data.

We plot more histograms for different LD and lr configurations in Appendix A, Figures A-4 through A-8. Some of these plots, especially the lr $= 10^{-4}$ ones, exhibit rather different characteristics from the LD $= 4$, lr $= 10^{-5}$ plot in Figure 5-4. The plots in Figures A-4 and A-6 (and even almost Figure A-8) continue to be skewed to the right for $\beta = 80$. This suggests that a higher learning rate may cause the data distribution to align more with the original data, which is also skewed to the right.

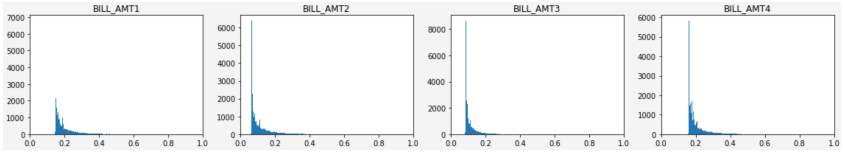
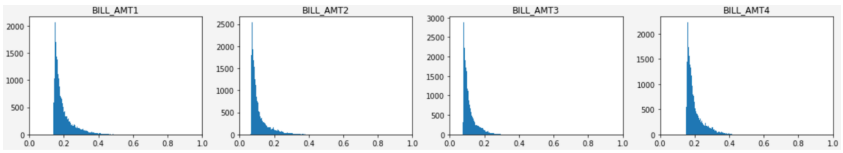
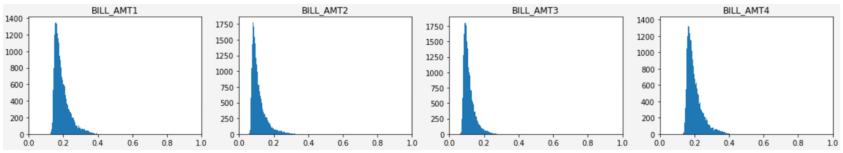


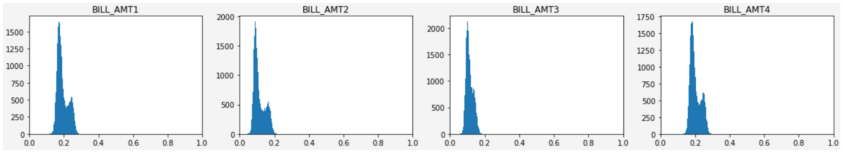
Figure 5-3: Distribution of original Taiwan credit score data



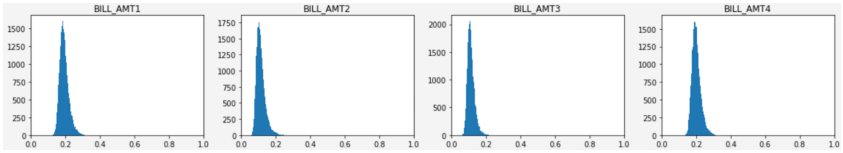
(a) $\beta = 1$



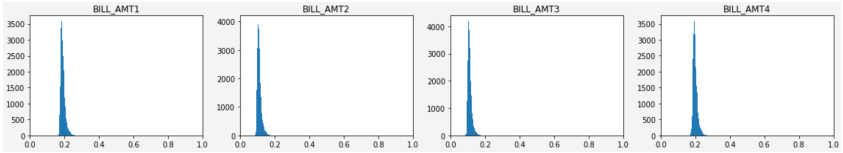
(b) $\beta = 5$



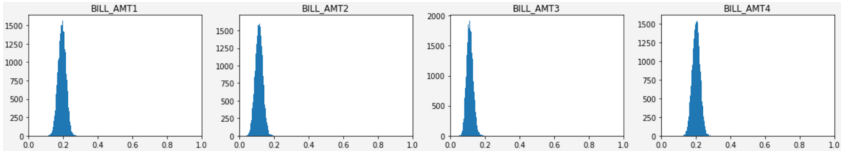
(c) $\beta = 10$



(d) $\beta = 30$



(e) $\beta = 50$



(f) $\beta = 80$

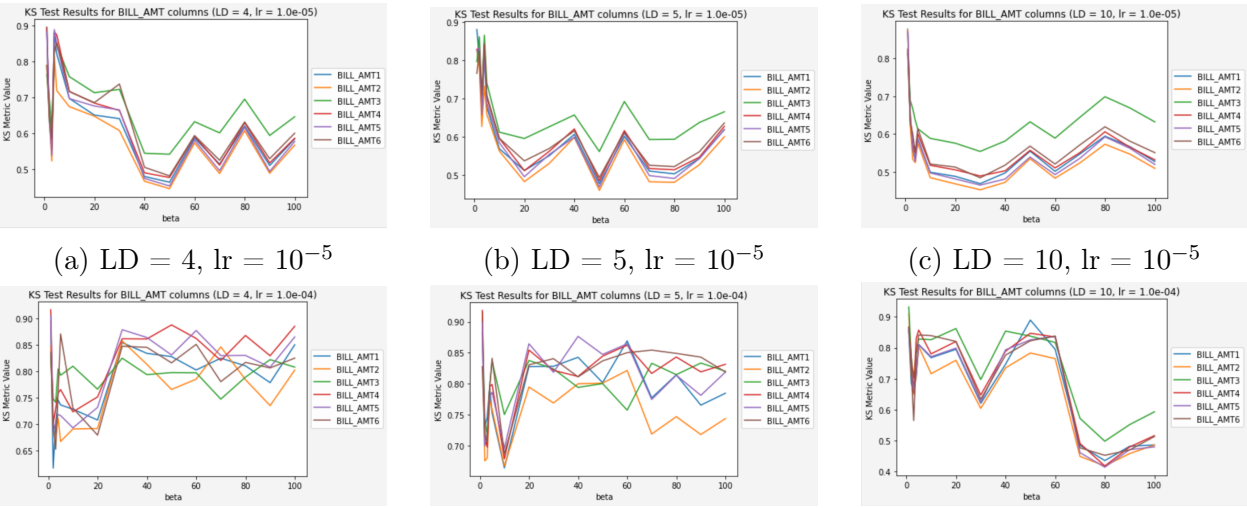
Figure 5-4: Plots for Distribution of Reconstruction Values for Taiwan credit score dataset [18] for LD = 4, lr = 10^{-5}. More plots can be found in Appendix A, Figures A-4 through A-8.

5.3.2 Quantitative Analysis: KS Test

We quantitatively analyze β-TCVAE reconstructions of the Taiwan credit score dataset and compare them to the original data.

As the β parameter for our β-TCVAE is varied, we would expect the reconstructions to be more disentangled, which in turn would cause the distribution of the β-TCVAE reconstructed data to not fit as well with the distribution of the original data. Quantitatively speaking, the curve fit between the original data's distribution and the reconstructed data's distribution should decrease as β increases. We use the Kolmogorov-Smirnov (KS) test [12] metric to evaluate goodness of fit between the original data and β-TCVAE reconstructed data, and we use the `KSTest` functions within the Synthetic Data Vault (SDV) library [13] to calculate the metric. The KS test metric takes in two distributions and outputs a value from 0 to 1, with a higher value indicating a better fit.

As in Section 5.3.1, we train β-TCVAEs with different LD (number of dimensions of latent space), lr (learning rate), and β values. In Figure 5-5, we plot the trends for β versus KS test metric for a few LD and lr combinations and show these plots here. We test the following β values: the integers from 1 to 5, and the multiples of 10 from 10 to 100.



(a) LD = 4, lr = 10^{-5} (b) LD = 5, lr = 10^{-5} (c) LD = 10, lr = 10^{-5}

(d) LD = 4, lr = 10^{-4} (e) LD = 5, lr = 10^{-4} (f) LD = 10, lr = 10^{-4}

Figure 5-5: β vs KS test metric trends for different (LD, lr) combinations.

For the smaller learning rate 10^{-5}, the KS test metric is smaller for our largest β value of 100 than for our smallest β value of 1. However, this particular decrease is not a steady decrease; as β approaches values greater than 40, the KS test metric appears to fluctuate. Nevertheless, the range in which this fluctuation takes place is still consistently at a lower

range than the KS test metric for $\beta = 1$. This indicates that there is a certain threshold β_0 where for all $\beta > \beta_0$, the KS test metric will fluctuate within a certain range. These results reveal an important characteristic of the β parameter: past the β threshold β_0, the β-TCVAEs trained are effectively equivalent in terms of goodness of fit between original representations and β-TCVAE reconstructed representations.

For the larger learning rate 10^{-4}, the β-TCVAE is not as effective in distinguishing the reconstruction distribution from the original data's distribution. As with learning rate 10^{-5}, our results fluctuate; however, for LD = 4 and LD = 5, there is no significant decrease in our KS test metric after $\beta = 1$. We also note in Figures A-4 and A-6 in Appendix A, which exactly correspond to these configurations, the distributions of the β-TCVAE reconstructed "BILL_AMT" values are still skewed to the right for our largest value of β. This is consistent with the KS test metric remaining high, since both the original histogram (Figure 5-3) and the reconstruction values histogram are skewed to the right.

In contrast to LD = 4 and LD = 5, the LD = 10 result for learning rate 10^{-4} contains a massive drop in the KS test metric from $\beta = 60$ to $\beta = 70$. This shows that a higher latent dimension gives the β-TCVAE enough expressivity to find a reconstruction that is independent of the original distribution. In addition, the LD = 10 results for both lr = 10^{-5} and lr = 10^{-4} undergo a massive drop in the KS test metric, from $\beta = 3$ to $\beta = 4$ and $\beta = 60$ to $\beta = 70$, respectively. This reaffirms the concept of our β threshold: with a large enough latent dimension, the β-TCVAE will be expressive enough to find a reconstruction of the data that is independent of the original distribution for all $\beta > \beta_0$. At the same time, the learning rate is still a major factor in how large β_0 is; the threshold is higher for higher learning rates.

5.4 Conclusion

The FactorVAE metric shows that it is indeed possible to create a β-TCVAE representation of image and tabular data with high disentanglement. However, because the image data has a significantly higher size than the tabular data, the tabular data requires a significantly

smaller latent dimension in order to satisfy proper disentanglement.

However, since high disentanglement is achievable, the data must still have certain characteristics that make high disentanglement achievable. More specifically, there must still be ground truth factors that do generate our datasets accurately. Finding concrete ground truth factors, however, may be somewhat elusive. The MIG metric from Section 5.2 attempted to test some ground truth factors to see whether we could achieve high disentanglement under the assumption that the data was generated based on these ground truth factors. If such ground truth factors do in fact exist, they will likely be more sophisticated and harder to describe than the ones we have studied.

On a positive note, our library has great potential to guide us towards the right direction in finding potential generative ground truth factors. More specifically, this library gives us the means to investigate how to generate a disentangled representation of a dataset given ground truth factors. In addition, our library is particularly flexible, as it supports both image and tabular data. We have therefore laid the groundwork for exploring and testing out sophisticated forms of ground truth factors, which could lead us to a high MIG disentanglement metric value.

Finally, our statistical analysis of the β-TCVAE reconstructions of tabular data are another source of hope. In particular, higher values of β mostly result in a more Gaussian-shaped distribution for the reconstructed values and lower KS test metrics, especially for higher latent dimensions values and lower learning rates. Therefore, our library provides means to tune the β-TCVAE model to discover the regimes that result in a lower statistical similarity with the original dataset. However, we must still exercise caution when making our conclusions; most importantly, the statistical analysis does not necessarily imply that higher β implies higher disentanglement. Nevertheless, our analysis does show that increasing β may still have some desired effects, but not necessarily the effects that we may have hoped for.

Chapter 6

Conclusions and Future Work

6.1 Conclusions

We investigated the MSVAE model and attempted to extend its application past image data (e.g. the celebA dataset [9]) to tabular data. As with the celebA dataset, we investigated the resulting reconstructions and latent spaces of different β-TCVAEs trained on the Taiwan credit score dataset. After training the parameters of a Gaussian distribution, the MSVAE was able to understand the true distribution of the generated data, which we were able to analyze. In addition, we analyzed disentanglement metrics to determine how well the β-TCVAE loss function worked in pushing the resulting reconstructions towards disentangled representations.

We were successful to a certain extent on the FactorVAE metric; its performance on the celebA dataset, and sometimes even the Taiwan credit score dataset [18], demonstrated that achieving high disentanglement is indeed possible. However, we encountered more difficulties with the MIG metric. We would have liked to show that latent factors capture some information from the known or unknown ground truth factors. Unfortunately, the assumption that ground truth factors from the given tabular data were also generative factors did not enable us to draw such a conclusion, as these factors did not appear as generative ones.

Nevertheless, we were able to generate unseen tabular and image data and examine the trends in the generated data under different environments. For example, in Section 4.3, we were able to use the MSVAE and investigate reconstructions of images in the celebA dataset by mixing elements of two different images into our MSVAE, effectively imposing one image's environment onto another image.

6.2 Future Work

We recommend the following steps for future work; by no means are they the only steps possible:

1. Applications: The β-TCVAE is applicable to many different contexts and settings. For example, we could incorporate known metadata and apply our β-TCVAE to causal settings, as in [11], which uses known disentangled embeddings to predict perturbations to generate new gene expression vectors. In the same vein, we could incorporate metadata such as economic conditions into a financial dataset like the Taiwan credit score dataset and use it to generate embeddings for the dataset in some latent space. Then, by doing latent traversals as in Section 4.2, we could generate new data under different economic conditions.

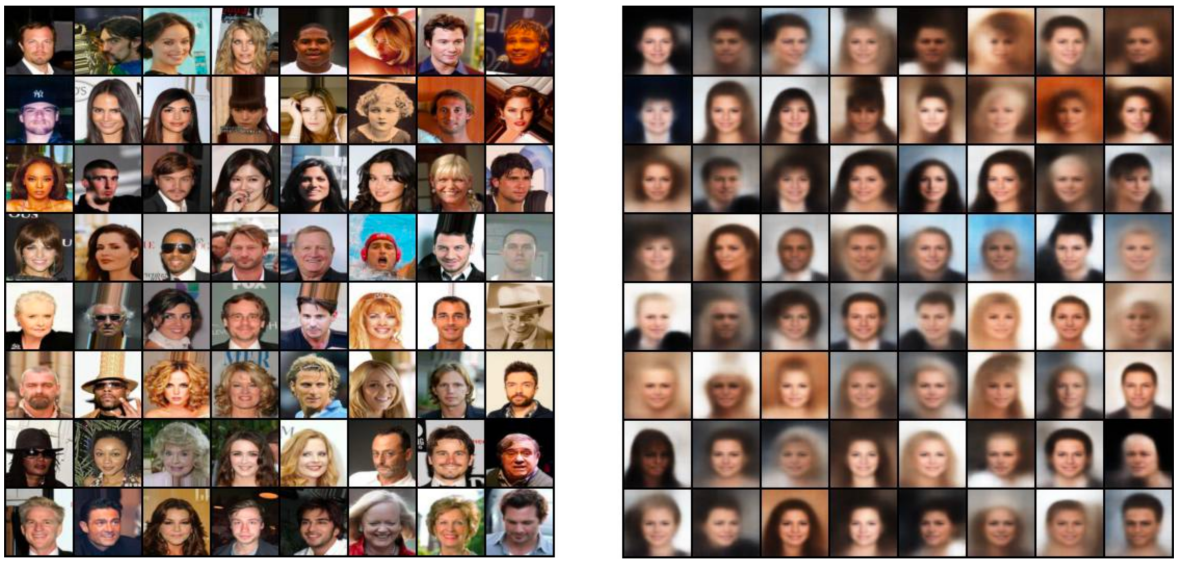
2. Metrics: We could examine disentanglement metrics other than FactorVAE and MIG. A low MIG metric does not necessarily indicate low disentanglement of the data itself, but is instead a measure of how good ground truth factors are relative to the data. In a similar vein, other metrics based on different factors could be more appropriate for the tabular data. As [11] does with medical data, we could aim for disentanglement of data according to financial factors.

3. Loss Functions: We could study different VAE loss functions and investigate the disentanglement representations. The β-TCVAE loss function derived in [3] and restated in Section 2.3.2 is far from the only paradigm that we can view VAEs from — there

may be a better loss function to train a β-TCVAE which may optimize a completely different quantity. We may then be able to compare these results with the results from our current loss function.

Appendix A

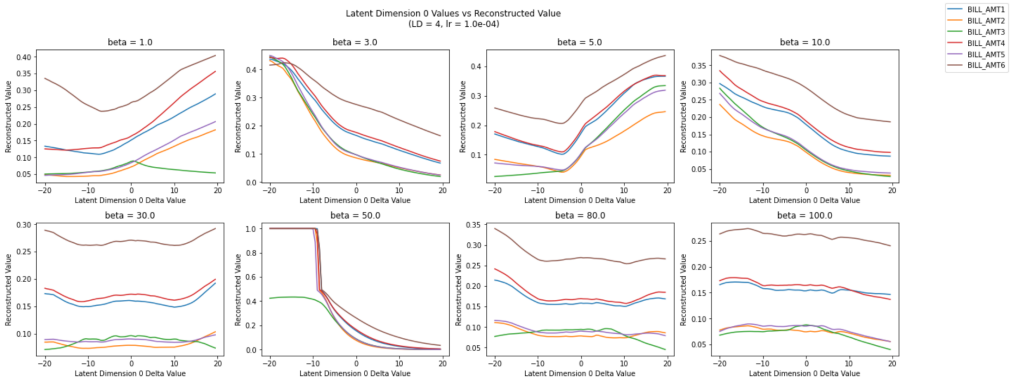
β-TCVAE Analysis



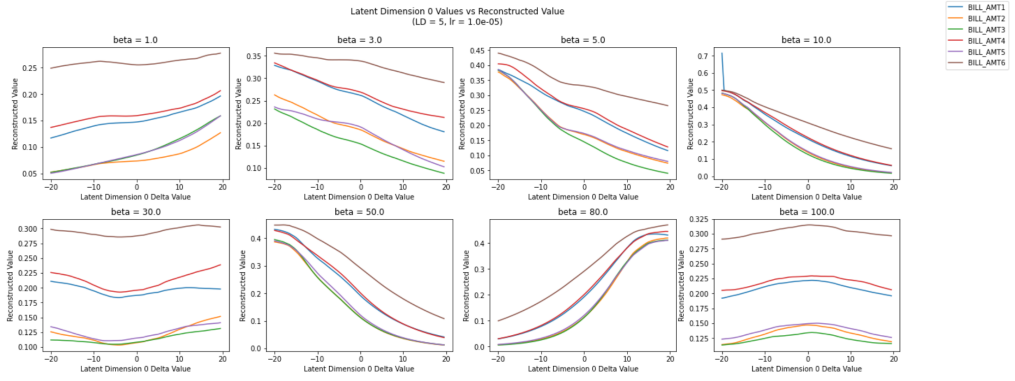
(a) Original celebA [9] images

(b) β-TCVAE reconstructed images

Figure A-1: More samples of β-TCVAE reconstructions for celebA dataset [9]

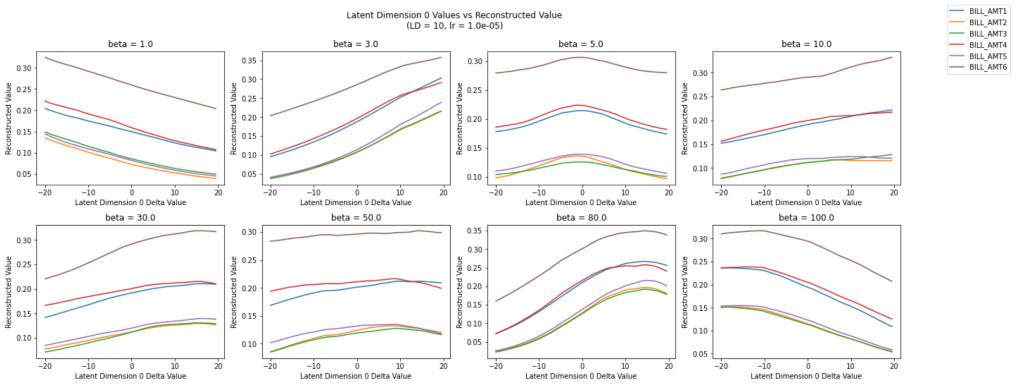


(a) LD = 4, lr = 10^{-4}

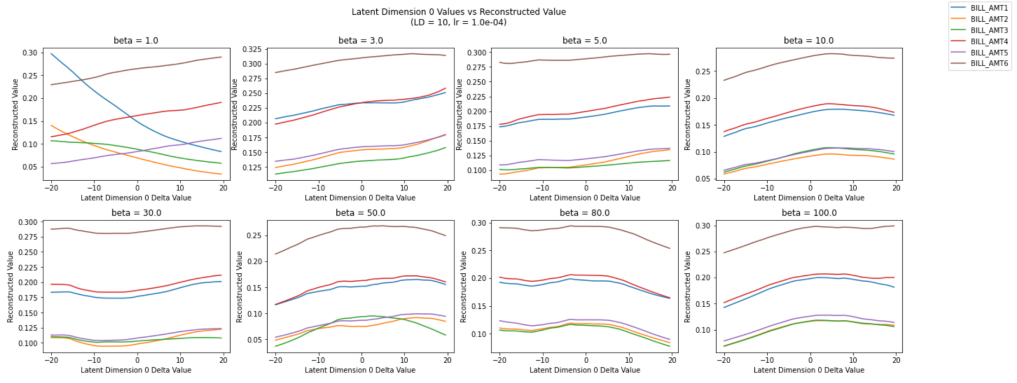


(b) LD = 5, lr = 10^{-5}

Figure A-2: More plots for Latent Dimension Value vs Reconstructed Value. The legend corresponds to the individual column names.

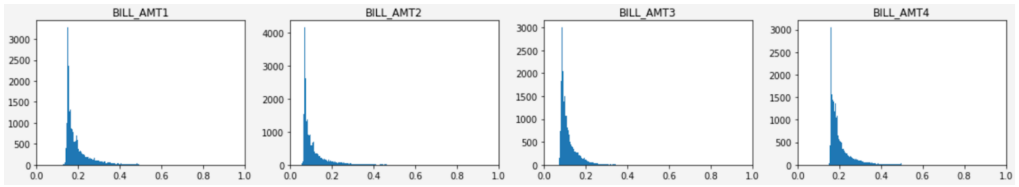


(a) LD = 10, lr = 10^{-5}

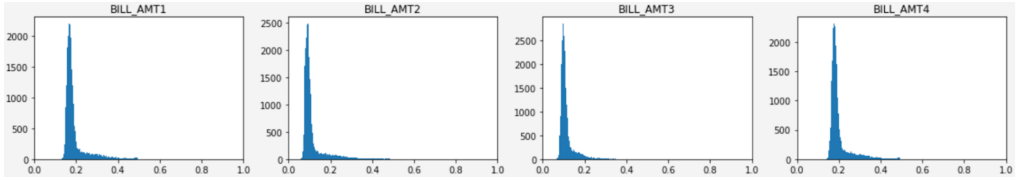


(b) LD = 10, lr = 10^{-4}

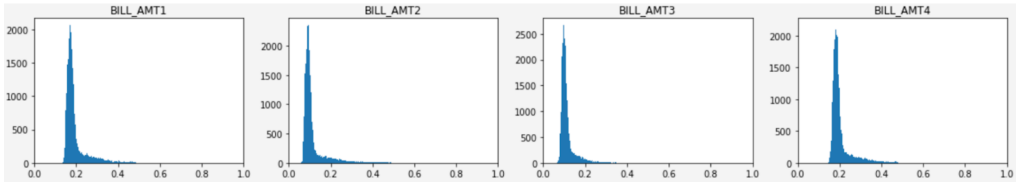
Figure A-3: More plots for Latent Dimension Value vs Reconstructed Value. The legend corresponds to the individual column names.



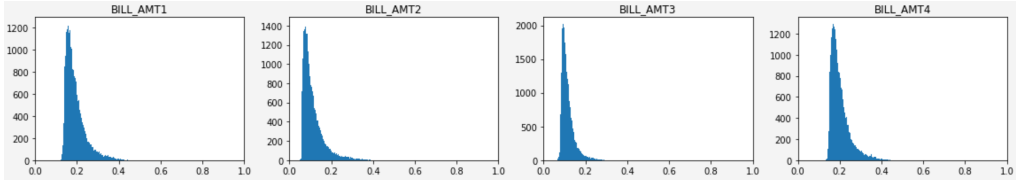
(a) $\beta = 1$



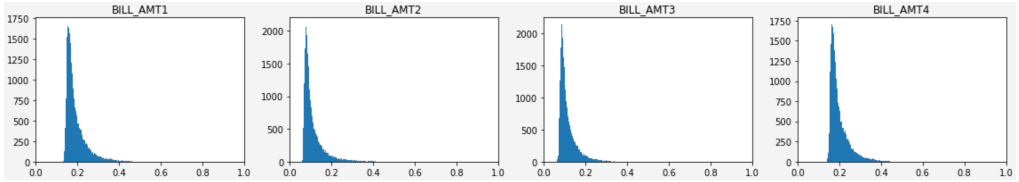
(b) $\beta = 5$



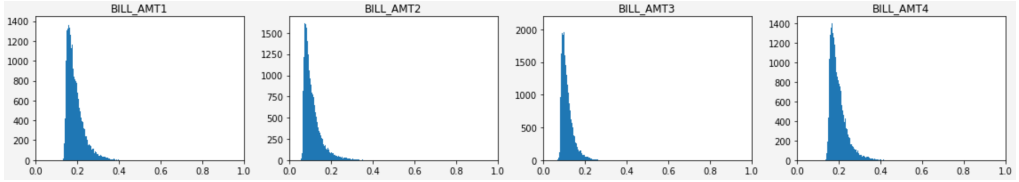
(c) $\beta = 10$



(d) $\beta = 30$

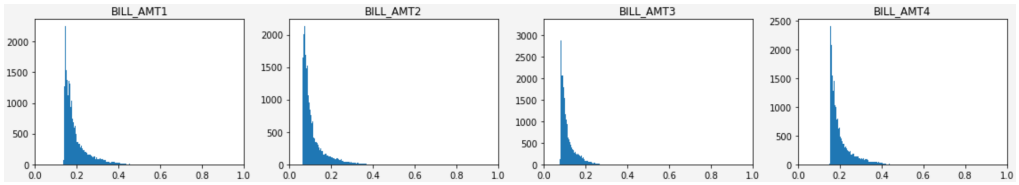


(e) $\beta = 50$

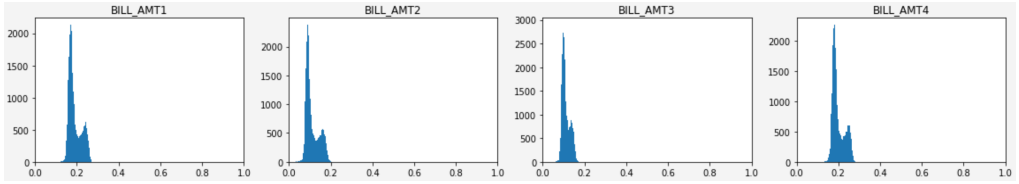


(f) $\beta = 80$

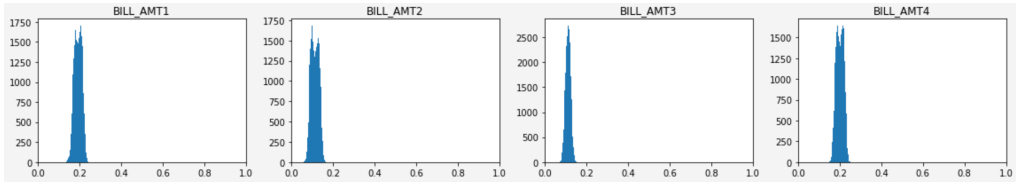
Figure A-4: Plots for Distribution of Reconstruction Values for Taiwan credit score dataset [18] for LD $= 4$, lr $= 10^{-4}$. Note that this differs from most other configurations, in that the plot still remains skewed to the right even for higher values of β.



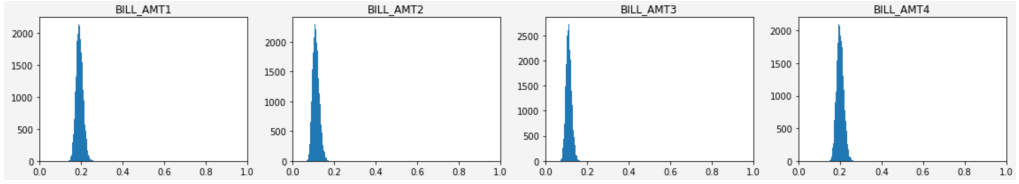
(a) $\beta = 1$



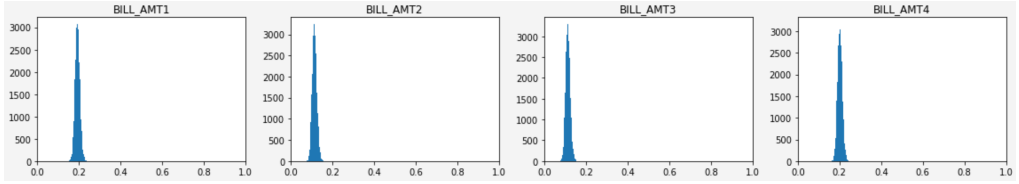
(b) $\beta = 5$



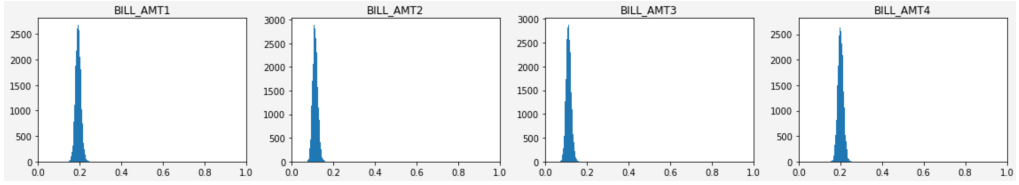
(c) $\beta = 10$



(d) $\beta = 30$

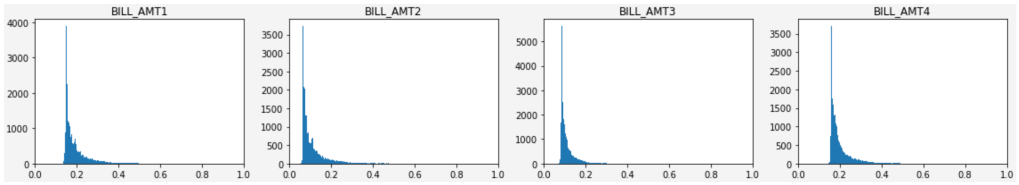


(e) $\beta = 50$

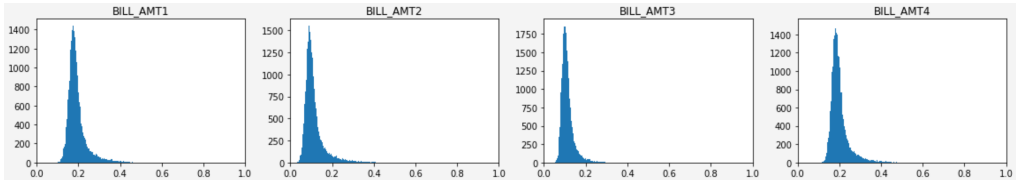


(f) $\beta = 80$

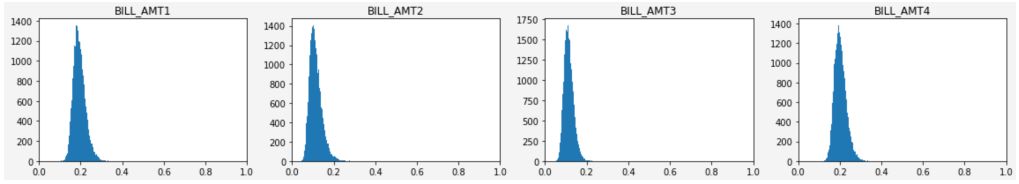
Figure A-5: Plots for Distribution of Reconstruction Values for Taiwan credit score dataset [18] for LD $= 5$, lr $= 10^{-5}$.



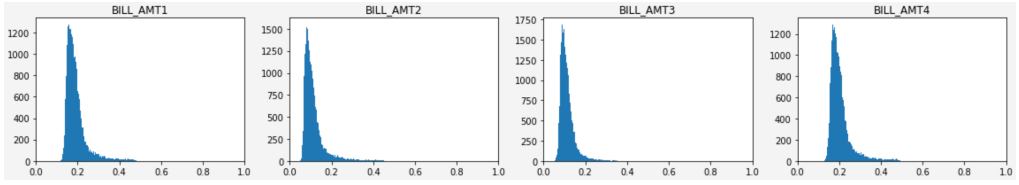
(a) $\beta = 1$



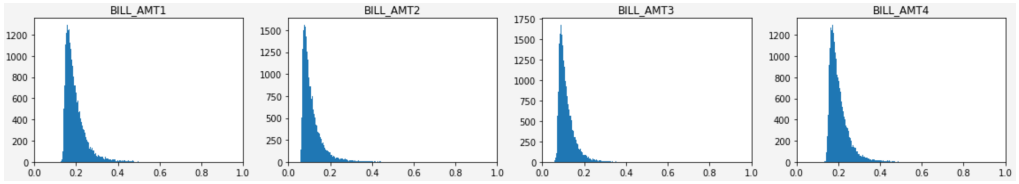
(b) $\beta = 5$



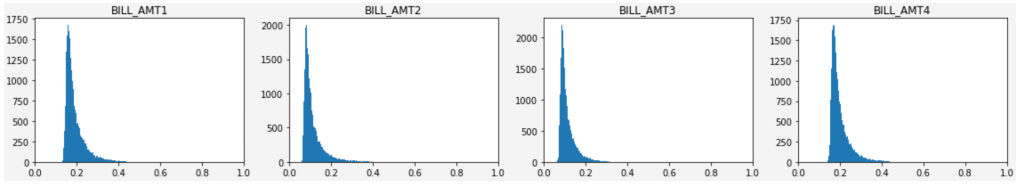
(c) $\beta = 10$



(d) $\beta = 30$

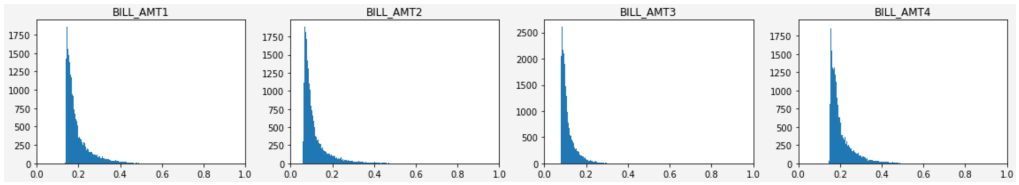


(e) $\beta = 50$

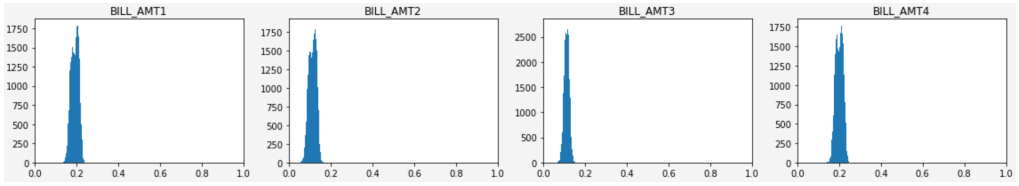


(f) $\beta = 80$

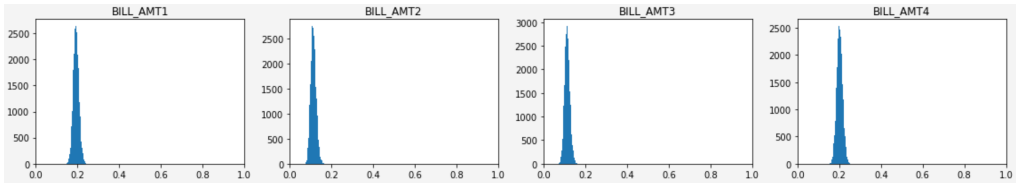
Figure A-6: Plots for Distribution of Reconstruction Values for Taiwan credit score dataset [18] for LD $= 5$, lr $= 10^{-4}$. Note that this differs from most other configurations, in that the plot still remains skewed to the right even for higher values of β.



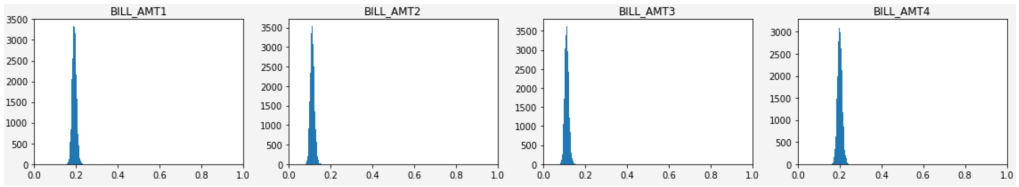
(a) $\beta = 1$



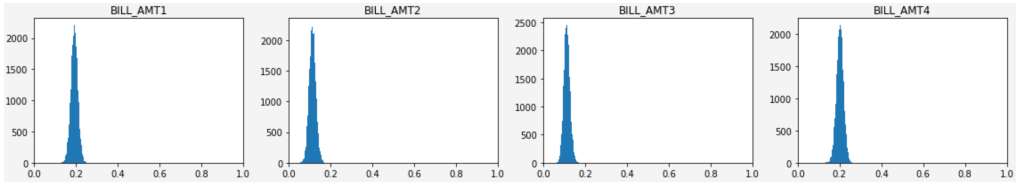
(b) $\beta = 5$



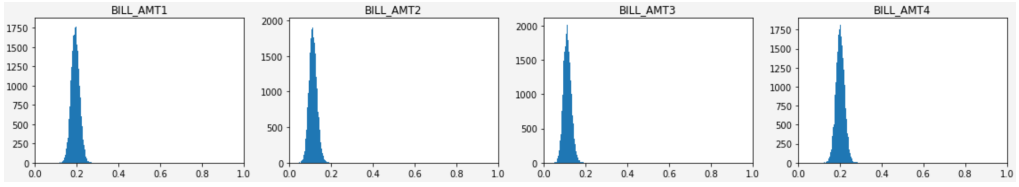
(c) $\beta = 10$



(d) $\beta = 30$

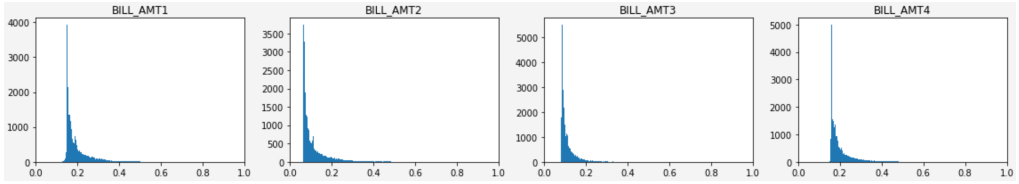


(e) $\beta = 50$

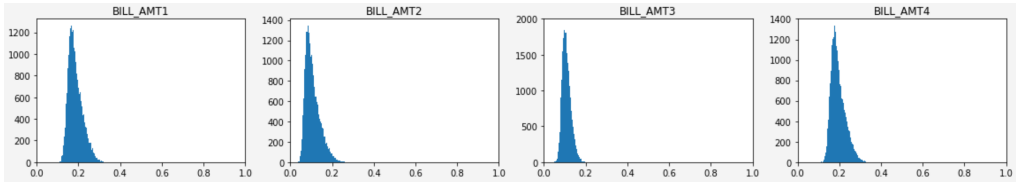


(f) $\beta = 80$

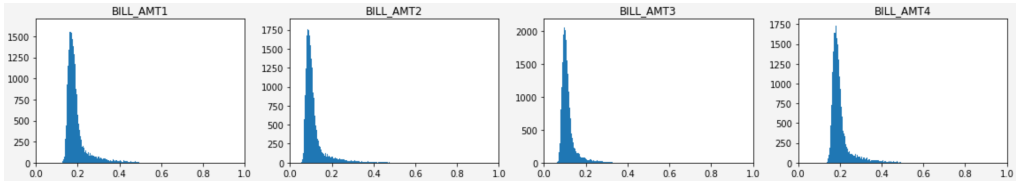
Figure A-7: Plots for Distribution of Reconstruction Values for Taiwan credit score dataset [18] for LD = 10, lr = 10^{-5}.



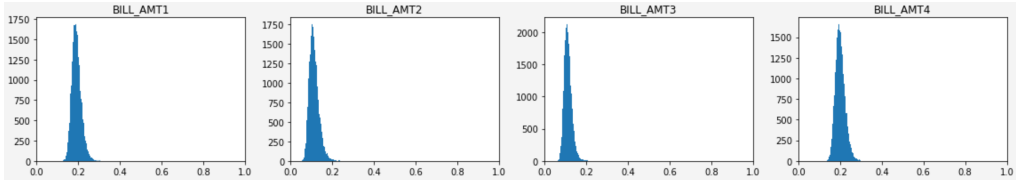
(a) $\beta = 1$



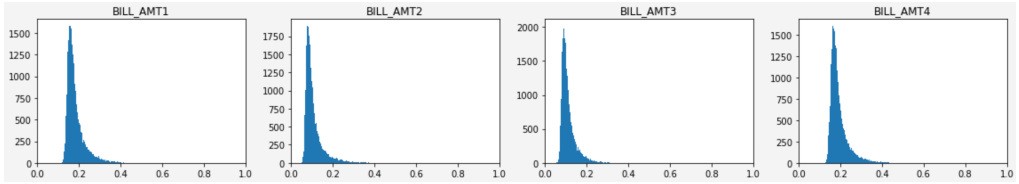
(b) $\beta = 5$



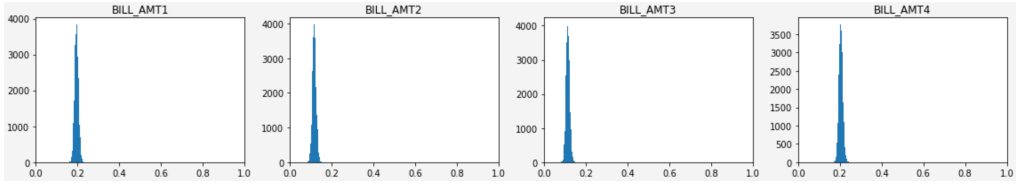
(c) $\beta = 10$



(d) $\beta = 30$



(e) $\beta = 50$

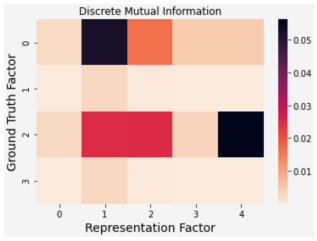


(f) $\beta = 80$

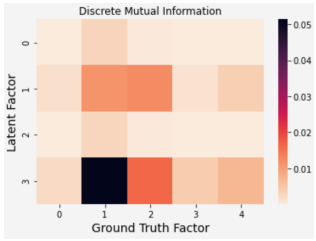
Figure A-8: Plots for Distribution of Reconstruction Values for Taiwan credit score dataset [18] for LD = 10, lr = 10^{-4}.

Appendix B

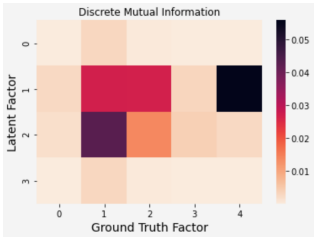
Disentanglement Metrics



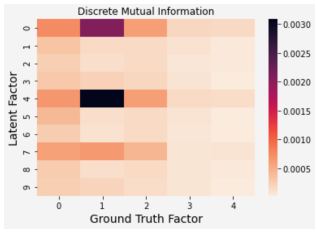
(a) LD = 4, lr = 10^{-5}, $\beta = 3$



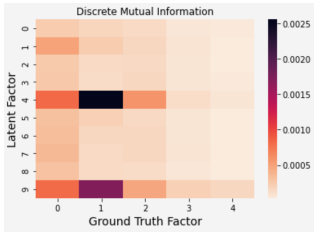
(b) LD = 4, lr = 10^{-5}, $\beta = 4$



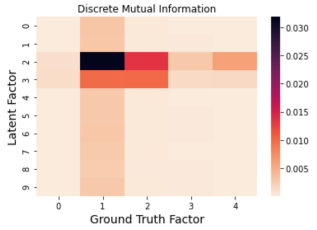
(c) LD = 4, lr = 10^{-5}, $\beta = 5$



(d) LD = 10, lr = 10^{-5}, $\beta = 2$



(e) LD = 10, lr = 10^{-5}, $\beta = 3$



(f) LD = 10, lr = 10^{-5}, $\beta = 5$

Figure B-1: Mutual Information (MI) heatmap between ground truth factors and latent factors (labeled "representation factors")

LD	**lr**	β	**MIG Metric**	**FactorVAE Train Accuracy**	**FactorVAE Eval Accuracy**
4	10^{-5}	2	0.014226	0.533	0.518
4	10^{-5}	3	0.014366	0.518	0.498
4	10^{-5}	4	0.009906	0.498	0.492
4	10^{-5}	5	0.013453	0.523	0.508
4	10^{-5}	10	0.024472	0.523	0.490
4	10^{-5}	20	0.023565	0.505	0.520
5	10^{-5}	2	0.014022	0.431	0.387
5	10^{-5}	3	0.014086	0.403	0.402
5	10^{-5}	4	0.013480	0.422	0.396
5	10^{-5}	5	0.013282	0.421	0.364
5	10^{-5}	10	0.006120	0.373	0.394
5	10^{-5}	20	0.017452	0.416	0.405
10	10^{-5}	2	0.014839	0.419	0.381
10	10^{-5}	3	0.012480	0.386	0.421
10	10^{-5}	4	0.003414	0.401	0.412
10	10^{-5}	5	0.005514	0.400	0.375
10	10^{-5}	10	0.025637	0.410	0.418
10	10^{-5}	20	0.002093	0.403	0.398
4	10^{-4}	2	0.014322	0.531	0.538
4	10^{-4}	3	0.014393	0.657	0.665
4	10^{-4}	4	0.013897	0.519	0.513
4	10^{-4}	5	0.013484	0.512	0.533
4	10^{-4}	10	0.012154	0.420	0.397
4	10^{-4}	20	0.037757	0.497	0.468
5	10^{-4}	2	0.014818	0.619	0.590
5	10^{-4}	3	0.014417	0.457	0.447
5	10^{-4}	4	0.037262	0.388	0.374
5	10^{-4}	5	0.013254	0.409	0.392
5	10^{-4}	10	0.008807	0.420	0.412
5	10^{-4}	20	0.036493	0.248	0.243
10	10^{-4}	2	0.012585	0.191	0.215
10	10^{-4}	3	0.014619	0.219	0.210
10	10^{-4}	4	0.014191	0.214	0.198
10	10^{-4}	5	0.013754	0.181	0.189
10	10^{-4}	10	0.037226	0.241	0.214
10	10^{-4}	20	0.033500	0.227	0.182

Table B.1: Full Table of MIG and FactorVAE Disentanglement Metric for Taiwan Credit Score Dataset [9] (1000 runs), with LD (number of latent dimensions), lr (learning rate), and β varied

Appendix C

Code

Here are the encoder and decoder architecture for the Taiwan credit score dataset [18]:

```
import torch.nn as nn

class PadlessEncoder(nn.Module):
    def __init__(self, latent_dim: int, input_dim: int):
        super().__init__()
        self._latent_dim = latent_dim
        self._input_dim = input_dim

        self.main = nn.Sequential(
            nn.Linear(input_dim, 512),
            nn.LeakyReLU(inplace=True),
            nn.Linear(512, 256),
            nn.LeakyReLU(inplace=True),
            nn.Linear(256, 128),
            nn.LeakyReLU(inplace=True),
            nn.Linear(128, 64),
            nn.LeakyReLU(inplace=True),
            nn.Linear(64, latent_dim, bias=True)
        )
        init_layers(self._modules)

```

```
    def forward(self, x):
        return self.main(x)


class PadlessGaussianConvEncoder(PadlessEncoder):
    def __init__(self, latent_dim, input_dim):
        super().__init__(latent_dim * 2, input_dim)
        # override value of _latent_dim
        self._latent_dim = latent_dim

    def forward(self, x):
        mu_logvar = self.main(x)
        mu = mu_logvar[:, :self._latent_dim]
        logvar = mu_logvar[:, self._latent_dim:]
        # Need to clip: loss function depends on the direct value of
   logvar; if var is 0 then logvar could become -inf
        return mu, torch.clip(logvar, min=-20.0, max=20.0)
```

Listing C.1: Encoder architecture for the Taiwan credit score dataset [18]

```
import torch.nn as nn

class SimpleConvDecoder(nn.Module):
    def __init__(self, latent_dim: int, input_dim: int):
        super().__init__()
        self.latent_dim = latent_dim
        self.input_dim = input_dim

        self.main = nn.Sequential(
            nn.Linear(latent_dim, 64, bias=True),
            nn.LeakyReLU(inplace=True),
            nn.Linear(64, 128),
            nn.LeakyReLU(inplace=True),
            nn.Linear(128, 256),
            nn.LeakyReLU(inplace=True),
```

```
            nn.Linear(256, 512),
            nn.LeakyReLU(inplace=True),
            nn.Linear(512, input_dim),
            nn.LeakyReLU(inplace=True),
            nn.Sigmoid()
        )
        init_layers(self._modules)

    def forward(self, x):
        return self.main(x)
```

Listing C.2: Decoder architecture for the Taiwan credit score dataset [18]

Note that the β-TCVAE encoder and decoder classes for the Taiwan credit score dataset are similar to the code in [1], except that it uses linear layers instead of 2D convolutional layers. This is because tabular data is actually single-dimension: each data point can be thought of as a list of values.

Another important note: the class names (`PadlessEncoder`, `PadlessGaussianConvEncoder`, and `SimpleConvDecoder`) are exactly the same as the class names in [1]. This is fine, because in our MSVAE library, encoders and decoders are custom defined within their individual Jupyter notebooks, not within library files.

```
import numpy as np
import torch
from vae import VAE

def _generate_training_sample(
    vae_model: VAE,
    Y_all: torch.Tensor,
    global_variances: np.ndarray,
    device: str,
):
    encoder, decoder = vae_model.encoder, vae_model.decoder

    # fix one latent factor
```

```
    num_samp, latent_dim = Y_all.shape # make sure that it has 2
    dimensions

    Y_all_samp = Y_all.detach().cpu().clone()
    latent_idx = np.random.randint(latent_dim)

    Y_all_samp[:, latent_idx] = Y_all_samp[np.random.randint(num_samp),
    latent_idx]
    Y_all_samp = Y_all_samp.to(device)

    observations = decoder(Y_all_samp).to(device)

    # We sample from the encoded versions of our factors.
    representations = vae_model(observations)["z"].cpu().detach().numpy()

    local_variances = np.var(representations, axis=0, ddof=1)

    argmin = np.argmin(local_variances / global_variances)

    return latent_idx, argmin
```

Listing C.3: Generate training samples in FactorVAE metric

Bibliography

[1] Amir H. Abdi, Purang Abolmaesumi, and Sidney Fels. Variational learning with disentanglement-pytorch. *arXiv preprint arXiv:1912.05184*, 2019.

[2] Cammarata L. Radhakrishnan A. et al. Belyaeva, A. Causal network models of SARS-CoV-2 expression and aging to identify candidates for drug repurposing. *Nature Communications*, 2021.

[3] Ricky T. Q. Chen, Xuechen Li, Roger Grosse, and David Duvenaud. Isolating sources of disentanglement in variational autoencoders, 2018.

[4] Li Deng. The mnist database of handwritten digit images for machine learning research. *IEEE Signal Processing Magazine*, 29(6):141–142, 2012.

[5] Ian J. Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial networks, 2014.

[6] David Heckerman. A tutorial on learning with bayesian networks. *CoRR*, abs/2002.00269, 2020.

[7] Hyunjik Kim and Andriy Mnih. Disentangling by factorising, 2018.

[8] Diederik P. Kingma and Max Welling. An introduction to variational autoencoders. *CoRR*, abs/1906.02691, 2019.

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

[10] Francesco Locatello, Stefan Bauer, Mario Lucic, Gunnar Raetsch, Sylvain Gelly, Bernhard Schölkopf, and Olivier Bachem. Challenging common assumptions in the unsupervised learning of disentangled representations. In *International Conference on Machine Learning*, pages 4114–4124, 2019.

[11] Mohammad Lotfollahi, Anna Klimovskaia Susmelj, Carlo De Donno, Yuge Ji, Ignacio L. Ibarra, F. Alexander Wolf, Nafissa Yakubova, Fabian J. Theis, and David Lopez-Paz.

Learning interpretable cellular responses to complex perturbations in high-throughput screens. *bioRxiv*, 2021.

[12] Frank J. Massey. The kolmogorov-smirnov test for goodness of fit. *Journal of the American Statistical Association*, 46(253):68–78, 1951.

[13] N. Patki, R. Wedge, and K. Veeramachaneni. The synthetic data vault. In *2016 IEEE International Conference on Data Science and Advanced Analytics (DSAA)*, pages 399–410, Oct 2016.

[14] Alec Radford, Luke Metz, and Soumith Chintala. Unsupervised representation learning with deep convolutional generative adversarial networks. 2015.

[15] Jürgen Schmidhuber. Deep learning in neural networks: An overview. *CoRR*, abs/1404.7828, 2014.

[16] Akash Srivastava, Yamini Bansal, Yukun Ding, Cole Hurwitz, Kai Xu, Bernhard Egger, Prasanna Sattigeri, Josh Tenenbaum, David D. Cox, and Dan Gutfreund. Improving the reconstruction of disentangled representation learners via multi-stage modelling. 2020.

[17] Aravind Subramanian, Rajiv Narayan, and Corsello et al. A next generation connectivity map: L1000 platform and the first 1,000,000 profiles. *bioRxiv*, 2017.

[18] I-Cheng Yeh and Che-Hui Lien. The comparisons of data mining techniques for the predictive accuracy of probability of default of credit card clients. *Expert Systems with Applications*, 36(2, Part 1):2473–2480, 2009.