Modeling Object Co-Occurrences in Images

Zeping Li

Master of Science

School of Informatics

University of Edinburgh

2023

Abstract

Generative models provide an elegant framework to efficiently represent high-dimensional data. However, previous studies of generative modeling mainly focus on continuous data or restricted discrete data. In this thesis, I test 5 generative models and their variants on modeling sparse object count data in natural scenes: Variational autoencoder, mixture model, sequence-to-set variational autoencoding transformer, discrete autoregressive flow and discrete autoregressive diffusion. I find that the sequence-to-set variational autoencoding transformer performs the best in terms of likelihood, while the discrete autoregressive diffusion generates samples which are closest to the observed data. Additionally, the negative binomial distribution and the categorical distribution are good choices for explicitly encoding the counts. I also observe that incorporating expressive priors does not necessarily improve the performances of the variational models[1].

[1]The experimental codes and dataset are available at https://github.com/RL-LBAM/ob-co

Research Ethics Approval

This project was planned in accordance with the Informatics Research Ethics policy. It did not involve any aspects that required approval from the Informatics Research Ethics committee.

Declaration

I declare that this thesis was composed by myself, that the work contained herein is my own except where explicitly stated otherwise in the text, and that this work has not been submitted for any other degree or professional qualification except as specified.

(*Zeping Li*)

Acknowledgements

I thank my supervisor Professor Chris Williams for his guidance and timely feedback on my ideas and drafts. I am also thankful to Riza Rizky for his help in downloading and pre-processing the dataset. Special thanks go to my friends Li Wang and Mingyu Chen for our insightful discussions about generative modeling.

I am grateful to my family for their accompany and supports. Lastly, I acknowledge myself for the courage and efforts to explore the field of probabilistic machine learning.

Table of Contents

Chapter 1

Introduction

From identifying potential risks in the environments based on sensory inputs to timely evaluating and updating the values of actions during interactions with other agents, efficiently representing high-dimensional data is a crucial task for real-world agents. Generative models [50] provide an elegant probabilistic framework to address this task, which usually assume that the observed data are generated from a latent distribution which is lower-dimensional or equal-dimensional but more tractable[1]. These models not only represent the high-dimensional data efficiently, but also benefit downstream tasks, such as tailored sample generation [24] and missing data imputation [76]. As the most popular machine learning technique currently, neural networks endow generative models with a more complex mapping from the latent variable to the observed data [66]. There have been many successful architectures for deep generative modeling like the variational autoencoder (VAE) [40] and the generative adversarial network (GAN) [21].

However, generative models often assume that the observed data follow a continuous distribution, or employ techniques like uniform dequantization [64] to construct a continuous distribution from a discrete one, ignoring the ubiquitous discreteness in life. Most real-world data, like the number of people waiting in a queue or the population of a species within a district, originate from some discrete events. The measurement precision can also make continuous data factually discrete (for example, if a scale's minimum unit is one kilogram, then any objects weighed using this scale will show masses that are multiples of one kilogram). Even for those generative models designed for discrete data, the data are mainly confined to languages [80] and consumer ratings [44].

[1]There are many other generative models not having this assumption, like the pure autoregressive models. But this thesis mainly focuses on the generative models assuming a latent distribution.

In this thesis, the focus lies in modeling the count data of different object classes in natural scenes using generative models [43]. Count data are a kind of discrete data characterized by a lower bound of 0 and no theoretical upper bounds. A larger value usually signifies a stronger underlying factor (e.g., more people queuing outside a shop indicates a stronger consumer demand). The object co-occurrences data reflect the physical correlations among object classes and the scene information. For example, when both a knife and a plate are observed, it is natural to infer that there is also a fork. Moreover, these object instances are more likely to appear in a kitchen compared to a toilet. Capturing the correlations and the scene information can offer some potential benefits. First, it may enhance our comprehension of the mechanisms of human vision. The knowledge concerning physical correlations among objects can play a role of top-down control when organizing contextual information and detecting objects in visual tasks [55, 65]. Second, incorporating the object co-occurrences information as prior knowledge into algorithms may help machines detect objects in ambiguous scenes, generate reasonable new instances (e.g., a hand with five fingers) and accommodate different data modalities (e.g., co-occurring object classes should have similar semantic embeddings) [26, 37].

1.1 Related Work

Generative modeling for discrete data is mainly restricted to language and consumer ratings data. Language data are the most common discrete data where tokens are encoded as categories. But the generation of languages is typically conditional (e.g., the embeddings of source language are provided in the machine translation task) and purely autoregressive [7] due to the languages' inherent sequential structures. There are some attempts using the VAE with an autoregressive decoder to reconstruct language data [6, 59], but requiring techniques to mitigate the posterior collapse problem (i.e., the inferred latent variable values are not used, which is not problematic if the latent values are not of interest). Non-autoregressive models like the discrete flow [68, 35] and the discrete diffusion [2, 34] are recently applied to language data as well, but they usually perform worse than the pure autoregressive models since language data are naturally sequential. Another non-autoregressive way to model language data is to treat languages as bags of words [49] or to directly model the counts of tokens [80, 31]. For the former, an important application lies in topic modelling [4, 5], where each token is usually assumed to be generated from a hierarchical latent process. While these models are tailored to identify the topics of sequences, they implicitly model the counts of tokens when the sequence lengths are known. For the latter, the task aligns with the task this thesis focuses on, where the explicit encoding of counts is typically required.

Consumer ratings constitute another form of discrete data commonly addressed in generative modeling. They are closer to the count data since the values indicate the strength of some latent factors too (e.g., a consumer should give higher ratings to a commodity if feeling more satisfied), although they usually have clear lower and upper bounds. The VAE has been applied in many recommendation system studies [44, 47], aiming to recover missing ratings. Bayesian non-negative matrix factorization models [9] are used to extract the latent factors of consumers and items [22] in a generative framework as well. There are also some studies [10, 17] using the GAN to implicitly learn the distribution of ratings and generate samples to alleviate the data sparsity challenge.

Additionally, some studies have investigated how to model count data explicitly in a generative way [1, 46, 69], but primarily within the realm of genomics. The relevant count data can be substantially different from the object count data (e.g., with a much higher dimension).

1.2 Thesis Outline

In this thesis, I focus on the count data extracted from the Microsoft Common Objects in Context (COCO) dataset [45]. Each data point is an 80-dimensional vector whose elements are non-negative integers. The data exhibit high sparsity, with over 90% of them containing fewer than 6 non-zero elements, and the counts are usually small values, with a global maximum value of 28. These properties lead me to apply different encoding methods.

Chapter 2 provides an elaborate exposition of the generative models tested in this thesis, including their assumptions about the latent distributions, the optimization objectives and the ways to generate new samples. All of them except the mixture model are parameterized by neural networks. Some of them have variants concerning the forms of the latent distribution, or the encoding of the counts.

Chapter 3 presents how the models are trained and evaluated. It begins with the introduction of the target dataset, including the pre-processing procedure and some summary statistics of the data. Subsequently, it turns to the final structure of each model after hyperparameter tuning. After showing the training schemes of different models, it finally demonstrates the metrics to evaluate the model performances and relevant results, including quantitative tests and some generated samples.

Chapter 4 analyses how the form of the latent distribution and the encoding method impact the model performances. It also discusses how the best models may offer insights to the potential generation process.

Chapter 5 concludes the thesis, encompassing the main findings, the limitations and some future directions.

The Appendix contains the visualization of the raw data, an overview of the hyperparameter values tested and a sanity check I conduct for a model proposed in this thesis.

Chapter 2

Models

This chapter introduces the models tested in this thesis. The first four models assume that the count vectors are generated from a lower-dimensional distribution (referred to as the latent or prior distribution afterwards). They mainly differ in the encoding of the counts and the choice of the latent distribution. The last two models assume that the count vectors are generated from an equal-dimensional tractable distribution (i.e., a factorized categorical distribution; referred to as the base distribution afterwards). They differ in the transformation of the base distribution.

2.1 Variational Autoencoder

This section presents the vanilla VAE [40]. Let $\boldsymbol{x}$ be the observed variable and $\boldsymbol{z}$ be the latent variable. The goal of the VAE is to fit a joint distribution $p(\boldsymbol{x},\boldsymbol{z})$ that maximizes the log-likelihood[1] of the observed data $\boldsymbol{X} = \{\boldsymbol{x}_1, \boldsymbol{x}_2, ..., \boldsymbol{x}_N\}$:

$$\log p(\boldsymbol{X}) = \sum_{i=1}^{N} \log p(\boldsymbol{x}_i) = \sum_{i=1}^{N} \mathbb{E}_{p(\boldsymbol{z}|\boldsymbol{x}_i)} \log \frac{p(\boldsymbol{x}_i, \boldsymbol{z})}{p(\boldsymbol{z}|\boldsymbol{x}_i)} \tag{2.1}$$

The latent, or prior distribution $p(\boldsymbol{z})$ is assumed to be a standard Gaussian in the vanilla VAE, but the conditional distribution $p(\boldsymbol{x}|\boldsymbol{z})$ is parameterized by a neural network to let $p(\boldsymbol{x})$ be sufficiently expressive, which makes the log-likelihood intractable. As a compromise, the VAE maximizes the evidence lower bound of likelihood (ELBO):

$$\mathrm{ELBO}(\boldsymbol{X}) = \sum_{i=1}^{N} \mathbb{E}_{q(\boldsymbol{z}|\boldsymbol{x}_i)} \log \frac{p(\boldsymbol{x}_i, \boldsymbol{z})}{q(\boldsymbol{z}|\boldsymbol{x}_i)} = \sum_{i=1}^{N} [\mathbb{E}_{q(\boldsymbol{z}|\boldsymbol{x}_i)} \log p(\boldsymbol{x}_i|\boldsymbol{z}) - D_{KL}(q(\boldsymbol{z}|\boldsymbol{x}_i)|p(\boldsymbol{z}))] \tag{2.2}$$

[1]The base number is e for all logarithmic functions in this thesis.

where $q(\boldsymbol{z}|\boldsymbol{x}_i)$ is the variational posterior distribution of $\boldsymbol{z}$ given $\boldsymbol{x}_i$. And the second term on the rightmost expression is the Kullback–Leibler (KL) divergence between the variational posterior and the prior. The variational posterior is assumed to be a factorized Gaussian in the vanilla VAE, ensuring that this term can be computed in a closed form. Thus, the VAE needs two networks for the parameterization of $q(\boldsymbol{z}|\boldsymbol{x})$ and $p(\boldsymbol{x}|\boldsymbol{z})$, which are called encoder[2] and decoder respectively. Figure 2.1 is a diagram of the architecture.

A remaining question is how to calculate the expectation of $\log p(\boldsymbol{x}_i|\boldsymbol{z})$ over the variational posterior. It is estimated by Monte Carlo approximation, and a trick called reparameterization [56] is used to ensure a continuous gradient flow. Specifically, for a given data point, the encoder infers the mean $\boldsymbol{h}_1$ and the standard deviation $\boldsymbol{h}_2$ of the Gaussian variational posterior, then a sample $\boldsymbol{\varepsilon}$ is drawn from a standard Gaussian. The affine transformation result $\boldsymbol{h}_1 + \boldsymbol{h}_2 * \boldsymbol{\varepsilon}$ (element-wise multiplication) is equivalent to a sample drawn from the variational posterior, which is used to approximate the expectation (the estimate should be more stable with a larger sample size, but 1 sample is used usually in practice due to the computational cost. Meanwhile, although the expectation of this estimate is the ELBO, the estimate itself can be larger than the true log-likelihood due to stochasticity).

From another perspective, the expectation term measures how likely the observed data point is when using samples from the variational posterior for generation. The divergence term measures the discrepancy between the prior and the variational posterior. So the negative expectation term and the divergence term are also called reconstruction and regularization losses respectively.

As a generative model, the VAE can not only explain the observed data but also generate new samples. Specifically, both $p(\boldsymbol{z})$ and $p(\boldsymbol{x}|\boldsymbol{z})$ (when $\boldsymbol{z}$ is known) should be tractable distributions. We[3] can get samples from $p(\boldsymbol{z})$, which are passed to the decoder to generate the parameter estimates of $p(\boldsymbol{x}|\boldsymbol{z})$. The samples drawn from the conditional distribution are the final generated samples from the VAE.

[2]The term "encode" should be self-evident in this thesis, but the readers should realize it refers to the way to understand the data when appearing in the contexts like "how to encode the counts", but the way to compress the data or infer the variational posterior when appearing in the contexts like "the encoder encodes the inputs".

[3]In this thesis, "I" refers to the author. "we" is a general pronoun of "people" or "researchers in this field".

2.1.1 Conditional Distributions

The decoder does not directly predict the elements of $\boldsymbol{x}$, but the parameter values for the conditional distribution $p(\boldsymbol{x}|\boldsymbol{z})$. It requires an explicit assumption about the conditional distribution. To simplify the model, I assume the conditional distribution is factorized. In other words, the counts of different object classes are conditionally independent once the value of $\boldsymbol{z}$ is specified. The requirement is then reduced to defining a 1-dimensional distribution for count data.

2.1.1.1 Zero-Inflated Poisson Distribution

The Poisson distribution is the most common distribution to model count data. However, the data in this thesis are highly sparse, so I employ its variant: the zero-inflated Poisson (ZIP) distribution [41]. For a count variable x following a ZIP distribution ZIP(α, λ), its probability mass function is:

$$p(x=k) = \begin{cases} \alpha + (1-\alpha)e^{-\lambda}, & k=0 \\ (1-\alpha)\dfrac{\lambda^k e^{-\lambda}}{k!}, & k>0 \end{cases} \tag{2.3}$$

The ZIP distribution can be seen as a mixture of 0 and a Poisson distribution. The parameter α ($0 < \alpha < 1$) is the mixture weight of 0. The parameter λ ($\lambda > 0$) controls the Poisson distribution component. Its mean and variance are larger when λ is larger. The hypothesis behind the ZIP distribution is intuitive: For a given image, there are two reasons for the absence of instances of a particular object class. One is that the object class and the image are incompatible. For example, It should be impossible to see an elephant in a kitchen. The other is the instance number happens to be 0. For example, forks may appear in a kitchen, but the owner put them into a dishwasher, so they do not appear in the kitchen image.

2.1.1.2 Zero-Inflated Negative Binomial Distribution

For the Poisson distribution, the mean and the variance are the same. So it cannot explain over-dispersed data where the variance exceeds the mean. The negative binomial (NB) distribution is an alternative in such situations. From a Bayesian perspective, the NB distribution is the marginal count distribution when the parameter λ in the Poisson distribution follows a Gamma distribution. Similarly, I consider its zero-inflation version [25]. For a count variable x following a zero-inflated negative binomial (ZINB) distribution ZINB(α, β, γ), its probability mass function is:

$$p(x=k)=\begin{cases}\alpha+(1-\alpha)\beta^{\gamma}, & k=0\\ (1-\alpha)\dfrac{(k+\gamma-1)!}{k!(\gamma-1)!}(1-\beta)^{k}\beta^{\gamma}, & k>0\end{cases} \tag{2.4}$$

The parameter α still controls the degree of zero-inflation. The parameters β ($0<\beta<1$) and γ ($\gamma>0$) control the NB distribution component. A larger γ leads to a larger mean and variance. A larger β leads to a smaller ratio between the variance and the mean, which is strictly larger than 1.

2.1.1.3 Categorical Distribution

The supports of both the ZIP and the ZINB distributions are all non-negative integers. However, very large counts are practically impossible for most object classes since the images only record object instances in a confined space. Furthermore, the counts tend to be small (See Section 3.1). One solution is encoding the counts as categories. Specifically, for a given count variable x, if its maximum observed value is L, I assume it follows a categorical distribution containing $L+1$ categories. Its probability mass function is:

$$p(x=k)=\omega_k \tag{2.5}$$

where $\sum_{i=0}^{L}\omega_i=1$ ($0<\omega_0,\omega_1,...,\omega_L<1$). This encoding method avoids putting probability mass to unrealistic values and can adapt to any multi-peaked count distributions (Both the ZIP and the ZINB distributions can only adapt to the count distributions with at most two peaks). However, it assumes the maximum count observed is indeed the maximum value that the variable can be, so it is unable to explain any counts larger than this value. Besides, it requires different parameter numbers to illustrate the conditional count distributions of different object classes since their observed maximum counts are different.

2.2 Variational Autoencoder With Expressive Priors

In theory, any continuous distributions can be generated by an equal-dimensional standard Gaussian through a sufficiently complex function including neural networks [14]. But the optimization techniques and the computational abilities restrict the expressiveness of the transformed distributions from a standard Gaussian. A compromise

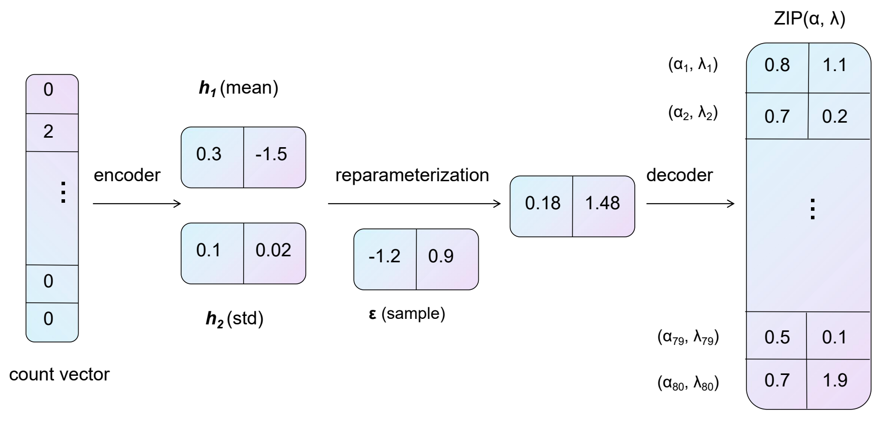


Figure 2.1: A diagram of the vanilla VAE assuming a ZIP conditional distribution

Note: The model assumes a 2-dimensional latent distribution. The calculation of the losses is omitted. All values in the diagram are for demonstration only.

is to directly use more expressive priors, which does improve model performances[4] [67, 11, 51, 23].

2.2.1 Variational Mixture of Posterior Prior

If we revisit equation 2.2 and regard it as a function of $p(\boldsymbol{z})$, the optimal $p(\boldsymbol{z})$ is the aggregated posterior, i.e., $\frac{1}{N}\sum_{i=1}^{N} q(\boldsymbol{z}|\boldsymbol{x}_i)$. This is the idea of the variational mixture of posterior (VAMP) prior [67]. However, directly calculating the aggregated posterior can overfit the data and is computationally costly. So the VAMP prior introduces L trainable pseudo-inputs $\boldsymbol{u}_1$, $\boldsymbol{u}_2$, ..., $\boldsymbol{u}_L$ and adopts their aggregated posterior as the prior.

To some extent, the VAMP prior assumes the data has L clusters, each cluster containing an equal number of data points. As the authors suggest, we can introduce the weights for the pseudo-inputs to make it more flexible. Thus, the final version I adopt is:

$$p(\boldsymbol{x}) = \sum_{i=1}^{L} \omega_i q(\boldsymbol{z}|\boldsymbol{u}_i) \tag{2.6}$$

where $\sum_{i=1}^{L} \omega_i = 1$ $(0 < \omega_1, ..., \omega_L < 1)$.

[4]The prior and the variational posterior restrict each other since the VAE optimizes the ELBO. So there is essentially no difference between using more expressive priors and using more expressive variational posteriors in helping the decoder to approach the theoretical upper bound of performance.

2.2.2 Masked Autoregressive Flow Prior

The normalizing flow [54] is another generative model. Let $\boldsymbol{z}$ be a D-dimensional continuous variable which follows a tractable base distribution like a Gaussian, and f be an invertible function: $\mathbb{R}^D \to \mathbb{R}^D$. For the random variable $\boldsymbol{x} = f(\boldsymbol{z})$, its probability density function is:

$$p(\boldsymbol{x}) = p_{\boldsymbol{z}}(f^{-1}(\boldsymbol{x}))|\det(\frac{d\boldsymbol{z}}{d\boldsymbol{x}})| \tag{2.7}$$

where $\frac{d\boldsymbol{z}}{d\boldsymbol{x}}$ is the inverse of the Jacobian matrix of f and $p_{\boldsymbol{z}}$ refers to the base distribution. Since the computational complexity of calculating the determinant of this matrix is $\mathrm{O}(D^3)$, the normalization flow usually adopts some special f to accelerate the computation, one of which is the autoregressive flow [53, 11].

The autoregressive flow assumes the i^{th} element of $\boldsymbol{x}$ is only transformed from the first i elements of $\boldsymbol{z}$:

$$\boldsymbol{x}^i = \mu_i(\boldsymbol{z}^{1:(i-1)}) + \sigma_i(\boldsymbol{z}^{1:(i-1)}) * \boldsymbol{z}^i \tag{2.8}$$

where μ_i and σ_i are the transformation functions for the i^{th} element, often implemented as neural networks. It ensures the Jacobian matrix is lower-triangular and the computational complexity of the determinant reduces to $\mathrm{O}(D)$.

An autoregressive flow is the most flexible if each transformation function is an independent neural network, but it can cause optimization and computational efficiency issues due to too many parameters. The masked autoregressive flow (MAF) [53] allows different transformation neural networks to share parameters and conduct inference simultaneously by stacking the masked autoencoder for distribution estimation (MADE) [18] layers. Thus, the second prior I test is a trainable distribution transformed by an MAF from a standard Gaussian.

For the flow model, inferring the base variable value is achieved by applying f^{-1} on the observed variable value. Generating samples from the model distribution is achieved by applying f on the samples drawn from the base distribution.

2.2.3 Autoencoder-Assisted Prior

The autoencoder (AE) [30] shares the same network architectures with the VAE, but the former is mainly used for dimension reduction and data denoising. For a given data point, the AE first compresses it to a lower-dimensional embedding using the encoder. The embedding is then passed to the decoder to reconstruct the data point. The AE is trained to minimize only the reconstruction loss.

For the decoder, the embeddings extracted by the AE encoder play the same role as the samples drawn from the variational posterior in the VAE. Actually, the AE is equivalent to a VAE with a Dirac delta variational posterior centered on the embedding value. But in this case, the regularization loss is infinite and non-differentiable for continuous priors. For this “special” VAE, the aggregated posterior is the empirical distribution of the embeddings. To make this distribution continuous, a natural idea is to fit a Gaussian mixture distribution on the extracted embeddings.

Thus, the third prior I test is the autoencoder-assisted (AEA) prior. Specifically, for a given encoder and decoder architecture, I first train an AE and fit a Gaussian mixture distribution with factorized Gaussian components[5] on the extracted embeddings. Then the Gaussian mixture is set as the prior to train a VAE. This idea is inspired by [19], although they regularize the decoder to mimic the influences of a stochastic variational posterior.

2.3 Mixture Model

If we are only interested in modeling and generation, the ultimate goal of training a VAE is to get a decoder that can approximate the observed distribution with the specified prior. But we still introduce the encoder since maximizing the log-likelihood using the decoder and the samples from the prior exclusively is computationally inefficient and unstable. The cost is that the VAE only optimizes the ELBO[6] and extra parameters are introduced. The issue can be mitigated if the latent variable follows a single categorical distribution with a limited number of states. This is the assumption of the mixture model.

When the latent variable has L possible states, the probability mass function of $\boldsymbol{x}$ is:

$$p(\boldsymbol{x}) = \sum_{i=1}^{L} \omega_i p_i(\boldsymbol{x}) \tag{2.9}$$

where $\sum_{i=1}^{L} \omega_i = 1$ $(0 < \omega_1, ..., \omega_L < 1)$ and p_i refers to the i^{th} conditional distribution, which can be a factorized ZIP, ZINB or categorical distribution. The posterior distribution of the latent variable has a closed form, so the log-likelihood can be directly optimized by the expectation-maximization algorithm [12] or the gradient descent.

[5]I have also tried the full Gaussian components, but the regularization loss diverges in such cases. Probably because we need to calculate the inverse of the full-rank covariance matrix for the probability density function of each Gaussian component, making the optimization unstable.

[6]Although some techniques, such as the importance sampling [8], can sufficiently tighten the bound.

Although the VAE with a categorical latent distribution is equivalent to the mixture model, there are still some VAE studies [71, 38] assuming a categorical latent distribution but optimizing the ELBO. In these studies, the latent variable is not illustrated by a single categorical distribution, but a combination of multiple categorical distributions. The distributed representation allows the latent variable to have even millions of states, restricting the direct optimization of the log-likelihood.

2.4 Sequence-to-Set Variational Autoencoding Transformer

No matter which conditional distribution is used, the widespread 0s are either encoded as a special value or an explicit category. But from an informatic-theoretic perspective, we can only encode the non-zero elements in the vectors without losing any information. For example, if an image only contains 2 chairs and 1 table, it can be encoded as [2 chairs, 1 table], rather than [2 chairs, 1 table, 0 fork, 0 elephant, ...]. To further simplify the representation, it can be reduced to a set of objects [chair, chair, table][7].

Since the set size varies across vectors, an intuitive idea is to model the set as a sequence. To convert sequence predictions to set predictions, we have to sum the probabilities over all possible sequences for a given set. For example, the set [chair, chair, table] corresponds to three possible sequences: "chair-chair-table", "chair-table-chair" and "table-chair-chair". Suppose a vector contains m non-zero elements, whose values are N_1, N_2, ..., N_m. The number of possible sequences for the transformed set is $\frac{(\sum_{i=1}^{m} N_i)!}{\prod_{i=1}^{m} N_i!}$. Let S be the set variable, $L(S)$ be the number of its possible sequences, l_1,l_2,..., $l_{L(S)}$ be the sequences induced the set, p_l be the probability mass function of the sequence variable under the model. The probability mass function of S is:

$$p(S) = \sum_{i=1}^{L(S)} p_l(l_i) = L(S) * \frac{1}{L(S)} \sum_{i=1}^{L(S)} p_l(l_i) = L(S)\mathbb{E}_{q(i)} p_l(l_i) \tag{2.10}$$

where q refers to a uniform distribution over all possible sequences. In other words, we only need to calculate the mean probability value of some sequences, and multiply it by the number of possible sequences to approximate $p(S)$.

As mentioned in Section 2.3, an encoder can accelerate training and enhance the result stability. So I choose the encoder-decoder transformer [73] as the sequence

[7]Strictly speaking, a set should not contain duplicate instances, but it can be understood as [chair-1, chair-2, table].

generation model. The traditional encoder-decoder transformer is like the AE, where the encoder extracts a set of embeddings from the input (which does not need to be the same as the target), then the embeddings are passed to the decoder to generate the output autoregressively.

Each encoder layer contains two sub-layers: The first is a multi-head attention sub-layer, and the second is a feed-forward neural network sub-layer whose input and output layers have the same width. The attention function operates on three matrices: the query matrix Q, the key matrix K and the value matrix V. Assuming their sizes are (d_Q,d_K), (d_V,d_K) and (d_V,d_E) respectively, the scaled dot-product attention function is:

$$\text{Attention}(Q,K,V) = \text{softmax}(\frac{QK^T}{\sqrt{d_K}})V \tag{2.11}$$

The size of the output matrix is (d_Q,d_E). For a multi-head attention function with L heads, it introduces L sets of transformation matrices: [W_i^Q, W_i^K, W_i^V] ($i = 1,2,...L$), whose sizes are (d_K, d_h), (d_K, d_h) and (d_E,d_{hE}) respectively. There is also a transformation matrix W^O, whose size is ($d_{hE} * L$, d_M). The function is:

$$\text{MultiHead}(Q,K,V) = \text{Concat}(\text{head}_1,\text{head}_2,...,\text{head}_L)W^O \tag{2.12}$$

where head$_i$ is Attention(QW_i^Q, KW_i^K, VW_i^V). The size of the output matrix is (d_Q, d_M). For the multi-head attention sub-layers in the encoder, all Q, K and V are the stacks of the token embeddings. To clarify, if there are n tokens in an input sequence and the embedding dimension is D, then $d_Q = d_V = n$ and $d_K = d_E = D$. If the head number is L, then $d_{hE} = D/L$ and $d_M = D$. This is also called self-attention. The output of the self-attention sub-layer is unpacked into n D-dimensional vectors. Each of them is then passed to the feedforward neural network sub-layer to get the "new" embedding of each token. Between the two sub-layers and after the feedforward neural network sub-layer, there are residual connection [27] and layer normalization [3] applied (there are also dropout layers [61] within and between the sub-layers, but omitted here). The input and the output of an encoder layer are both a set of token embeddings.

Each decoder layer has three sub-layers. The first is a multi-head self-attention sub-layer to process decoder input (i.e., the ground-truth tokens during training, and the tokens having been generated during generation). The second is a multi-head attention sub-layer whose matrix Q is the stacks of the decoder input's embeddings, while the matrices K and V are the stacks of the final extracted embeddings from the encoder. This is also called cross-attention. The third is still a feedforward neural network sub-layer. Similar to the encoder layer, there are residual connection and layer normalization applied between the sub-layers and after the final sub-layer. The input and the output of a decoder layer are both a set of token embeddings, but the number of tokens is equal to that of the decoder input.

In a sequence-to-sequence generation task, the encoder maps the source sequence to embeddings. The embeddings are transformed through several encoder layers. When predicting the i^{th} token of the target sequence, the decoder maps the first $i-1$ tokens (again, there are given/generated during training/generation) to another set of embeddings. The embeddings are transformed through several decoder layers. The final last embedding (or the pooled final embedding) is passed to a feedforward neural network to get the corresponding prediction.

Nevertheless, to make it a full generative model, the final embeddings extracted by the encoder should be treated as the parameter estimates of the variational posterior. It causes problems since the input sequence length is not fixed and using samples from multiple variational posteriors is unstable. To solve this, I introduce a multi-head attention layer to pool the embeddings inspired by the set transformer [42]. For this layer, the Q is a matrix of size $(2, D)$, whose elements are trainable. The K and V are both the stacks of extracted final embeddings. Its output is unpacked into two D-dimensional vectors, representing the mean and the (log) standard deviation of the variational posterior. The decoder can only attend to a sample from the variational posterior for the cross-attention [74]. Figure 2.2 is a diagram of the architecture.

For a given set, the sequence-to-set variational autoencoding transformer (SSVAT) processes it as follows: The set is first transformed to some possible sequences. For each sequence, the encoder extracts the embedding of each token. The embeddings are pooled through the multi-head attention layer and become two vectors, representing the mean and the (log) standard deviation of the Gaussian variational posterior. A sample drawn from the variational posterior is passed to the decoder for the cross-attention. Then we can calculate the reconstruction and the regularization losses and transform them to the ELBO. The exponentiated ELBO is a surrogate of the sequence probability. The mean of all sequence probabilities, multiplied by the number of possible sequences, is the final estimate of the set probability.

Although I model the set as a sequence, the order is meaningless. So no position embeddings are added for both the encoder and the decoder. In such cases, the encoder is permutation-invariant due to its architecture, which is an extra benefit.

To generate a sample from the model distribution using the SSVAT, we should pass the decoder a sample drawn from the prior. The generated sequence is then transformed

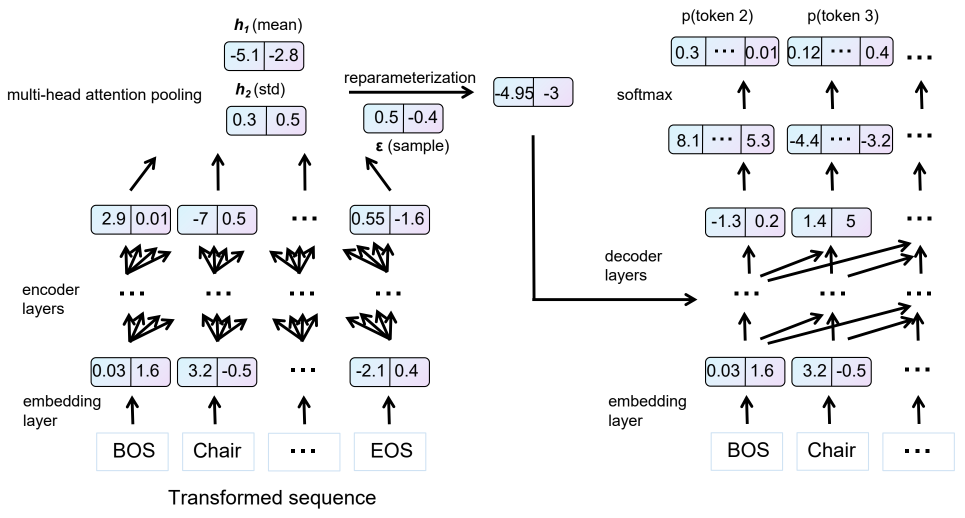


Figure 2.2: A diagram of the SSVAT

Note: The model assumes a 2-dimensional latent distribution. All values in the diagram are for demonstration only. The calculation of the losses and the transformation between the sequence and the set are omitted. The arrows indicating the attention are simplified in the encoder layers to avoid a mess. This diagram illustrates the teacher-forcing training process [77], where the ground-truth first $i-1$ tokens are provided when predicting the i^{th} ($i = 2, 3, ...$) token. During generation, only a sample from the prior and the [BOS] (beginning of sequence) token are provided initially for the decoder. After predicting the i^{th} token's conditional distribution, a sample from it is added to the current sequence. The decoder will use the sequence to predict the $(i+1)^{\text{th}}$ token's conditional distribution until the [EOS] (end of sequence) token is generated.

to the count vector. I also test two expressive priors for the SSVAT: the MAF prior and the AEA prior. I do not test the VAMP prior since the inputs are sequences whose lengths are not fixed, which makes it difficult to set the trainable inputs.

To my best knowledge, the only study which generates sets of discrete data using sequence models is [63]. However, they do not directly optimize and evaluate the set probabilities, and they do not use the transformer encoder and the variational posterior to accelerate learning. To ensure the SSVAT can produce reasonable probability estimates for the count data, I also conduct a sanity check using the samples generated from some mixture model variants. See Appendix B for the procedures and the results.

2.5 Discrete Autoregressive Flow

As mentioned in Section 2.2.2, the normalizing flow is another way to represent high-dimensional distributions. But the studies of the discrete flow are still in infancy since the change of volume (i.e., the Jacobian matrix determinant) is not applicable. There have been some attempts [68, 35, 36], but a systematic comparison is lacking. [2] compares the flows in [68] and [35], revealing that the former performs better in modeling language data. Thus, I choose the winner, the discrete autoregressive flow (DAF) to be tested on the count data.

For the DAF with an invertible function f and a base distribution $p_{\boldsymbol{z}}$, the probability mass function of the random variable $\boldsymbol{x} = f(\boldsymbol{z})$ is:

$$p(\boldsymbol{x}) = p_{\boldsymbol{z}}(f^{-1}(\boldsymbol{x})) \tag{2.13}$$

The DAF encodes each element of the base and the observed variables as a categorical variable, and applies the modulo location-scale transformation:

$$\boldsymbol{x}^i = [\mu_i(\boldsymbol{z}^{1:(i-1)}) + \sigma_i(\boldsymbol{z}^{1:(i-1)}) * \boldsymbol{z}^i] \; mod \; L(\boldsymbol{x}^i) \tag{2.14}$$

where $L(\boldsymbol{x}^i)$ is the number of possible categories of that element, i.e., the maximum observed count of the object class plus 1 (see Section 2.1.1.3).

Different from equation 2.8, the outputs of $\mu_i(\boldsymbol{z}^{1:(i-1)})$ and $\sigma_i(\boldsymbol{z}^{1:(i-1)})$ are both integers representing the categories to ensure all elements are integers. It is achieved by adding an *argmax* function after the neural network outputs. The gradient is estimated using a $softmax$ straight-through gradient estimator [48]. This whole transformation is invertible only when $L(\boldsymbol{x}^i)$ and $\sigma_i(\boldsymbol{z}^{1:(i-1)})$ are coprime. To ensure it, I set the σ transformation to 1 and use an MAF to construct the μ transformation. The base distribution is a factorized categorical distribution.

2.6 Discrete Autoregressive Diffusion

The diffusion model is a popular generative model recently [57, 32, 60]. Its core idea is to gradually inject noise into the observed distribution to make it the base distribution, and train a model to gradually recover the observed distribution from the base distribution. The joint distribution of the states during the noise injection process is a pre-specified variational posterior. The model is trained to learn a conditional joint distribution of states during the denoising process to maximize the ELBO with the base distribution as the prior. Similar to the flow model, the discrete diffusion model is still under-explored and is currently restricted to the categorical variables [35, 2, 34]. Among the existing approaches, the discrete autoregressive diffusion (DAD) [34] is a competitive participant due to its computational efficiency in both training and sample generation.

The DAD takes a form of an order-agnostic autoregression model, but the authors have proved that it is equivalent to an infinite time limit absorbing diffusion, with the base distribution a Dirac delta distribution where all elements are a [MASK] category. The DAD assumes that the observed variable can be generated in any order autoregressively. For a D-dimensional variable, there are $D!$ possible orders. Let l_1, l_2, ..., $l_{D!}$ be the possible orders, $l_i(j)$ be the j^{th} element in the i^{th} order, the log-likelihood of $\boldsymbol{x}$ is:

$$\begin{aligned}\log\ p(\boldsymbol{x}) &= \log\ \frac{1}{D!}\sum_{i=1}^{D!} p(\boldsymbol{x}|l_i) > \mathbb{E}_{q_1(i)} \sum_{j=1}^{D} log\ p(\boldsymbol{x}^{l_i(j)}|\boldsymbol{x}^{l_i(<j)}) \\ &= \mathbb{E}_{q_1(i)} D * \mathbb{E}_{q_2(j)} \log\ p(\boldsymbol{x}^{l_i(j)}|\boldsymbol{x}^{l_i(<j)}) \\ &= D\mathbb{E}_{q_1(i)}\mathbb{E}_{q_2(j)} \frac{1}{D-j+1} \sum_{k\in l(\geq j)} \log\ p(\boldsymbol{x}^{l_i(k)}|\boldsymbol{x}^{l_i(<j)})\end{aligned} \tag{2.15}$$

where q_1 refers to a uniform distribution over all possible orders, and q_2 refers to a uniform distribution over all possible element positions.

Thus, the DAD is also optimizing the ELBO due to the first inequation. During the training, we need to uniformly select an order and a position, and use elements prior to that position to predict the elements after the position simultaneously. The prediction is conducted by a neural network. All elements are encoded as categorical variables and the masked state is encoded as a additional category [MASK]. To generate a sample, we need to uniformly select an order and generate the elements autoregressively.

Besides the 5 models mentioned above, I test a baseline model which assumes a factorized ZIP, ZINB or categorical distribution. And as an ablation study, for all models which should consider the conditional distribution, I test two extra distributions: the Poisson and the NB distributions, whose probability mass functions can be represented by the equations 2.3 and 2.4 when the parameter α is 0 respectively.

Chapter 3

Experiments

The chapter delves into the experiment details. Section 3.1 introduces the pre-processing and some summary statistics of the data for modeling. Section 3.2 describes the final architecture of each model. Section 3.3 illustrates the training schemes of different models. Section 3.4 introduces the metrics to evaluate the model performances. Section 3.5 shows the model performances and some generated samples.

3.1 Dataset

The object count data is extracted from the COCO dataset [45], a dataset mainly for object detection, segmentation and captioning. I use the training set of COCO 2017, which considers 80 classes of objects in 118,287 images. Each image corresponds to an 80-dimensional count vector, whose elements are non-negative integers. The object classes are: person, bicycle, car, motorcycle, airplane, bus, train, truck, boat, traffic light, fire hydrant, stop sign, parking meter, bench, bird, cat, dog, horse, sheep, cow, elephant, bear, zebra, giraffe, backpack, umbrella, handbag, tie, suitcase, frisbee, skis, snowboard, sports ball, kite, baseball bat, baseball glove, skateboard, surfboard, tennis racket, bottle, wine glass, cup, fork, knife, spoon, bowl, banana, apple, sandwich, orange, broccoli, carrot, hot dog, pizza, donut, cake, chair, couch, potted plant, bed, dining table, toilet, TV, laptop, mouse, remote, keyboard, cell phone, microwave, oven, toaster, sink, refrigerator, book, clock, vase, scissors, teddy bear, hair drier, toothbrush.

Since the count vectors are very sparse (1,021 of them are even all-zero vectors), I order them by the number of non-zero elements (i.e., how many object classes appearing in the corresponding image) and the sum of elements (i.e., how many object instances appearing in the corresponding image), and choose the first 110,000 vectors.

After the selection, there are still 90.2% of vectors containing fewer than 6 non-zero elements. Besides, the element values are generally small. The sum of elements is less than 11 for 74.5% of the vectors, indicating that each object class usually has no or few instances in one image (see Appendix A for the empirical count distributions of different object classes). The following is an example count vector: [0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 4 0 1 0 3 2 1 1 0 0 3 0 0 0 0 0 0 5 0 0 0 1 0 1 1 0 0 0 0 1 1 0 0 1 1 0 2 0 0 0 0]

The count vectors are shuffled and split into the training, the validation and the test sets, which contains 80,000, 10,000 and 20,000 vectors respectively.

3.2 Model Architectures

The hyperparameter tuning procedures and some patterns can be found in Appendix C.

For the VAEs, the architecture of the encoder is [80, 160, 160, 4*2] if using the ZIP conditional distribution, where *2 indicates that it has two output layers corresponding to the mean and the standard deviation of the variational posterior. The architecture of their decoder is [4, 160, 160, 80*2], where *2 indicates that it has two output layers corresponding to the estimates of the conditional distribution's two parameters. When using the ZINB conditional distribution, their encoder architecture is [80, 80, 8*2] and their decoder architecture is [8, 80, 80*3]. When using the categorical conditional distribution, their encoder architecture is [80, 160, 8*2] and their decoder architecture is [8, 160, max-counts]. Here max-counts is an 80-dimensional vector where the i^{th} ($i = 1, 2, ..., 80$) element is the observed maximum count of the i^{th} object class plus 1. It indicates the decoder has 80 output layers, whose widths are dependent on the observed maximum counts of different object classes. When using the Poisson/NB distribution, the encoder/decoder architecture is the same as that when using its zero-inflated version (although the number of the output layers is reduced by 1 for the decoder). This is the same for other hyperparameters below.

For the VAEs with expressive priors, the number of pseudo-inputs is 8 for all variants using the VAMP prior. When using the MAF prior, the numbers of MADE layers are 5, 3 and 3 for the VAE-ZIP, VAE-ZINB and VAE-categorical models respectively. A MADE layer contains two hidden layers whose widths are five times the input layer width. It has two output layers corresponding to the outputs of the μ and the σ transformations respectively. When using the AEA prior, the number of mixture components is 2 for all variants.

For the mixture model, the numbers of mixture components are 900, 900 and 800 respectively for the mixture-ZIP, the mixture-ZINB and the mixture-categorical models respectively. Since the numbers are large, some components may only have small weights. I calculate the effective number of components by:

$$N = \exp^{(\sum_{i=1}^{L} -\omega_i \log \omega_i)} \tag{3.1}$$

where ω_i $(i = 1, 2, ..., L)$ is the weight of the i^{th} component of a mixture model with L components. The entropy of the mixture weight distribution is equal to that of a uniform distribution over N categories. The effective numbers of mixture components are 189.623, 92.276, 7.258, 62.294 and 52.454 for the mixture-ZIP, the mixture-ZINB, the mixture-categorical, the mixture-Poisson and the mixture-NB models respectively. Only the value from the mixture-categorical model indicates that the data may be explained by a limited number of components. But it is not the best mixture model variant, even not a competitive member among all models (see Section 3.5.1). Collectively, these results suggest the distribution of the count vectors is complex enough that cannot be captured by a small set of components/clusters[1].

For the SSVATs, the embedding dimension is 20, and they have 3 encoder/decoder layers. The dropout rate and the number of heads for the encoder/decoder layers are 0.1 and 4 respectively. The feedforward neural network sub-layers have 1 hidden layer, whose width is 40. When using the MAF prior, the number of MADE layers is 5. When using the AEA prior, the number of mixture components is 2.

For the DAF, it has 4 MADE layers. Each MADE layer has 3 hidden layers, whose widths are twice the input layer width.

For the DAD, It has 2 hidden layers, whose widths are the twice the input layer width.

For both the DAF MADE layers and the DAD, the input is a concatenation of count embeddings of different object classes. They output layers are the same as those of the VAE-categorical models. Besides, the input layer width of the DAD is larger than the sum of the output layer widths since it needs to consider the extra [MASK] category.

3.3 Implementation Details

For each model, I run 5 experiments. Each experiment is controlled by a random seed. The 5 seeds are generated by a primary random seed, which is the same for all models.

[1]I have also tried to reduce the variational posterior means extracted by the VAEs to 2-dimensional using t-SNE [72], but do not observe clear clusters visually either.

In each experiment, for the VAEs, the DAF and the DAD, the model is trained with a batch size of 100. The weight parameters are initialized using the Xavier uniform method [20] while the bias parameters are initialized with 0. I use ReLU as the activation function [16]. The negative ELBO as the loss, is optimized using the Adam optimizer [39]. The initial learning rate is 0.005. The gradients are clipped when their norms are larger than 4. After each epoch, the mean validation loss is calculated. The learning rate is halved when the mean validation loss has not been improved for 5 consecutive epochs. The experiment is stopped early when the mean validation loss has not been improved for 10 consecutive epochs. The maximum epoch is 50.

To maintain a healthy gradient flow, the gradients are replaced with the noise from a standard Gaussian if NaN value appears. The α and the β parameters in the ZIP and the ZINB/NB distributions are restricted between 0.01 and 0.99 (i.e., the values which are beyond the threshold are replaced with the threshold value. So the gradient flow through the corresponding parameter is truncated then. Same below). The λ and the γ parameters in the ZIP/Poisson and the ZINB/NB distributions are restricted between 0 and 50. For the categorical distribution, I implement the label smoothing technique [62] during training to ensure the model assigns some probability mass to the unseen but possible categories. Specifically, the one-hot encoded labels are replaced with a mixture of themselves and a uniform distribution over all possible labels (but the number of possible labels still varies across object classes). The mixture weight is 10^{-5} for the uniform distribution.

One sample is used in the reparameterization trick. The sample is also used to approximate the regularization loss if the KL divergence does not have a closed form. Specifically, the divergence term in equation 2.2 can be written as:

$$D_{KL}(q(\boldsymbol{z}|\boldsymbol{x}_i)|p(\boldsymbol{z})) = \mathbb{E}_{q(\boldsymbol{z}|\boldsymbol{x}_i)} \log q(\boldsymbol{z}|\boldsymbol{x}_i) - \mathbb{E}_{q(\boldsymbol{z}|\boldsymbol{x}_i)} \log p(\boldsymbol{z}) \tag{3.2}$$

where the former term has a closed form since the variational posterior is a factorized Gaussian, and the latter can be approximated by sampling as long as the probability density of the given sample can be computed.

The best mean validation loss is the final model performance in that experiment. The experiment with the best final model performance is chosen. I use the parameter values in the checkpoint which produce the best mean validation loss in the chosen experiment as the final parameter estimates of the model.

For the mixture model, the best architectures require hundreds of mixture components, making the training process slow. So I increase the batch size to 4000 and the initial learning rate to 0.05. Besides, the loss is the negative log-likelihood (same for the baseline and the DAF).

For the SSVATs, each vector is first transformed into a sequence randomly. This sequence is called the forward sequence. I then reverse this sequence to create the backward sequence. After adding the [BOS] and [EOS] symbols, the sequences are passed to the model. Their mean exponential ELBO, multiplied by the number of possible sequences serves as the estimate of the corresponding set probability. This approach is intended to mitigate variance. The model are not allowed to predict/generate the [BOS] token and the token for padding by setting their logit values to the negative infinity.

For the VAE/SSVAT with AEA prior, the training process of the AE is the same as above. After training, the AE encoder extracts the embeddings of points in the training and the validation sets. The embeddings are used to fit the mixture model with factorized Gaussian components.

For the baseline model, the factorized ZIP [13], Poisson and categorical distributions have closed-form maximum likelihood estimates, which are directly used. The log-likelihood of the factorized ZINB and NB models is optimized by the gradient descent, with a fixed learning rate of 0.2, a norm threshold of 10 and a maximum epoch of 1000, which are the empirical values ensuring convergence.

3.4 Evaluation Metrics

3.4.1 Likelihood

The main metric I use is the mean loss on the test set, i.e., the negative log-likelihood or ELBO. Although the ELBO is a lower bound of the log-likelihood, if a model's ELBO is significantly larger than another model's log-likelihood, it sufficiently supports the former model is better. Also, I conduct paired-sample t tests to compare the mean test losses since the test set is fixed.

Besides, an important fact is I am using the test set distribution as a surrogate of the ground-truth distribution that generates the data. Since the data is discrete, the mean test loss is actually the cross entropy (although the base number is e rather than 2) between the model distribution and the test set data's empirical distribution. The lower bound of the mean test loss is the entropy of the test set data's empirical entropy, which is 8.652. Knowing this value can help us to better understand how the models perform.

3.4.2 Sample Quality

The ELBO is a surrogate of the log-likelihood, but it is also influenced by the compatibility between the true and the variational posterior. It is possible that a model with a lower ELBO actually has a larger log-likelihood when comparing two models using the ELBO. Meanwhile, all models tested are generative models. It is a natural idea to compare the quality of their generated samples. Theoretically, if a model can better approximate the observed data distribution, the samples it generates should be more similar to the observed data.

However, the sparse count data are very different from language or image data, traditional pre-trained-classifier-based metrics like the Fréchet inception distance [29] may be biased. Meanwhile, there are no commonly used labels for images in the COCO dataset (not to mention the count vectors may be not a good surrogate of the original images), so evaluating the sample quality through downstream tasks [58] is also unrealistic. Inspired by the GAN [21], I introduce a discriminator to distinguish whether the data are generated or observed. Specifically, I use a simple neural network containing 1 hidden layer (whose width is 80, equal to the number of object classes) with ReLU activation as the discriminator.

For each model, I let it generate 90,000 points. They are randomly split into two groups, containing 80,000 and 10,000 points respectively. The final training set is the concatenation of the original training set (positive labels, same below) and the first group (negative labels, same below). The final validation set is the concatenation of the original validation set and the second group. The training scheme of the neural network is the same as that in Section 3.3.

After training the discriminator, it makes predictions for the points in the test set. I consider two metrics to measure the model performances: One is the mean loss of the discriminator (i.e., the negative log-likelihood). The other is the accuracy of the discriminator with a decision threshold of 0.5.

3.5 Results

3.5.1 Model Performances

The mean test losses with standard errors of different models are shown in Table 3.1 (lower is better). The mean test losses with standard errors of the discriminator for different models are shown in Table 3.2 (higher is better). The mean accuracy with standard errors of the discriminator for different models is shown in Table 3.3 (lower is better). The theoretical loss and accuracy should be $\log(0.5) = 0.693$ and 50.0% respectively if the discriminator cannot distinguish the observed and the generated data. So values closer to the two values indicate a higher model performance.

For models having variants assuming different conditional distributions, the best variant is labelled in italics. It is asterisked if it performs significantly better than the second best variant at a significance level of 0.05. The global best model is labelled in bold. It is asterisked if it performs significantly better than the second global best model.

The mean test losses of the best variants (The DAF and the DAD have no variants. The best variants are the baseline-categorical, the VAE-MAF-categorical, the mixture-NB and the SSVAT-MAF respectively for the rest models based on the metrics) of different models, and the corresponding discriminator accuracy (the mean discriminator test losses indicate a similar pattern) is shown in Figure 3.1.

3.5.2 Generated Samples

Here are some samples generated by the best variants of different models. Since the data are sparse, I only show the non-zero counts.

Baseline:

[1 person, 4 giraffe, 3 knife]

[1 truck]

[9 person, 1 car, 1 tie, 5 bottle, 1 spoon]

[1 person, 1 airplane, 1 chair]

VAE:

[1 bottle, 1 knife, 2 pizza, 1 cake, 1 chair, 1 dining table]

[1 person, 1 dog]

[2 zebra]

[1 person, 11 bottle, 1 dining table, 1 sink]

Mixture:

[1 person]

[1 toilet]

[5 car, 1 airplane, 1 truck]

[9 person, 3 bus, 1 fire hydrant, 3 bench, 2 bird, 1 cow, 1 elephant, 2 giraffe, 1 handbag, 1 tie, 6 skis, 3 snowboard, 1 kite, 1 baseball bat, 8 cup, 33 knife, 3 spoon, 1 bowl, 35 sandwich, 1 orange, 6 pizza, 1 donut, 1 cake, 3 potted plant, 15 bed, 1 dining table, 3 laptop, 4 remote, 3 vase, 4 teddy bear, 103 toothbrush]

(I choose this extreme example deliberately to show a potential problem of the NB conditional distribution. See Section 4.1)

SSVAT:

[5 person, 15 tie],

[1 person, 1 handbag, 7 bottle, 4 wine glass, 2 cup],

[4 airplane, 7 truck],

[1 bottle, 1 bowl, 1 refrigerator]

DAF:

[4 person, 1 fire hydrant, 14 book, 1 vase],

[1 bottle, 1 bed, 1 dining table],

[1 person, 1 sandwich, 1 book],

[1 person, 1 train, 1 skateboard, 1 cake, 2 couch, 1 bed]

DAD:

[9 person, 1 frisbee, 1 clock],

[2 person, 1 cow, 2 chair],

[1 person, 1 dog, 1 donut],

[2 donut]

As can be seen, these variants (except the baseline) capture the physical correlations among the object classes more or less. For example, [2 zebra] may correspond to an image of animals in the wild. [1 bottle, 1 bowl, 1 refrigerator] may correspond to an image of cooking utensils in a kitchen. But there are still some vectors hard to understand like [1 person, 1 handbag, 7 bottle, 4 wine glass, 2 cup] (the counts mainly imply that there should be several people, but the count of person is 1) and extreme like the final one generated by the mixture model.

Conditional Distribution	**Baseline**	**Vanilla VAE**	**VAE-VAMP**	**VAE-MAF**	**VAE-AEA**	**Mixture**
ZIP	17.097(0.066)	13.746(0.050)	13.756(0.050)	13.656(0.050)	15.476(0.050)	14.286(0.053)
ZINB	15.673(0.058)	13.635(0.048)	13.691(0.049)	13.578(0.049)	*14.551(0.049)**	13.962(0.048)
Categorical	*15.459(0.057)**	*13.408(0.049)**	*13.427(0.050)**	*13.320(0.050)**	15.416(0.050)	13.731(0.048)
Poisson	25.300(0.145)	14.445(0.061)	14.393(0.061)	14.239(0.061)	15.662(0.061)	14.299(0.060)
NB	15.683(0.058)	13.642(0.048)	13.677(0.049)	13.615(0.049)	14.675(0.050)	*13.270(0.046)**
	SSVAT	**SSVAT-MAF**	**SSVAT-AEA**	**DAF**	**DAD**	
	13.162(0.052)	**11.979(0.056)***	17.627(0.074)	15.467(0.057)	13.050(0.101)	

Table 3.1: Mean test losses with standard errors of different models

Note: A lower value indicates a better model performance. For models having variants assuming different conditional distributions, the best variant is labeled in italics, which is also asterisked if performing significantly better than the second best variant at a significance level of 0.05. The global best model is bolded, which is also asterisked if performing significantly better than the second best global model.

Conditional Distribution	**Baseline**	**Vanilla VAE**	**VAE-VAMP**	**VAE-MAF**	**VAE-AEA**	**Mixture**
ZIP	0.237 (0.005)	0.513 (0.004)	0.501 (0.004)	0.545 (0.004)	0.443 (0.005)	0.421 (0.006)
ZINB	*0.312 (0.006)*	0.521 (0.004)	0.515 (0.004)	0.534 (0.004)	0.508 (0.004)	0.432 (0.005)
Categorical	0.294 (0.005)	*0.569 (0.004)**	0.540 (0.004)	*0.588** (0.004)	0.514 (0.004)	0.476 (0.004)
Poisson	0.115 (0.004)	0.537 (0.004)	*0.545 (0.004)*	0.555 (0.004)	0.462 (0.004)	0.543 (0.004)
NB	0.311 (0.006)	0.514 (0.005)	0.526 (0.004)	0.541 (0.005)	*0.559** (0.005)	*0.570 (0.004)**
	SSVAT	**SSVAT-MAF**	**SSVAT-AEA**	**DAF**	**DAD**	
	0.596(0.004)	0.568(0.004)	0.528(0.004)	0.313(0.005)	**0.666(0.003)**	

Table 3.2: Mean test losses with standard errors of the discriminator

Note: A higher value indicates a better model performance. For models having variants assuming different conditional distributions, the best variant is labeled in italics, which is also asterisked if performing significantly better than the second best variant at a significance level of 0.05. The global best model is bolded, which is also asterisked if performing significantly better than the second best global model.

Conditional Distribution	**Baseline**	**Vanilla VAE**	**VAE-VAMP**	**VAE-MAF**	**VAE-AEA**	**Mixture**
ZIP	91.3 (0.2)	80.2 (0.3)	79.8 (0.3)	77.7 (0.3)	82.1 (0.3)	84.5 (0.3)
ZINB	88.7 (0.2)	80.0 (0.3)	79.5 (0.3)	78.5 (0.3)	79.8 (0.3)	84.1 (0.3)
Categorical	89.8 (0.2)	77.2 (0.3)	79.8 (0.3)	76.4 (0.3)	81.6 (0.3)	83.0 (0.3)
Poisson	95.7 (0.1)	*76.8 (0.3)*	*74.3 (0.3)**	*74.6 (0.3)**	81.7 (0.3)	75.4 (0.3)
NB	*88.7 (0.2)*	79.8 (0.3)	78.7 (0.3)	77.4 (0.3)	*76.2 (0.3)**	*75.2 (0.3)*
	SSVAT	**SSVAT-MAF**	**SSVAT-AEA**	**DAF**	**DAD**	
	67.5(0.3)	70.9(0.3)	74.8(0.3)	88.8(0.2)	**61.1(0.3)***	

Table 3.3: Mean accuracy with standard errors (%) of the discriminator

Note: A lower value indicates a better model performance. For models having variants assuming different conditional distributions, the best variant is labeled in italics, which is also asterisked if performing significantly better than the second best variant at a significance level of 0.05. The global best model is bolded, which is also asterisked if performing significantly better than the second best global model.

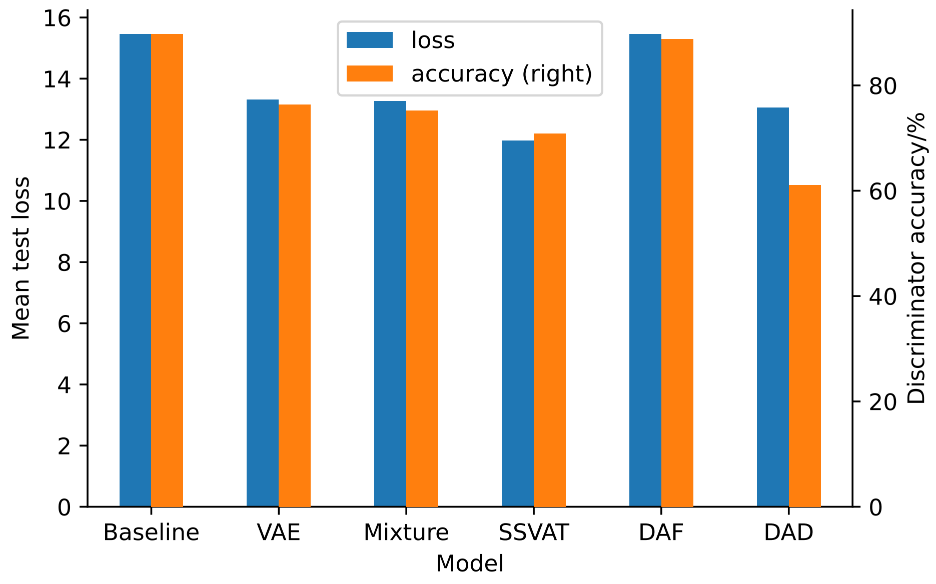


Figure 3.1: Performances of the best variants

Note: For both metrics, a lower value indicates a better model performance.

Chapter 4

Discussions

This chapter discusses the experiment results. Section 4.1 analyses the effects of different conditional distributions. Section 4.2 analyses the effects of different priors. Section 4.3 discusses the best ways to model object co-occurrences.

4.1 Choice of the Conditional Distribution

The following discussion is restricted to the models which need an explicit choice of the conditional distribution.

When the model performances are measured by likelihood as shown in Table 3.1, the categorical distribution holds a dominant advantage across models. As mentioned in Section 2.1.1.3, the categorical distribution avoids putting probability mass to unrealistic values and can flexibly adapt to the multi-peaked distributions. The empirical count distributions of some object classes like sheep and bananas are indeed multi-peaked (see Appendix A), lending some supports to this distribution.

However, among the models, the best one[1] is the mixture model with the NB distribution components. And the NB distribution shows a close, or better performance than the ZINB distribution across models. This is surprising since the former is nested in the latter. After checking the original distributions, I find for most object classes, the most frequent non-zero count is 1, and some of them like books and bananas can have a small peak at a large count. It causes an over-dispersed pattern: most counts are small values (0 or 1), but few counts are considerably larger. This pattern can be explained by the NB distribution when its two parameters are both small [80]. In other words, the NB

[1]The conclusion is not rigorous since the loss is the negative log-likelihood for the baseline, the mixture model and the DAF, but the negative ELBO for others.

distribution is flexible enough to account for the sparsity of the count data. Introducing the zero-inflation parameter may make the loss landscape more multi-modal and let the model get stuck in a worse sub-optimal solution. This can also be the reason why it can perform better than the categorical distribution, although the latter is theoretically the best choice as it can mimic any finite-state discrete distributions.

When the model performances are measured by sample quality as shown in Tables 3.2 and 3.3, the results become less clear. But the categorical and the NB distributions are still the main winners. Interestingly, the Poisson distribution shows an advantage in the VAE-VAMP and the VAE-MAF models. It can be explained since the expressive priors can allow the conditional Poisson distribution to have a low λ mostly, but a large λ occasionally. This can help explain the over-dispersion of data.

Nevertheless, the model can generate some outlier samples like the one shown in Section 3.5.2 if not using a categorical conditional distribution. The vector is very dense (containing many object classes) and some elements are unrealistic (e.g., 103 toothbrush), indicating a strong activation of the latent factors. In other words, coupled with a count conditional distribution, an extreme sample from the prior can lead to an extreme generated count vector. To further investigate it, I calculate the numbers of outlier samples for all models assuming a count conditional distribution[2]. The standard of outlier is for at least one object class, the generated count is larger than twice the maximum observed count. The result is shown in Table 4.1. As can be seen, the models assuming a Poisson/ZIP conditional distribution generally produce fewer outlier samples than those assuming a NB/ZINB conditional distribution, which may be because the Poisson distribution restricts the mean and the variance to be the same. And the zero-inflation assumption generally decreases the number of outlier samples, which is expected since it explicitly removes some probability mass from the non-zero counts. Combined with Tables 3.2 and 3.3, the number of outlier samples and sample quality has a positive correlation, which somewhat explains why the Poisson distribution performs better when measuring the model performances using sample quality.

In conclusion, the zero-inflation seems unnecessary to explain the sparse counts. When we need a count distribution as the conditional distribution, the NB distribution is a good choice. But it can lead to the problem of extreme generated samples. The categorical distribution is generally the best choice if we believe the observed maximum counts do reflect the factual maximum counts, although the extra parameters may also

[2]The term "count distribution" refers to the distributions which encode the counts as non-negative integers rather than categories if not referring the empirical distribution a specific object class's counts.

Conditional Distribution	Baseline	Vanilla VAE	VAE-VAMP	VAE-MAF	VAE-AEA	Mixture
ZIP	0	0	5	3	5246	1194
ZINB	77	85	88	96	281	516
Poisson	0	2	5	26	5673	19
NB	104	182	237	218	356	229

Table 4.1: Numbers of outlier samples out of 90,000 samples for different models

cause some problems during the optimization, and adding a output layer for each object class is computationally costly.

4.2 Choice of the Prior

The following discussion is restricted to the models which can have an expressive prior.

The effects of the expressive priors are consistent across the evaluation metrics and the conditional distributions. That is, the VAMP prior does not have an obvious effect on the model performances, the MAF prior improves the model performances, while the AEA prior harms the model performances.

For the VAMP prior, I check the mixture weights of the components, and find the mixture distribution is always dominated by one component. This component is close to a standard Gaussian. Take the VAE-VAMP-ZIP model as an example, the mean and variance are: [-0.033 -0.024 0.174 -0.059 -0.018 -0.567 0.463 -0.022] and [0.932 0.911 0.262 0.878 0.941 3.181 5.296 0.895]. I have also tried to use the aggregated rather than the mixture of the variational posteriors of the pseudo-inputs as the prior, the result is all components converge to a standard Gaussian.

For the MAF prior, I compare it with a standard Gaussian in a rough way. Specifically, I generate some samples from the prior, reduce them to 2-dimensional using t-SNE [72], and compare them with a 2-dimensional standard Gaussian visually. It seems that their empirical distribution is more dispersed. Figure 4.1 is an example from the SSVAT-MAF model.

As for the inferior performances of the AEA prior, a possible reason is it overfits the data. As mentioned in Section 2.2.3, this prior is the aggregated Dirac delta variational posterior of all inputs before fitting the Gaussian mixture. This definitely has overfitted the data, not to mention the variational posterior is actually deterministic. And in the

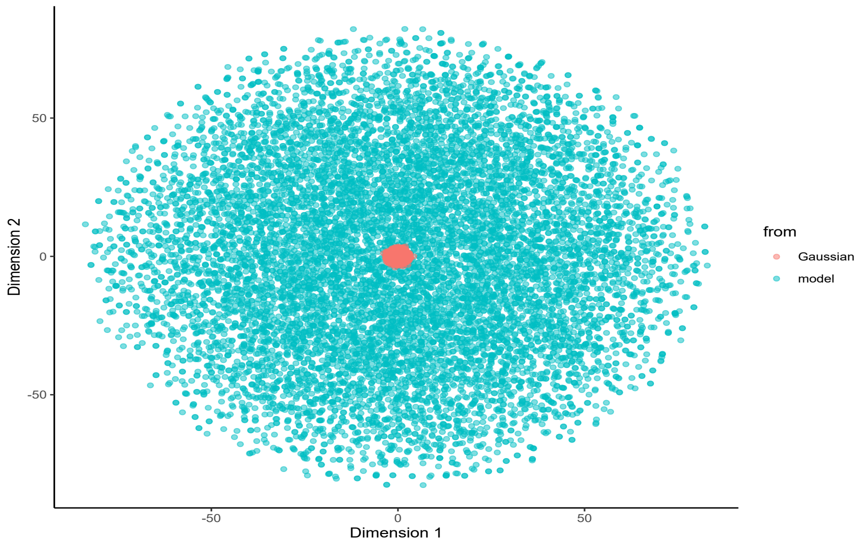


Figure 4.1: The empirical distribution of samples from the MAF prior (SSVAT-MAF) after dimension reduction

Note: The blue points are the samples from the MAF prior after dimension reduction. The red points are the samples from a 2-dimensional standard Gaussian. Each group contains 10,000 points.

study of [19], they regularize the decoder when training the AE, which helps to smooth the latent distribution. So the effectiveness of the AEA prior can potentially depend on the regularization.

In conclusion, the expressive priors do not always improve the model performances. In my models, the VAMP prior does not change the model performances since it degenerates to a standard Gaussian. The MAF prior improves the model performances probably by constructing a more dispersed distribution. The AEA prior harms the model performances, which may be due to the overfitting of data.

4.3 Best Models

No matter whether the model performances are measured by likelihood or sample quality, the SSVAT (or its variants) and the DAD consistently outperform the others. But they encode the data in different ways.

The SSVAT treats a count vector as a set of object one-hot embeddings. It can be regarded as an order-agnostic autoregression model where even the element number is variable across inputs. It is designed for the sparse data but may reflect the generation process. That is, the count data can be seen as a compression of sequence data. When the object instances are recognized from an image, the process is very likely to be sequential since human attention and visual receptive field are limited. A participant who glances over the image may follow a specific direction (e.g., top to down, left to right) and recognize the objects successively (extensive psychological and physiological studies have supported the sequential object recognition in human vision [79, 15, 33]). So the original count data are sequences but the sequential information is lost when they are compressed into count vectors. What the SSVAT does is trying to recover the count vectors into sequences and maximize the ELBO. Since the original direction is unknown, or may follow a uniform distribution, it is reasonable that the SSVAT does not consider any position embedding information.

The DAD treats a count vector as a concatenation of count one-hot embeddings, where the order is also agnostic. Although it is trained and generates samples in an autoregressive way, it is equivalent to a diffusion model and shows some relevant patterns (i.e., lower ELBO but higher sample quality). Compared to the SSVAT, it should be more stable since the number of elements is fixed and it can be extended to other data. But as mentioned in Section 2.1.1.3, it requires an explicit assumption that the observed maximum counts are the factual maximum counts. Besides, it is not very intuitive why the count data can be generated from a Dirac delta distribution where all elements are masked.

Both the SSVAT and the DAD generate elements in an autoregressive way, showing the ability of autoregression in representing high-dimensional distributions. But another autoregressive model, the DAF performs poorly among the models. A further check shows it degenerates to a factorized categorical distribution, i.e., the μ transformation is insensitive to the inputs. I speculate the reason is the *mod* manipulation is incompatible with the categorical encoding of the count data. Consider a critical condition where $\boldsymbol{x}_i$ is $K-2$ (suppose the maximum count plus 1 is K). When $\mu_i(\boldsymbol{z}^{1:(i-1)})$ is increased by 1, $\boldsymbol{x}_i$ will be $K-1$, the maximum count value. But when $\mu_i(\boldsymbol{z}^{1:(i-1)})$ is further increased by 1, $\boldsymbol{x}_i$ will be 0, the minimum count value. This is a abrupt transition that makes the function unsmooth, which should cause a large loss and finally makes the model degenerate to a factorized categorical distribution.

Chapter 5

Conclusions

In this thesis, I compare 5 generative models and their variants assuming different priors or conditional distributions on modeling object co-occurrences in images. The high-dimensional sparse count data are a representative discrete data which have not been thoroughly investigated before. This thesis contributes to this field by exploring how different generation processes and encoding methods can accommodate the data.

The most important finding is the SSVAT and the DAD perform best among the models, while the former is better on likelihood and the latter is better on sample quality. It indicates the potential of encoding the sparse counts as sequences and the advantage of the diffusion process in sample generation.

The idea of encoding the count vectors as sets comes from the trick of storing only the non-zero elements in sparse matrices, which are a common data type in machine learning, especially the natural language processing tasks. It can be extended to any vectors if most elements share the same value. The idea of transforming set data to sequential data is not novel, no matter in traditional statistical learning models [78], or in neural network models [63]. However, they either make unrealistic assumptions like the independence between set items, or use a simple autoregressive model without directly optimizing the set probabilities. The SSVAT overcomes the problems by introducing a sampling process over all possible sequences. The sanity check and the main experiment results show the effectiveness of this model. The success of the SSVAT demonstrates the potential of using neural autoregreesive models to explain and generate non-sequential data. From another perspective, the SSVAT can be viewed as a topic model [4, 5] which also considers the sequence length since the order of tokens is irrelevant. Then the generation process can be decomposed as follows: First, sample a topic of the sequence (i.e., the image type, like indoor or outdoor. We can use a distributed representation to represent the topics, even assume each element of the representation vector is not a category value, but a sample from a Dirichlet distribution). Second, choose a sequence length based on the topic (some images may contain more object instances). Finally, generate each token one by one conditioned on the topic (each topic should correspond to a multinomial distribution over object classes).

However, the SSVAT's flaw is obvious too. It is tailored for the sparse data. The computational cost will increase when the number of object classes and their counts increase. The stability of the estimate will also decrease in such situations. A possible solution is to merge items of the same object class. For example, the set [chair, chair, table] can be represented as [2 chairs, 1 table]. When the model predict a token, it should both predict the object class and its count (i.e., adding another output layer for the count value, which should be simple if not using a categorical conditional distribution[1]). But we should introduce a masking mechanism to ensure the model only predict the object classes which have not been given or predicted before. The generation still stop once a [EOS] token is predicted (and its corresponding count does not matter). This method restrict the maximum sequence length to be the number of object classes plus 2 ([BOS] and [EOS]) but is still in a sequence-to-set framework, and can be explored in future studies.

The diffusion model [57, 32, 60] has got much attention recently due to its ability to generate diverse and high-quality samples. Some studies of the discrete diffusion [2, 34] has shown it works well on generating image and language data. This thesis demonstrates it also perform well in generating the count data when encoding the counts as categories. The diffusion model comes from the observation that the structured data can be transformed to noise easily. The DAD circumvents the direct optimization of the denoising process but train and generate samples in an order-agnostic autoregressive manner. However, [34] find the DAD perform worse than the pure autoregressive models on language data. Since the thesis does not consider such models, whether the pure autoregressive models can perform even better than the DAD (and the SSVAT) remains a subject for future studies.

This thesis also find how to encode the sparse counts can significantly influence the model performances. Although the counts are sparse, the zero-inflation assumption is not necessary when encoding them as non-negative integers. The main reason should be that we do not directly rely on the conditional distribution to explain the

[1]If we assume the maximum counts are the same for all object classes, even a categorical conditional distribution can be used since different object classes can share the same layer to predict counts.

counts. The latent distribution and the neural network are flexible enough to explain the widespread 0s (One piece of evidence is the performance differences between the Poisson distribution and other conditional distributions are much smaller when using a latent distribution). As a count distribution, the NB distribution itself can explain the sparsity of the data. It also performs well in more complex model architectures. But a potential problem is it may generate unrealistic samples, which can influence downstream tasks. The categorical distribution provides a flexible way to encode the sparse counts. It is also the DAD's encoding method. But it can be unstable and computationally costly since many output layers are introduced. Besides, implicitly encoding the counts like the SSVAT may also be a feasible choice for the sparse data. For future studies, an interesting direction is to use a continuous distribution to represent the count data. For example, we can apply $\log(x+z)$ where z is a positive value to transform a non-negative integer x. Some techniques like uniform dequantization [64] can also help realize it. After transforming the counts to continuous data, there are definitely more possible model options.

Another finding is the expressive priors do not necessarily improve the performances of the VAE and the SSVAT. The VAMP prior degenerates to a standard Gaussian. The AEA prior appears to harm the model performances probably due to the overfitting of data. Only the MAF prior obviously improves the model performances. Among the 3 priors, the MAF's form is also the most flexible since the two others are restricted to a mixture model with factorized Gaussian components. Its success demonstrates the strength of combing the autoregressive models and the normalizing flow, showing a combination of generative models can be beneficial. In the meantime, the empirical distribution of samples from the MAF prior after dimension reduction seems like a spherical Gaussian with large variances, indicating a dispersed prior may be advantageous (although theoretically the variances of a factorized Gaussian prior should not matter [14]. It may have some effects[2] in practice like why we use the expressive priors). In future work, we can try some other priors constructed by generative models asides from the MAF, like a diffusion prior [75].

Due to the time limit, this thesis only test some representative generative models. There are many other options like the pure autoregressive models and the energy-based models. Also, this thesis only considers some shallow network architectures due to the computational cost, future work can delve into deeper and wider network architectures,

[2]For example, a dispersed prior may play a role of practical regularization like putting a 0-centered prior belief on parameters in traditional Bayesian models.

even more stochastic layers [70]. Besides, the success of the SSVAT and and that of the MAF prior demonstrate the potential benefits of combining multiple generative models. This idea of hybrid modelling [66] can be further explored in the future.

The final concern is about the dataset. On the one hand, the object count data seem to be under-represented even the minimum sub-dataset contains 10,000 vectors. An assumption of this thesis is that the 3 sub-datasets do reflect the ground-truth distribution that generates the count vectors. The factual lower bounds of the mean losses on the three sub-datasets are their empirical entropy, which is 9.384, 8.207 and 8.652 respectively. The minimum value is equal to the entropy of a uniform distribution over 3666.526 categories, while the maximum value is equal to the entropy of a uniform distribution over 11896.506 categories. The discrepancy in the empirical entropy indicates that the 3 sub-datasets can be heterogeneous and at least the validation set is under-representing the ground-truth distribution since the empirical entropy based on the largest sub-dataset should be more trustworthy[3]. Future work can consider larger datasets or a more efficient way to split the dataset to mitigate the problem. On the other hand, this thesis is restricted to the object count data, future studies can consider testing the models (especially the SSVAT) on other count datasets like those in genomics [1, 46, 69].

[3]In this situation, sample quality seems to be a more robust metric as it is not explicitly bounded by the sub-dataset empirical entropy (although still influenced by it). A potential direction is to see whether the data under-representation also explains why the diffusion can have better sample quality but lower ELBO compared to most generative models.

Bibliography

[1] Xusheng Ai, Melissa C Smith, and Frank Alex Feltus. Generative adversarial networks applied to gene expression analysis: An interdisciplinary perspective. *Computational and Systems Oncology*, 3(3):e1050, 2023.

[2] Jacob Austin, Daniel D. Johnson, Jonathan Ho, Daniel Tarlow, and Rianne van den Berg. Structured denoising diffusion models in discrete state-spaces. In Marc'Aurelio Ranzato, Alina Beygelzimer, Yann N. Dauphin, Percy Liang, and Jennifer Wortman Vaughan, editors, *Advances in Neural Information Processing Systems 34: Annual Conference on Neural Information Processing Systems 2021, NeurIPS 2021, December 6-14, 2021, virtual*, pages 17981–17993, 2021.

[3] Lei Jimmy Ba, Jamie Ryan Kiros, and Geoffrey E. Hinton. Layer normalization. *CoRR*, abs/1607.06450, 2016.

[4] David M. Blei. Probabilistic topic models. *Communications of the ACM*, 55(4):77–84, 2012.

[5] David M. Blei, Andrew Y. Ng, and Michael I. Jordan. Latent dirichlet allocation. *Journal of Machince Learning Research*, 3:993–1022, 2003.

[6] Samuel R. Bowman, Luke Vilnis, Oriol Vinyals, Andrew M. Dai, Rafal Józefowicz, and Samy Bengio. Generating sentences from a continuous space. In Yoav Goldberg and Stefan Riezler, editors, *Proceedings of the 20th SIGNLL Conference on Computational Natural Language Learning, CoNLL 2016, Berlin, Germany, August 11-12, 2016*, pages 10–21. ACL, 2016.

[7] Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter,

Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. Language models are few-shot learners. In Hugo Larochelle, Marc'Aurelio Ranzato, Raia Hadsell, Maria-Florina Balcan, and Hsuan-Tien Lin, editors, *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*, 2020.

[8] Yuri Burda, Roger B. Grosse, and Ruslan Salakhutdinov. Importance weighted autoencoders. In Yoshua Bengio and Yann LeCun, editors, *4th International Conference on Learning Representations, ICLR 2016, San Juan, Puerto Rico, May 2-4, 2016, Conference Track Proceedings*, 2016.

[9] Ali Taylan Cemgil. Bayesian inference for nonnegative matrix factorisation models. *Computational Intelligence and Neuroscience*, 2009.

[10] Dong-Kyu Chae, Jin-Soo Kang, Sang-Wook Kim, and Jaeho Choi. Rating augmentation with generative adversarial networks towards accurate collaborative filtering. In Ling Liu, Ryen W. White, Amin Mantrach, Fabrizio Silvestri, Julian J. McAuley, Ricardo Baeza-Yates, and Leila Zia, editors, *The World Wide Web Conference, WWW 2019, San Francisco, CA, USA, May 13-17, 2019*, pages 2616–2622. ACM, 2019.

[11] Xi Chen, Diederik P. Kingma, Tim Salimans, Yan Duan, Prafulla Dhariwal, John Schulman, Ilya Sutskever, and Pieter Abbeel. Variational lossy autoencoder. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*. OpenReview.net, 2017.

[12] A. P. Dempster, N. M. Laird, and D. B. Rubin. Maximum likelihood from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society: Series B*, 39:1–38, 1977.

[13] Stefanie Dencks, Marion Piepenbrock, and Georg Schmitz. Assessing vessel reconstruction in ultrasound localization microscopy by maximum likelihood estimation of a zero-inflated poisson model. *IEEE Transactions on Ultrasonics, Ferroelectrics, and Frequency Control*, 67(8):1603–1612, 2020.

[14] Carl Doersch. Tutorial on variational autoencoders. *CoRR*, abs/1606.05908, 2016.

[15] Tom Foulsham and Geoffrey Underwood. What can saliency models predict about eye movements? Spatial and sequential aspects of fixations during encoding and recognition. *Journal of Vision*, 8(2):6–6, 02 2008.

[16] Kunihiko Fukushima. Visual feature extraction by a multilayered network of analog threshold elements. *IEEE Transactions on Systems Science and Cybernetics*, 5(4):322–333, 1969.

[17] Min Gao, Junwei Zhang, Junliang Yu, Jundong Li, Junhao Wen, and Qingyu Xiong. Recommender systems based on generative adversarial networks: A problem-driven perspective. *Information Sciences*, 546:1166–1185, 2021.

[18] Mathieu Germain, Karol Gregor, Iain Murray, and Hugo Larochelle. MADE: masked autoencoder for distribution estimation. In Francis R. Bach and David M. Blei, editors, *Proceedings of the 32nd International Conference on Machine Learning, ICML 2015, Lille, France, 6-11 July 2015*, volume 37 of *JMLR Workshop and Conference Proceedings*, pages 881–889. JMLR.org, 2015.

[19] Partha Ghosh, Mehdi S. M. Sajjadi, Antonio Vergari, Michael J. Black, and Bernhard Schölkopf. From variational to deterministic autoencoders. In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. OpenReview.net, 2020.

[20] Xavier Glorot and Yoshua Bengio. Understanding the difficulty of training deep feedforward neural networks. In Yee Whye Teh and D. Mike Titterington, editors, *Proceedings of the 13th International Conference on Artificial Intelligence and Statistics, AISTATS 2010, Chia Laguna Resort, Sardinia, Italy, May 13-15, 2010*, volume 9 of *JMLR Proceedings*, pages 249–256. JMLR.org, 2010.

[21] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In Z. Ghahramani, M. Welling, C. Cortes, N. Lawrence, and K.Q. Weinberger, editors, *Advances in Neural Information Processing Systems 27*. Curran Associates, Inc., 2014.

[22] Prem Gopalan, Jake M. Hofman, and David M. Blei. Scalable recommendation with hierarchical poisson factorization. In Marina Meila and Tom Heskes, editors, *Proceedings of the 31st Conference on Uncertainty in Artificial Intelligence, UAI*

2015, July 12-16, 2015, Amsterdam, The Netherlands, pages 326–335. AUAI Press, 2015.

[23] Prasoon Goyal, Zhiting Hu, Xiaodan Liang, Chenyu Wang, Eric P. Xing, and Carnegie Mellon. Nonparametric variational auto-encoders for hierarchical representation learning. In *IEEE International Conference on Computer Vision, ICCV 2017, Venice, Italy, October 22-29, 2017*, pages 5104–5112. IEEE Computer Society, 2017.

[24] Roberto Gozalo-Brizuela and Eduardo C. Garrido-Merchán. Chatgpt is not all you need. A state of the art review of large generative AI models. *CoRR*, abs/2301.04655, 2023.

[25] William H. Greene. Accounting for excess zeros and sample selection in poisson and negative binomial regression models. *ERN: Discrete Regression & Qualitative Choice Models (Single) (Topic)*, 1994.

[26] Tanmay Gupta, Alexander G. Schwing, and Derek Hoiem. Vico: Word embeddings from visual co-occurrences. In *2019 IEEE/CVF International Conference on Computer Vision, ICCV 2019, Seoul, Korea (South), October 27 - November 2, 2019*, pages 7424–7433. IEEE, 2019.

[27] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *2016 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2016, Las Vegas, NV, USA, June 27-30, 2016*, pages 770–778. IEEE Computer Society, 2016.

[28] Kristoffer H. Hellton, Jeffrey Cummings, Audun Osland Vik-Mo, Jan Erik Nordrehaug, Dag Aarsland, Geir Selbaek, and Lasse Melvaer Giil. The truth behind the zeros: A new approach to principal component analysis of the neuropsychiatric inventory. *Multivariate Behavioral Research*, 56(1):70–85, 2021. PMID: 32329370.

[29] Martin Heusel, Hubert Ramsauer, Thomas Unterthiner, Bernhard Nessler, and Sepp Hochreiter. Gans trained by a two time-scale update rule converge to a local nash equilibrium. In Isabelle Guyon, Ulrike von Luxburg, Samy Bengio, Hanna M. Wallach, Rob Fergus, S. V. N. Vishwanathan, and Roman Garnett,

editors, *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA*, pages 6626–6637, 2017.

[30] G. E. Hinton and R. R. Salakhutdinov. Reducing the dimensionality of data with neural networks. *Science*, 313(5786):504–507, 2006.

[31] Geoffrey E. Hinton and Ruslan Salakhutdinov. Discovering binary codes for documents by learning deep generative models. *Topics of Cognitive Science*, 3(1):74–91, 2011.

[32] Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. In Hugo Larochelle, Marc'Aurelio Ranzato, Raia Hadsell, Maria-Florina Balcan, and Hsuan-Tien Lin, editors, *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*, 2020.

[33] James E Hoffman and Baskaran Subramaniam. The role of visual attention in saccadic eye movements. *Perception and Psychophysics*, 57(6):787–795, 1995.

[34] Emiel Hoogeboom, Alexey A. Gritsenko, Jasmijn Bastings, Ben Poole, Rianne van den Berg, and Tim Salimans. Autoregressive diffusion models. In *The 10th International Conference on Learning Representations, ICLR 2022, Virtual Event, April 25-29, 2022*. OpenReview.net, 2022.

[35] Emiel Hoogeboom, Didrik Nielsen, Priyank Jaini, Patrick Forré, and Max Welling. Argmax flows and multinomial diffusion: Learning categorical distributions. In Marc'Aurelio Ranzato, Alina Beygelzimer, Yann N. Dauphin, Percy Liang, and Jennifer Wortman Vaughan, editors, *Advances in Neural Information Processing Systems 34: Annual Conference on Neural Information Processing Systems 2021, NeurIPS 2021, December 6-14, 2021, virtual*, pages 12454–12465, 2021.

[36] Emiel Hoogeboom, Jorn W. T. Peters, Rianne van den Berg, and Max Welling. Integer discrete flows and lossless compression. In Hanna M. Wallach, Hugo Larochelle, Alina Beygelzimer, Florence d'Alché-Buc, Emily B. Fox, and Roman Garnett, editors, *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, December 8-14, 2019, Vancouver, BC, Canada*, pages 12134–12144, 2019.

[37] Phillip Isola, Daniel Zoran, Dilip Krishnan, and Edward H. Adelson. Learning visual groups from co-occurrences in space and time. *CoRR*, abs/1511.06811, 2015.

[38] Eric Jang, Shixiang Gu, and Ben Poole. Categorical reparameterization with gumbel-softmax. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*. OpenReview.net, 2017.

[39] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In Yoshua Bengio and Yann LeCun, editors, *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*, 2015.

[40] Diederik P. Kingma and Max Welling. Auto-encoding variational bayes. In Yoshua Bengio and Yann LeCun, editors, *2nd International Conference on Learning Representations, ICLR 2014, Banff, AB, Canada, April 14-16, 2014, Conference Track Proceedings*, 2014.

[41] Diane Lambert. Zero-inflated poisson regression, with an application to defects in manufacturing. *Technometrics*, 34(1):1–14, 1992.

[42] Juho Lee, Yoonho Lee, Jungtaek Kim, Adam R. Kosiorek, Seungjin Choi, and Yee Whye Teh. Set transformer: A framework for attention-based permutation-invariant neural networks. In Kamalika Chaudhuri and Ruslan Salakhutdinov, editors, *Proceedings of the 36th International Conference on Machine Learning, ICML 2019, 9-15 June 2019, Long Beach, California, USA*, volume 97 of *Proceedings of Machine Learning Research*, pages 3744–3753. PMLR, 2019.

[43] Zeping Li. Informatics poject proposal: Modelling object co-occurrences in images, 2023.

[44] Dawen Liang, Rahul G. Krishnan, Matthew D. Hoffman, and Tony Jebara. Variational autoencoders for collaborative filtering. In Pierre-Antoine Champin, Fabien Gandon, Mounia Lalmas, and Panagiotis G. Ipeirotis, editors, *Proceedings of the 2018 World Wide Web Conference on World Wide Web, WWW 2018, Lyon, France, April 23-27, 2018*, pages 689–698. ACM, 2018.

[45] Tsung-Yi Lin, Michael Maire, Serge J. Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C. Lawrence Zitnick. Microsoft COCO: Common Objects in Context. In David J. Fleet, Tomás Pajdla, Bernt Schiele, and Tinne Tuytelaars, editors, *Computer Vision - ECCV 2014 - 13th European Conference, Zurich, Switzerland, September 6-12, 2014, Proceedings, Part V*, volume 8693 of *Lecture Notes in Computer Science*, pages 740–755. Springer, 2014.

[46] Romain Lopez, Jeffrey Regier, Michael B Cole, Michael I Jordan, and Nir Yosef. Deep generative modeling for single-cell transcriptomics. *Nature Methods*, 15(12):1053–1058, 2018.

[47] Chao Ma, Wenbo Gong, José Miguel Hernández-Lobato, Noam Koenigstein, Sebastian Nowozin, and Cheng Zhang. Partial vae for hybrid recommender system. In *Bayesian Deep Learning*, 2018. NIPS 2018 Workshop ; Conference date: 07-12-2018 Through 07-12-2018.

[48] Chris J. Maddison, Andriy Mnih, and Yee Whye Teh. The concrete distribution: A continuous relaxation of discrete random variables. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*. OpenReview.net, 2017.

[49] Yishu Miao, Lei Yu, and Phil Blunsom. Neural variational inference for text processing. In Maria-Florina Balcan and Kilian Q. Weinberger, editors, *Proceedings of the 33nd International Conference on Machine Learning, ICML 2016, New York City, NY, USA, June 19-24, 2016*, volume 48 of *JMLR Workshop and Conference Proceedings*, pages 1727–1736. JMLR.org, 2016.

[50] Kevin P. Murphy. *Probabilistic Machine Learning: Advanced Topics*, chapter Generation, pages 763–914. MIT Press, 2023.

[51] Eric T. Nalisnick and Padhraic Smyth. Stick-breaking variational autoencoders. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*. OpenReview.net, 2017.

[52] Radford M. Neal. *Bayesian learning for neural networks*. PhD thesis, University of Toronto, Canada, 1995.

[53] George Papamakarios, Iain Murray, and Theo Pavlakou. Masked autoregressive flow for density estimation. In Isabelle Guyon, Ulrike von Luxburg, Samy Bengio,

Hanna M. Wallach, Rob Fergus, S. V. N. Vishwanathan, and Roman Garnett, editors, *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA*, pages 2338–2347, 2017.

[54] George Papamakarios, Eric T. Nalisnick, Danilo Jimenez Rezende, Shakir Mohamed, and Balaji Lakshminarayanan. Normalizing flows for probabilistic modeling and inference. *Journal of Machine Learning Research*, 22:1–64, 2021.

[55] Laurent U Perrinet and James A Bednar. Edge co-occurrences can account for rapid categorization of natural versus animal images. *Scientific Reports*, 5:11400, 2015.

[56] Danilo Jimenez Rezende, Shakir Mohamed, and Daan Wierstra. Stochastic backpropagation and approximate inference in deep generative models. In *Proceedings of the 31th International Conference on Machine Learning, ICML 2014, Beijing, China, 21-26 June 2014*, volume 32 of *JMLR Workshop and Conference Proceedings*, pages 1278–1286. JMLR.org, 2014.

[57] Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. High-resolution image synthesis with latent diffusion models. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR 2022, New Orleans, LA, USA, June 18-24, 2022*, pages 10674–10685. IEEE, 2022.

[58] Shibani Santurkar, Ludwig Schmidt, and Aleksander Madry. A classification-based study of covariate shift in GAN distributions. In Jennifer G. Dy and Andreas Krause, editors, *Proceedings of the 35th International Conference on Machine Learning, ICML 2018, Stockholmsmässan, Stockholm, Sweden, July 10-15, 2018*, volume 80 of *Proceedings of Machine Learning Research*, pages 4487–4496. PMLR, 2018.

[59] Huajie Shao, Shuochao Yao, Dachun Sun, Aston Zhang, Shengzhong Liu, Dongxin Liu, Jun Wang, and Tarek F. Abdelzaher. Controlvae: Controllable variational autoencoder. In *Proceedings of the 37th International Conference on Machine Learning, ICML 2020, 13-18 July 2020, Virtual Event*, volume 119 of *Proceedings of Machine Learning Research*, pages 8655–8664. PMLR, 2020.

[60] Jascha Sohl-Dickstein, Eric A. Weiss, Niru Maheswaranathan, and Surya Ganguli. Deep unsupervised learning using nonequilibrium thermodynamics. In Francis R.

Bach and David M. Blei, editors, *Proceedings of the 32nd International Conference on Machine Learning, ICML 2015, Lille, France, 6-11 July 2015*, volume 37 of *JMLR Workshop and Conference Proceedings*, pages 2256–2265. JMLR.org, 2015.

[61] Nitish Srivastava, Geoffrey E. Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: a simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15(1):1929–1958, 2014.

[62] Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jonathon Shlens, and Zbigniew Wojna. Rethinking the inception architecture for computer vision. In *2016 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2016, Las Vegas, NV, USA, June 27-30, 2016*, pages 2818–2826. IEEE Computer Society, 2016.

[63] Longtao Tang, Ying Zhou, and Yu Yang. Sequence-to-set generative models. In S. Koyejo, S. Mohamed, A. Agarwal, D. Belgrave, K. Cho, and A. Oh, editors, *Advances in Neural Information Processing Systems*, volume 35, pages 14986–14997. Curran Associates, Inc., 2022.

[64] Lucas Theis, Aäron van den Oord, and Matthias Bethge. A note on the evaluation of generative models. In Yoshua Bengio and Yann LeCun, editors, *4th International Conference on Learning Representations, ICLR 2016, San Juan, Puerto Rico, May 2-4, 2016, Conference Track Proceedings*, 2016.

[65] Sushrut Thorat, Genevieve L. Quek, and Marius V. Peelen. Statistical learning of distractor co-occurrences facilitates visual search. *Journal of Vision*, 22(10):2, 2022.

[66] Jakub M. Tomczak. *Deep Generative Modeling*. Springer, 2022.

[67] Jakub M. Tomczak and Max Welling. VAE with a vampprior. In Amos J. Storkey and Fernando Pérez-Cruz, editors, *International Conference on Artificial Intelligence and Statistics, AISTATS 2018, 9-11 April 2018, Playa Blanca, Lanzarote, Canary Islands, Spain*, volume 84 of *Proceedings of Machine Learning Research*, pages 1214–1223. PMLR, 2018.

[68] Dustin Tran, Keyon Vafa, Kumar Krishna Agrawal, Laurent Dinh, and Ben Poole. Discrete flows: Invertible generative models of discrete data. In Hanna M. Wallach,

Hugo Larochelle, Alina Beygelzimer, Florence d'Alché-Buc, Emily B. Fox, and Roman Garnett, editors, *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, December 8-14, 2019, Vancouver, BC, Canada*, pages 14692–14701, 2019.

[69] Martin Treppner, Harald Binder, and Moritz Hess. Interpretable generative deep learning: an illustration with single cell gene expression data. *Human Genetics*, 141(9):1481–1498, 2022.

[70] Arash Vahdat and Jan Kautz. NVAE: A deep hierarchical variational autoencoder. In Hugo Larochelle, Marc'Aurelio Ranzato, Raia Hadsell, Maria-Florina Balcan, and Hsuan-Tien Lin, editors, *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*, 2020.

[71] Aäron van den Oord, Oriol Vinyals, and Koray Kavukcuoglu. Neural discrete representation learning. In Isabelle Guyon, Ulrike von Luxburg, Samy Bengio, Hanna M. Wallach, Rob Fergus, S. V. N. Vishwanathan, and Roman Garnett, editors, *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA*, pages 6306–6315, 2017.

[72] Laurens van der Maaten and Geoffrey Hinton. Visualizing data using t-sne. *Journal of Machine Learning Research*, 9(86):2579–2605, 2008.

[73] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. In Isabelle Guyon, Ulrike von Luxburg, Samy Bengio, Hanna M. Wallach, Rob Fergus, S. V. N. Vishwanathan, and Roman Garnett, editors, *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA*, pages 5998–6008, 2017.

[74] Kexin Wang, Nils Reimers, and Iryna Gurevych. Tsdae: Using transformer-based sequential denoising auto-encoder for unsupervised sentence embedding learning. In *Conference on Empirical Methods in Natural Language Processing*, 2021.

[75] Antoine Wehenkel and Gilles Louppe. Diffusion priors in variational autoencoders. *CoRR*, abs/2106.15671, 2021.

[76] Christopher K. I. Williams and Charlie Nash. Autoencoders and probabilistic inference with missing data: An exact solution for the factor analysis case. *CoRR*, abs/1801.03851, 2018.

[77] Ronald J. Williams and David Zipser. A learning algorithm for continually running fully recurrent neural networks. *Neural Computation*, 1(2):270–280, 1989.

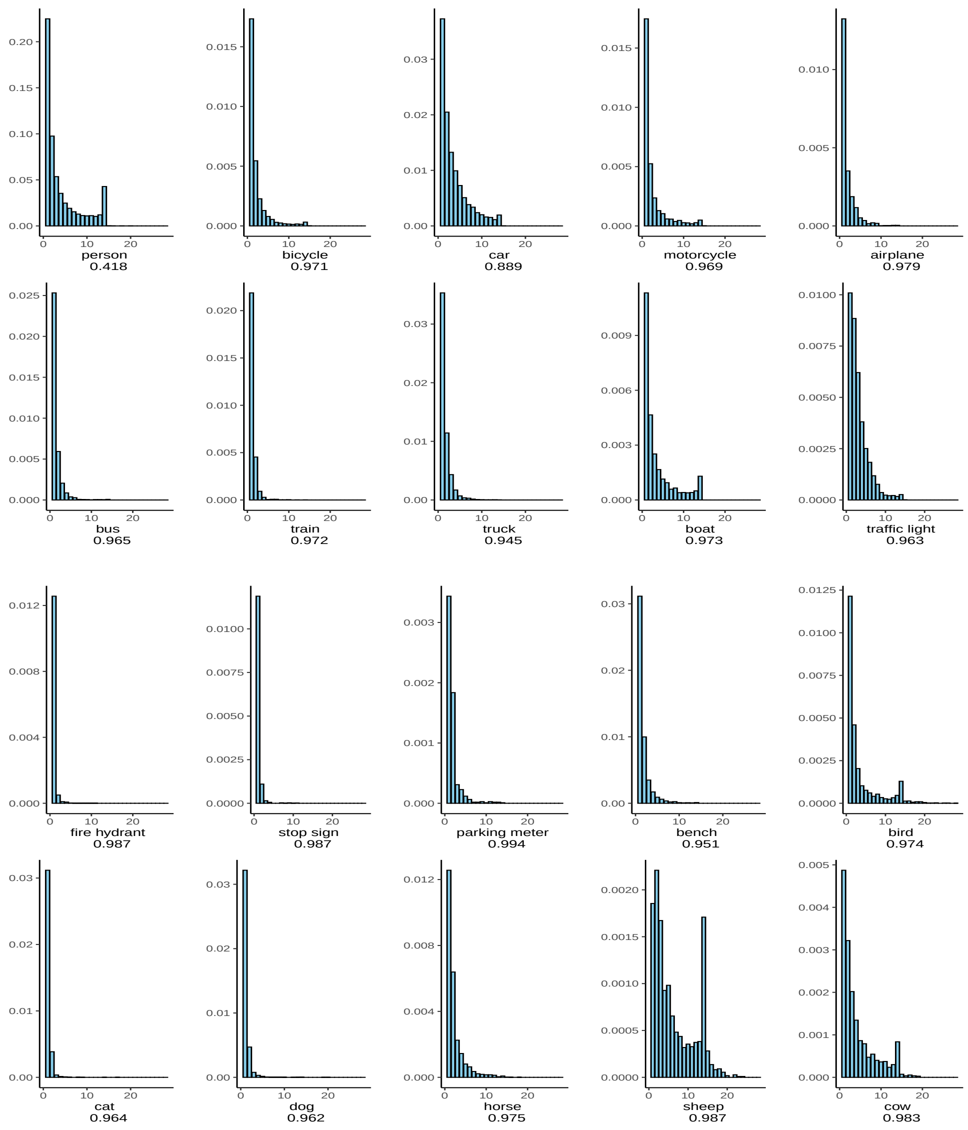
[78] Tongwen Wu, Yu Yang, Yanzhi Li, Huiqiang Mao, Liming Li, Xiaoqing Wang, and Yuming Deng. Representation learning for predicting customer orders. In Feida Zhu, Beng Chin Ooi, and Chunyan Miao, editors, *KDD '21: The 27th ACM SIGKDD Conference on Knowledge Discovery and Data Mining, Virtual Event, Singapore, August 14-18, 2021*, pages 3735–3744. ACM, 2021.

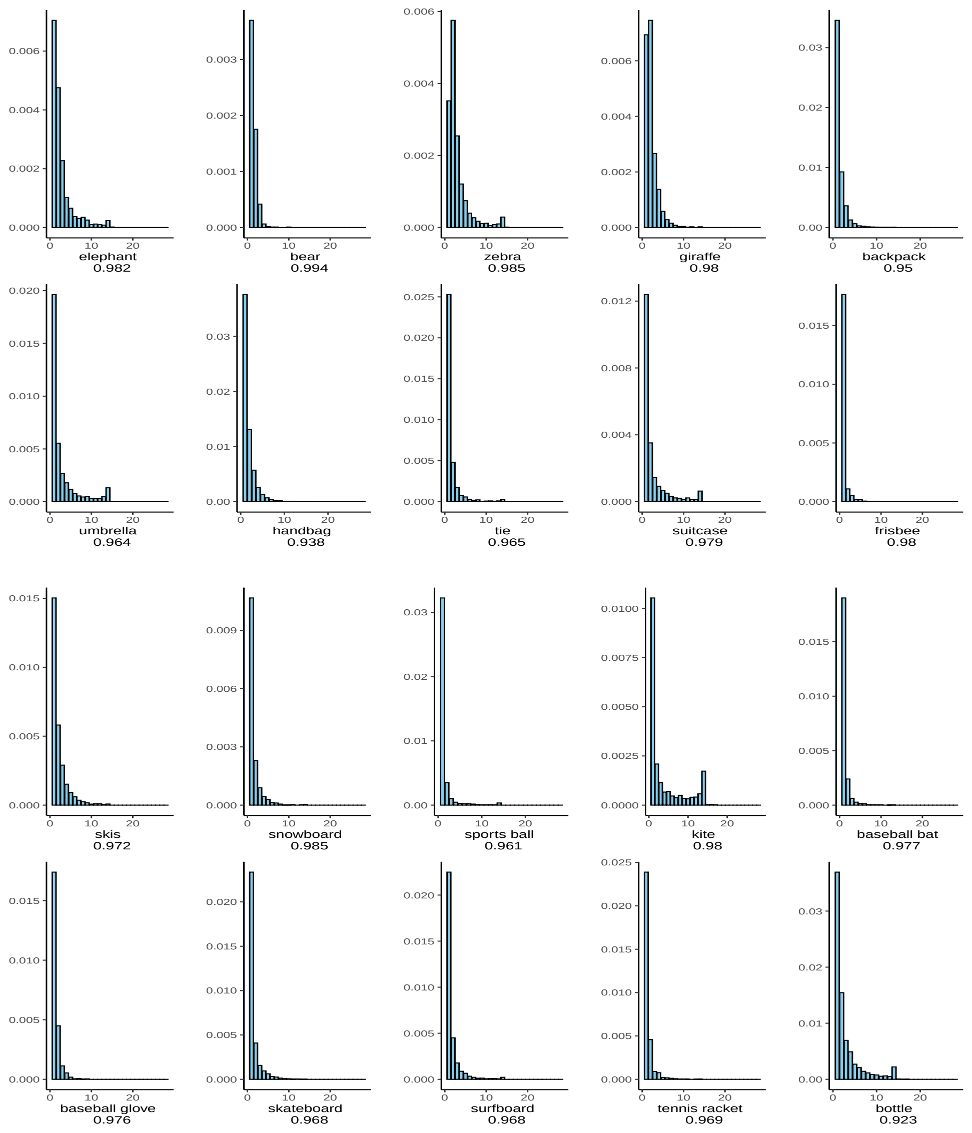
[79] Alfred L. Yarbus. *Eye Movements During Fixation on Stationary Objects*, pages 103–127. Springer US, Boston, MA, 1967.

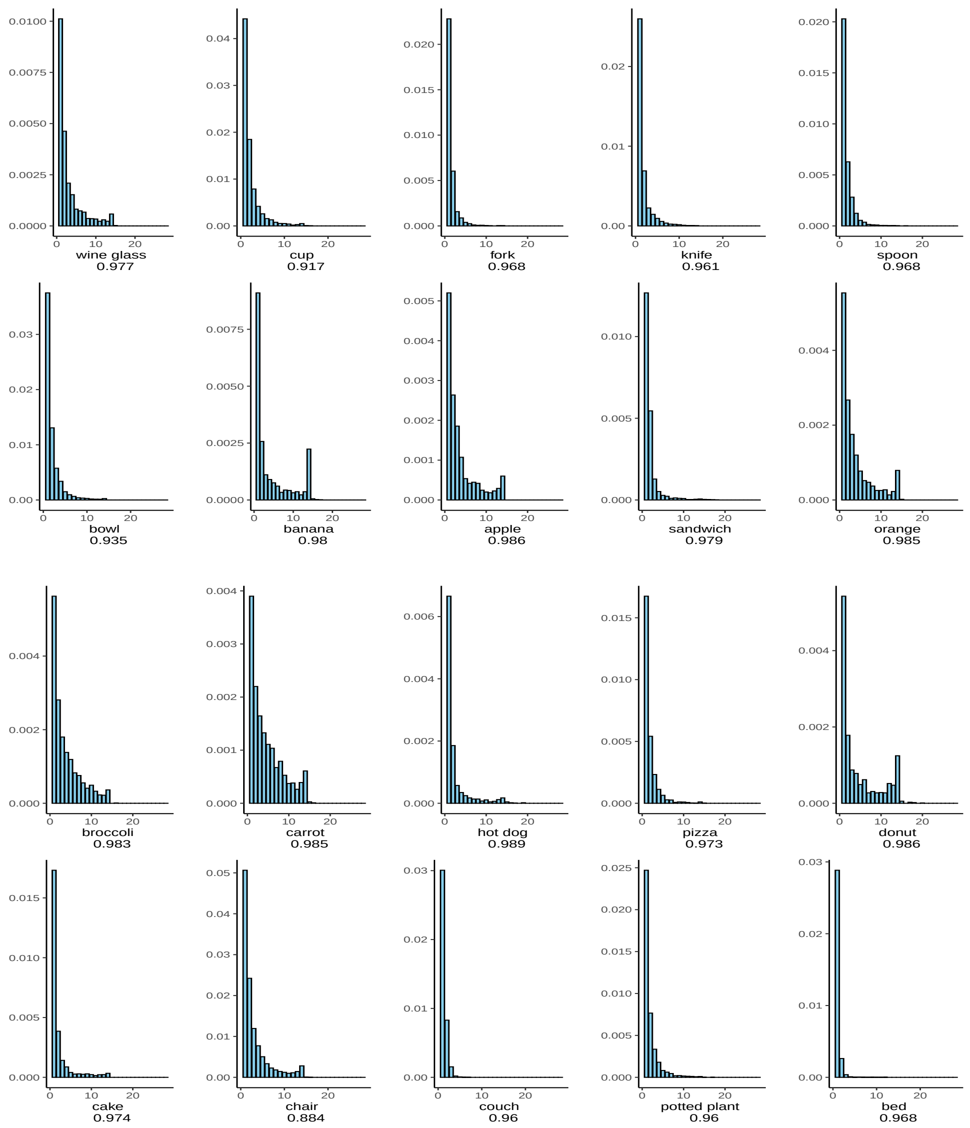
[80] He Zhao, Piyush Rai, Lan Du, Wray L. Buntine, Dinh Phung, and Mingyuan Zhou. Variational autoencoders for sparse and overdispersed discrete data. In Silvia Chiappa and Roberto Calandra, editors, *The 23rd International Conference on Artificial Intelligence and Statistics, AISTATS 2020, 26-28 August 2020, Online [Palermo, Sicily, Italy]*, volume 108 of *Proceedings of Machine Learning Research*, pages 1684–1694. PMLR, 2020.

Appendix A

Empirical Count Distributions of Different Object Classes


person
0.418
bicycle
0.971
car
0.889
motorcycle
0.969
airplane
0.979
bus
0.965
train
0.972
truck
0.945
boat
0.973
traffic light
0.963
fire hydrant
0.987
stop sign
0.987
parking meter
0.994
bench
0.951
bird
0.974
cat
0.964
dog
0.962
horse
0.975
sheep
0.987
cow
0.983


elephant
0.982
bear
0.994
zebra
0.985
giraffe
0.98
backpack
0.95
umbrella
0.964
handbag
0.938
tie
0.965
suitcase
0.979
frisbee
0.98
skis
0.972
snowboard
0.985
sports ball
0.961
kite
0.98
baseball bat
0.977
baseball glove
0.976
skateboard
0.968
surfboard
0.968
tennis racket
0.969
bottle
0.923


wine glass
0.977
cup
0.917
fork
0.968
knife
0.961
spoon
0.968
bowl
0.935
banana
0.98
apple
0.986
sandwich
0.979
orange
0.985
broccoli
0.983
carrot
0.985
hot dog
0.989
pizza
0.973
donut
0.986
cake
0.974
chair
0.884
couch
0.96
potted plant
0.96
bed
0.968

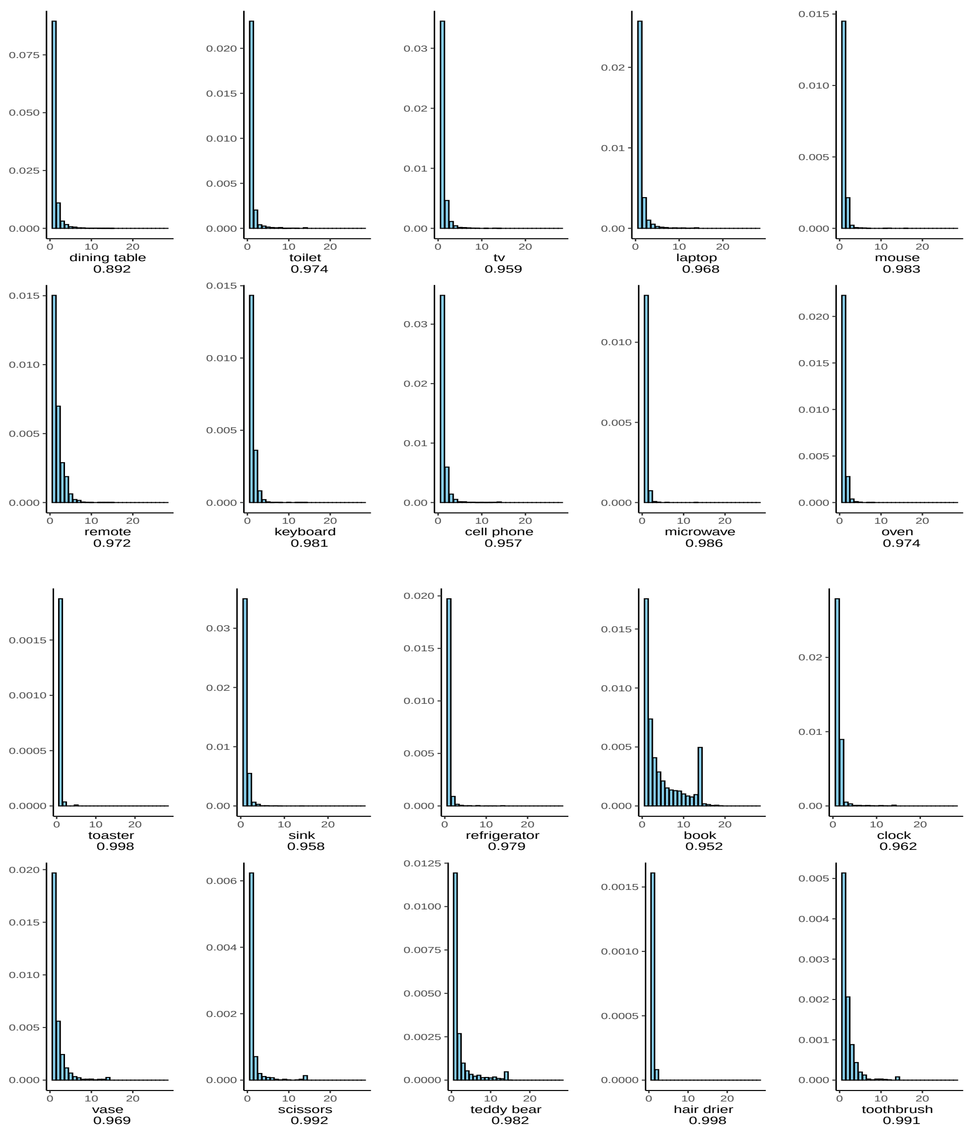


Figure A.1: Empirical count distributions of different object classes

Note: For each object class, the content below the horizontal axis indicates the class name and the relative frequency of the count 0. The horizontal axis shows the non-zero count values, and the vertical axis shows the corresponding relative frequencies. I avoid showing the relative frequencies of the count 0s since they can overwhelm the relative frequencies of other values in the visualization.

Appendix B

Sanity Check

For the SSVAT, a problem is that the probability estimates can be unrealistic and unstable with a large number of possible sequences. Consider a set having many possible sequences and we happen to use a sequence with a high probability under the current model to represent the set, the model can produce a probability estimate larger than 1, which is theoretically impossible (This problem also exists for the DAD, see Section 2.6, but [34] has shown it a reliable model). Although the expectation of the estimates is still a valid probability, it is worthwhile to conduct a sanity check. That is, whether the model can effectively approximate different distributions that are known.

In this sanity check, I consider three distributions: The mixture ZIP, the mixture ZINB and the mixture categorical distributions. Each mixture distribution has 10 components and the parameter values are estimated from the training set (see Section 3.1). Then I use each distribution to generate 3 sub-datasets, whose sizes are equal to those of the training, the validation and the test sets respectively. I fit and evaluate 3 models on the data. The first is a vanilla VAE assuming the same conditional distribution. It has no hidden layers in the encoder and the decoder, and the latent distribution dimension is the optimal value based on the hyperparameter tuning experiments (see Section 3.2 and Appendix C). The second is a SSVAT using 1 encoder/decoder layer. The embedding dimension is 4, with 4 heads for the attention sub-layer and a dropout rate of 0.1. The feedforward neural network sub-layer has 1 hidden layer, whose width is the twice the input layer width (i.e., the embedding dimension). The third is the best SSVAT variant (See Section 3.5.1). It has 3 encoder/decoder layers and a embedding dimension of 20. It also use an MAF prior. The other hyperparameters are the same as those in the second. I test it to see how close the SSVAT can approximate the distributions.

Distribution	**Ground Truth**	**VAE**	**SSVAT**	**SSVAT Best**
Mixture ZIP	64.095(0.072)	74.709(0.097)	87.821(0.142)	65.802(0.086)
Mixture ZINB	47.595(0.119)	53.591(0.126)	79.324(0.248)	51.158(0.148)
Mixture Categorical	15.823(0.064)	15.951(0.079)	23.043(0.149)	16.277(0.100)

Table B.1: Mean test losses with standard errors of different models for different distributions

The model fitting process is the same as that in the main text (see Section 3.3), and the evaluation metric is the mean loss (i.e., the negative ELBO) on the test set. I also calculate the mean test loss (i.e., the negative log-likelihood) of the ground-truth model (i.e., the mixture model with the parameter values mentioned above), which is an estimate of the entropy (although the base number is e rather than 2) of the ground-truth distribution. Its expectation is the lower bound of the expected mean test loss of any other models. The test loss of the VAE/SSVAT is the estimate of the cross entropy between the model distribution and the ground-truth distribution. The mean test losses with standard errors of different models for different distributions are shown in Table B.1.

It is expected that the mean test losses of the ground-truth model are the minimum since the data are generated by itself. It is also expected that the VAE can achieve performances close to those of the ground-truth model even without hidden layers, since they share the same conditional distribution assumption, while the SSVAT can only approximate the distributions implicitly. An arousing result is that the mean test loss of the SSVAT decreases when that of the ground-truth model decreases, indicating the SSVAT can adapt to the ground-truth distribution and give reliable probability estimates. Besides, the SSVAT's best variant achieves performances close to those of the ground-truth model, showing it can sufficiently approximate the sparse distributions with a complicated architecture.

Appendix C

Hyperparameter Tuning

For the VAEs, the encoder has 1 input layer, whose width is 80. It has 2 output layers, which are the mean and the (log) standard deviation of the variational posterior respectively. Their widths are equal to the latent distribution dimension. The decoder has 1 input layer, whose width is equal to the latent distribution dimension. It has 1 (Poisson), 2 (NB or ZIP), 3 (ZINB) or 80 (categorical) output layers, depending on the conditional distribution. The output layer widths are all 80 unless using the categorical conditional distribution (where the widths depend on the maximum observed counts of different object classes). To reduce the computational cost, I assume the hidden layers of the encoder and those of the decoder are symmetric. For example, if the hidden layers for the encoder are [160, 80, 40], those for the decoder are [40, 80, 160]. Then the task reduces to tuning the hyperparameters of the encoder.

For the encoder, I consider 3 hyperparameters: the hidden layer depth, width and the latent distribution dimension. To decide the range of the latent distribution dimensions to test, I run a principal component analysis which uses a zero-inflated bivariate Poisson distribution [28] to model the pairwise covariance between dimensions on the training set. The result suggests 95% percent of the variance can be explained by 17 principal components. Since neural networks should be more expressive than this linear method, I consider 5 values for the latent distribution dimension: 4, 8, 12, 16, 20.

The hidden layer widths in the VAE encoder are usually between the input layer width and the output layer width, which gradually decrease with network depth for information compression. However, it is also known that the neural networks with wider hidden layers are more expressive [52]. Thus, I consider 3 possible hidden layer width values: 40, 80 and 160, which are half, the same and twice the input layer width respectively.

For the hidden layer depth, I consider 4 possible values: 0, 1, 2 and 3.

To reduce the computational cost, I apply 2 tricks. First, the hidden layers are combined in a non-increasing way. If the i^{th} (i = 1, 2) hidden layer's width is w, the $(i+1)^{\text{th}}$ hidden layer's width should not be larger than w. Second, I adjust the latent distribution dimension incrementally. If the best latent distribution dimension for the encoder with i (i = 0, 1, 2) hidden layers is d, I do not test the values larger than d for the encoder with $i+1$ hidden layers.

For each hyperparameter setting, I conduct experiments as those in the main text (see Section 3.3). I use the best mean validation loss from the best experiment as the standard to choose the hyperparameters. The main pattern[1] is that increasing the depth from 0 to 1 obviously improves the model performances, but further increasing the depth does not. And wider layers usually lead to better model performances.

For the VAE/SSVAT with expressive priors, I only tune the hyperparameters relevant to the prior. For the variants assuming different conditional distributions, I directly use the best network architectures based on the vanilla VAE hyperparameter tuning experiments. When using the VAMP prior, I consider 5 values for the number of pseudo-inputs: 2, 4, 6, 8 and 10. When using the MAF prior, I consider 5 values for the number of MADE layers: 1, 2, 3, 4 and 5. When using the AEA prior, I consider 5 values for the number of mixture components: 2, 4, 6, 8 and 10. For the AEA prior, the main pattern is the model performances decrease with the number of mixture components.

For the mixture model, the hyperparameter is the number of mixture components. I initially try values from 2 to 10 with a step size of 2, but find the model performances stably and obviously improve with more mixture components. Then I try values from 20 to 100 with a step size of 10, where the pattern does not change. Finally I try values from 200 to 1000 with a step size of 100, where the improvement is not obvious anymore. It indicates the data have a complex architecture which cannot be illustrated by few components.

For the SSVAT, I assume the number of encoder layers and that of the decoder layers are the same. Then I consider 2 hyperparameters: the number of encoder layers and the embedding dimension. For the former, I consider 3 values: 1, 2 and 3. For the latter, I consider 5 values: 4, 8, 12, 16 and 20. The main pattern is the model performances improve with more encoder layers and a larger embedding dimension, but the effect of the latter is much more obvious.

For the DAF, I consider 3 hyperparameters: the number of MADE layers, the width

[1]I do not mention "main pattern" if the pattern is not clear below.

of MADE layers (measured by the ratio between the hidden layer width and the input layer width), the depth of MADE layers. For the first hyperparameter, I consider 4 values: 1, 2, 3 and 4. For the second hyperparamter, I consider 2 values: 2 and 4. For the third hyperparameter, I consider 3 values: 1, 2 and 3. The main pattern is the model performances are insensitive to the hyperparameter values and are close to those of the model assuming the data follows a factorized categorical distribution. It indicates the model degenerates to the factorized categorical model under all hyperparameter settings.

For the DAD, I consider 2 hyperparameters: the hidden layer width (measured by the ratio between the hidden layer width and the input layer width) and depth. For the first hyperparameter, I consider 3 values: 1, 2 and 3. For the second hyperparameter, I consider 4 values: 1, 2, 3 and 4.