5th Symposium on Advances in Approximate Bayesian Inference, 2023 1–27

Variational Partitioning

Thomas M. Sutter[*] THOMAS.SUTTER@INF.ETHZ.CH
Alain Ryser[*] ALAIN.RYSER@INF.ETHZ.CH
Joram Liebeskind JORAM@LIEBESKIND.ORG
Julia E. Vogt JULIA.VOGT@INF.ETHZ.CH
ETH Zurich, Switzerland

Abstract

Partitioning a set of elements into an unknown number of mutually exclusive subsets is essential in many machine-learning problems. However, assigning elements to an unknown and discrete number of subsets is inherently non-differentiable, prohibiting end-to-end gradient-based optimization of parameters. We propose a novel two-step method for learning distributions over partitions, including a reparametrization trick, to allow the inclusion of partitions in variational inference tasks. Our method works by first inferring the number of elements per subset and then sequentially filling these subsets in an order learned in a second step. We highlight the versatility of our general-purpose approach on two different experiments: multitask learning and unsupervised conditional sampling.

1. Introduction

Partitioning a set of elements into subsets is a classical mathematical problem that attracted much interest over the last few decades. Formally, a partition over a given set is defined as a collection of non-overlapping subsets such that their union results in the original set. While there are many well-studied combinatorial partitioning problems (31; 8), recent advances in machine learning give rise to new challenges revolving around set-partitioning. In machine learning, partitioning a set of elements into different subsets is essential for many applications. However, due to the discrete nature of partitions, previous gradient-based methods tackle non-continuous loss functions and zero gradients almost everywhere with over-restrictive *i.i.d* assumptions (14).

Random partition models (RPM, 10) define a probability distribution over the space of partitions. In contrast to many modern machine learning methods, RPMs do not necessarily assume *i.i.d.* data points and can explicitly take the dependencies between samples into account. On the other hand, most existing RPMs either lack a reparameterization scheme or are computationally intractable for large datasets, prohibiting their use in modern machine learning pipelines (22; 29; 28).

In this work, we propose the differentiable random partition model (DRPM), a fully-differentiable relaxation for RPMs that allows reparametrizable sampling. DRPM follows a two-stage procedure: first, we model the number of elements per subset, and second, we learn an ordering of the elements with which we fill the elements into the subsets. DRPM enables the integration of partition models into modern machine learning frameworks and learning RPMs from data using stochastic optimization.

In experiments, we demonstrate the versatility of the proposed method by applying DRPM to weakly supervised learning and generative modeling.

[*] Equal contribution.

Related Work Following the proposition of the Gumbel-Softmax trick (GS, 13; 23), interest in research around continuous relaxations for discrete distributions and non-differentiable algorithms rose. The GS trick enabled the reparameterization of categorical distributions and their integration into gradient-based optimization pipelines. Based on the same trick, Xie and Ermon (35) describe a top-k elements selection procedure, and Sutter et al. (32) propose a differentiable formulation for the multivariate hypergeometric distribution. Multiple works on differentiable sorting procedures and permutation matrices have been proposed, e.g., (19; 30; 27). Grover et al. (9) described the distribution over permutation matrices $p(\pi)$ for a permutation matrix π using the Plackett-Luce (PL) distribution (29; 21). Previous non-differentiable works on RPMs include product partition models (10), species sampling models (28), and model-based clustering approaches (2).

2. Method

We want to partition n elements $[n] = \{1, \dots, n\}$ into K subsets $\{\mathcal{S}_1, \dots, \mathcal{S}_K\}$ where K is *a priori* unknown. For a partition $\rho = (\mathcal{S}_1, \dots, \mathcal{S}_K)$ to be valid, it must hold that $\mathcal{S}_1 \cup \dots \cup \mathcal{S}_K = [n]$ and $\mathcal{S}_i \cap \mathcal{S}_j = \emptyset, \ \forall\, i \neq j$. Put differently, every element i has to be assigned to precisely one subset $\mathcal{S}_k$. We denote the size of the k-th subset $\mathcal{S}_k$ as $n_k = |\mathcal{S}_k|$. Alternatively, we describe a partition ρ as an assignment matrix $Y = [\boldsymbol{y}_1, \dots, \boldsymbol{y}_K]^T \in \{0,1\}^{K \times n}$. Every row $\boldsymbol{y}_k \in \{0,1\}^{1 \times n}$ is a multi-hot vector, where $\boldsymbol{y}_{ki} = 1$ assigns element i to subset $\mathcal{S}_k$.

In this work, we propose a new two-stage procedure for partition models. The proposed formulation separately infers the number of elements per subset n_k and the assignment of elements to subsets $\mathcal{S}_k$ by inducing an order on the n elements and filling $\mathcal{S}_1, ..., \mathcal{S}_K$ sequentially in this order. See Figure 2 in the appendix for an example.

Definition 1 (Two-stage partition model) *Let $\boldsymbol{n} = [n_1, \dots, n_K] \in \mathbb{N}_0^K$ be the subset sizes in ρ, with $\mathbb{N}_0$ the set of natural numbers including 0 and $\sum_{k=1}^K n_k = n$, where n is the total number of elements. Let $\pi \in \{0,1\}^{n \times n}$ be a permutation matrix that defines an order over the n elements. We define the two-stage partition model of n elements into K subsets as an assignment matrix $Y = [\boldsymbol{y}_1, \dots, \boldsymbol{y}_K]^T \in \{0,1\}^{K \times n}$ with*

$$\boldsymbol{y}_k = \sum_{i=\nu_k+1}^{\nu_k+n_k} \boldsymbol{\pi}_i, \quad \textit{where} \ \ \nu_k = \sum_{\iota=1}^{k-1} n_\iota \tag{1}$$

such that $Y = \left[\{\boldsymbol{y}_k \mid n_k > 0\}_{k=1}^K\right]^T$.

Note that in contrast to previous works on partition models (24), we allow $\mathcal{S}_k$ to be the empty set $\emptyset$. Hence, K defines the maximum number of possible subsets, not the effective number of non-empty subsets.

To model the order of the elements, we use a permutation matrix $\pi = [\boldsymbol{\pi}_1, \dots, \boldsymbol{\pi}_n]^T \in \{0,1\}^{n \times n}$ which is a square matrix where every row and column sums to 1. This doubly-stochastic property of all permutation matrices π (25) thus ensures that the columns of Y remain one-hot vectors. At the same time, its rows correspond to n_k-hot vectors $\boldsymbol{y}_k$ in Definition 1 and therefore serve as subset assignment vectors. We provide the proof for the following corollary in Appendix B.1.1.

Corollary 2 *A two-stage partition model Y, which follows Definition 1, is a valid partition.*

2.1. Differentiable Random Partition Models

An RPM $p(Y)$ defines a probability distribution over partitions Y. In this section, we derive how to extend the two-stage procedure from Definition 1 to the probabilistic setting to create a two-stage RPM. To derive the two-stage RPM's probability distribution $p(Y)$, we need to model distributions over $\boldsymbol{n}$ and π. We choose the Multivariate Hypergeometric (MVHG) distribution $p(\boldsymbol{n};\boldsymbol{\omega})$ (see Appendix A.1) and the PL distribution $p(\pi;\boldsymbol{s})$ (see Appendix A.3). For the remainder of this paper, we denote $p(Y)$ as $p(Y;\boldsymbol{\omega},\boldsymbol{s})$ to indicate dependence on the MVHG parameter $\boldsymbol{\omega}$ and PL parameter $\boldsymbol{s}$. In Appendix B we derive the following proposition:

Proposition 3 (DRPM) *The probability density function $p(Y;\boldsymbol{\omega},\boldsymbol{n})$ of the differentiable RPM is given by*

$$p(Y;\boldsymbol{\omega},\boldsymbol{s}) = p(\boldsymbol{y}_1,\dots,\boldsymbol{y}_K;\boldsymbol{\omega},\boldsymbol{s}) = p(\boldsymbol{n};\boldsymbol{\omega})\sum_{\pi\in\Pi_Y} p(\pi;\boldsymbol{s}) \tag{2}$$

where $\Pi_Y = \{\pi : \boldsymbol{y}_k = \sum_{i=\nu_k+1}^{\nu_k+n_k} \boldsymbol{\pi}_i, k=1,\dots,K\}$.

2.1.1. Approximating the distribution over RPMs

The number of permutations per subset $|\Pi_{\boldsymbol{y}_k}|$ scales factorially with the subset size n_k, i.e. $|\Pi_{\boldsymbol{y}_k}| = n_k!$. Consequently, the number of valid permutation matrices $|\Pi_Y|$ is given as a function of $\boldsymbol{n}$, i.e.

$$|\Pi_Y| = \prod_{k=1}^{K} |\Pi_{\boldsymbol{y}_k}| = \prod_{k=1}^{K} n_k! \tag{3}$$

Although Proposition 3 describes a well-defined distribution for $p(Y;\boldsymbol{\omega},\boldsymbol{s})$, it is in general computationally intractable due to Equation (3). In practice, we thus approximate $p(Y;\boldsymbol{\omega},\boldsymbol{s})$ using the following Lemma (proof in Appendix B.3.1).

Lemma 4 *$p(Y;\boldsymbol{\omega},\boldsymbol{s})$ can be upper and lower bounded as follows*

$$\forall\pi\in\Pi_Y: \; p(Y;\boldsymbol{\omega},\boldsymbol{s}) \geq p(\boldsymbol{n};\boldsymbol{\omega})p(\pi;\boldsymbol{s}) \tag{4}$$

$$p(Y;\boldsymbol{\omega},\boldsymbol{s}) \leq |\Pi_Y| p(\boldsymbol{n};\boldsymbol{\omega}) \max_{\pi} p(\pi;\boldsymbol{s}) \tag{5}$$

2.2. Sampling partitions from the DRPM

To sample a partition from our DRPM, we use the two methods from Appendices A.1 and A.3, which introduced differentiable and reparameterizable distributions for $p(\pi;\boldsymbol{s})$ and $p(\boldsymbol{n};\boldsymbol{\omega})$ respectively (9; 32). Sampling a partition Y thus works by first sampling $\pi\sim p(\pi;\boldsymbol{s})$ and $\boldsymbol{n}\sim p(\boldsymbol{n};\boldsymbol{\omega})$ from the PL and MVHG distributions respectively. We can then calculate $Y = f(\pi,\boldsymbol{n})$ according to Definition 1 by summing the rows of π according to

Table 1: We evaluate the learned latent representations of the four methods (LabelVAE, AdaVAE, HGVAE, DRPMVAE) in the weakly-supervised experiment with respect to the shared (S) and independent (I) generative factors. We do this by fitting linear classifiers on the shared and independent dimensions of the representation, predicting the respective generative factors. We report the results in adjusted balanced accuracy (32) across five seeds.

	$n_s = 0$	$n_s = 1$		$n_s = 3$		$n_s = 5$	
	I	S	I	S	I	S	I
Label	$0.14_{\pm 0.01}$	$0.19_{\pm 0.03}$	$0.16_{\pm 0.01}$	$0.10_{\pm 0.00}$	$0.23_{\pm 0.01}$	$0.34_{\pm 0.00}$	$0.00_{\pm 0.00}$
Ada	$0.12_{\pm 0.01}$	$0.19_{\pm 0.01}$	$0.15_{\pm 0.01}$	$0.10_{\pm 0.03}$	$0.22_{\pm 0.02}$	$0.33_{\pm 0.03}$	$0.00_{\pm 0.00}$
HG	$0.18_{\pm 0.01}$	$0.22_{\pm 0.05}$	$0.19_{\pm 0.01}$	$0.08_{\pm 0.02}$	$0.28_{\pm 0.01}$	$0.28_{\pm 0.01}$	$0.01_{\pm 0.00}$
DRPM (Ours)	$\mathbf{0.26}_{\pm 0.02}$	$\mathbf{0.39}_{\pm 0.07}$	$\mathbf{0.2}_{\pm 0.01}$	$\mathbf{0.15}_{\pm 0.01}$	$\mathbf{0.29}_{\pm 0.02}$	$\mathbf{0.42}_{\pm 0.03}$	$0.01_{\pm 0.00}$

$\boldsymbol{n}$. Using this two-stage procedure, we can infer and resample partitions in a differentiable and reparameterizable way. Additionally, due to Proposition 3 and Lemma 4, we are also able to efficiently estimate $p(Y; \boldsymbol{\omega}, \boldsymbol{s})$ for a given DRPM sample Y.

In summary, we introduce an efficient model to deterministically and probabilistically learn partitions in an end-to-end fashion. In contrast to previous RPMs, which often need exponentially many distribution parameters (29), the proposed DRPM needs only $(n + K)$ parameters to create an RPM for n elements: the score parameters $\boldsymbol{s} \in \mathbb{R}_+^n$ and the group importance parameters $\boldsymbol{\omega} \in \mathbb{R}_+^K$. Our DRPM enables us to integrate partition models in any gradient-based optimization pipeline. In the following experiments, we present how to use the DRPM with both deterministic and probabilistic models. Additionally, we demonstrate how our method enables us to learn variational posterior approximations and priors using Stochastic Gradient Variational Bayes (SGVB, 17) optimization in variational inference models.

3. Experiments

We demonstrate the versatility and effectiveness of the proposed DRPM in two different experiments. First, we show how to use DRPM in weakly-supervised learning where we are interested in finding shared and independent latent generative factors. Second, we introduce a novel variational autoencoder that incorporates the DRPM to learn a conditional generative model in an unsupervised fashion.

3.1. Weakly-Supervised Learning

Data modalities that are not collected as *i.i.d.* samples, such as consecutive frames in a video, provide a weak-supervision signal for generative models and representation learning (32). Compared to learning from single frames, where we are interested in learning meaningful representations (3), here, we are also interested in discovering the relationship between the coupled, non-*i.i.d.* samples. Assuming that the data is generated from underlying generative factors, weak supervision implies that certain factors are shared between coupled pairs while others are independent. The supervision is only weak because we neither know the underlying generative factors nor the number of shared and independent factors. In such a setting, our DRPM not only enables us to infer the number of shared and independent generative factors but also lets us assign latent factors to be either shared or independent.

In this experiment, we use paired frames $\boldsymbol{X} = [\boldsymbol{x}_1, \boldsymbol{x}_2]$ from the *mpi3d* toy dataset (7). Every pair of frames shares a subset of their seven generative factors. The model maximizes an evidence lower bound on the marginal log-likelihood of images through a VAE (17). We use the code from Locatello et al. (20) and follow the evaluation in Sutter et al. (32). We refer to Appendix C.1 for more details on the setup and baseline methods of this experiment.

For DRPMVAE, we model the distribution of shared and independent latent factors as RPM using the proposed DRPM $p(Y; \boldsymbol{\omega}, \boldsymbol{s})$. We add a posterior approximation of the form $q(Y; \boldsymbol{\omega}(\boldsymbol{X}), \boldsymbol{s}(\boldsymbol{X}))$ where the notation $\boldsymbol{\omega}(\boldsymbol{X})$ and $\boldsymbol{s}(\boldsymbol{X})$ implies that the distribution parameters are inferred from data $\boldsymbol{X}$, and additionally a prior distribution of the form $p(Y; \boldsymbol{\omega}_p, \boldsymbol{s}_p)$. The ELBO we optimize when assuming this generative process is given by:

$$\begin{aligned}\log p(\boldsymbol{X}) \geq & \mathbb{E}_{q(\boldsymbol{z},Y|\boldsymbol{X})}\left[\log p(\boldsymbol{x}_1 \mid \boldsymbol{z}_s, \boldsymbol{z}_1)\right] + \mathbb{E}_{q(\boldsymbol{z},Y|\boldsymbol{X})}\left[\log p(\boldsymbol{x}_2 \mid \boldsymbol{z}_s, \boldsymbol{z}_2)\right] \\ & - \mathbb{E}_{q(\boldsymbol{z},Y|\boldsymbol{X})}\left[\log \frac{q(\boldsymbol{z}_s, \boldsymbol{z}_1, \boldsymbol{z}_2 \mid Y, \boldsymbol{X})}{p(\boldsymbol{z}_s, \boldsymbol{z}_1, \boldsymbol{z}_2)}\right] - \mathbb{E}_{q(\boldsymbol{z},Y|\boldsymbol{X})}\left[\log \frac{q(Y; \boldsymbol{\omega}(\boldsymbol{X}), \boldsymbol{s}(\boldsymbol{X}))}{p(Y; \boldsymbol{\omega}_p, \boldsymbol{s}_p)}\right] \end{aligned} \quad (6)$$

We refer to Appendix C.1 for details on the implementation.

We evaluate all methods according to how well they partition the latent representations into shared and independent factors (Table 1). Because we have access to the data-generating process, we can control the number of shared n_s and independent n_i factors. We compare the methods on four different weakly-supervised datasets with $n_s \in \{0, 1, 3, 5\}$. In Table 1, we see a considerable performance improvement compared to previous work when assessing the learned latent representations. We assume this to be due to its ability to not only estimate the subset sizes of latent and shared factors like HGVAE but also to learn assigning latent dimensions to corresponding shared or independent representations. Thus, DRPMVAE can dynamically learn more meaningful representations of both the shared and independent subspaces for all dataset versions. See Appendix C.1 for more results. DRPMVAE provides empirical evidence of how RPMs can help with specific weakly-supervised learning tasks where we are interested in maximizing the data likelihood while also learning representations that capture the relation between coupled data samples. Additionally, we can explicitly model the data-generating process in a theoretically grounded fashion instead of relying on heuristics.

3.2. Generative Model

In a second experiment, we explore whether we can use the DRPM to learn a generative model for unsupervised conditional generation in a variational fashion. The main idea is to learn latent approximate posterior distributions that are close to class conditional priors by partitioning the approximate posteriors according to an RPM. More specifically, we assume that each sample $\boldsymbol{x_i}$ of a dataset $\mathcal{X}$ is generated by a latent vector $\boldsymbol{z_i}$. Instead of assuming a single Gaussian prior $\mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\sigma}\mathbb{I})$ for all latent vectors like in traditional variational autoencoders (VAE, 17), we assume every z_i to be sampled from one of K different latent Gaussian distributions $\mathcal{N}(\boldsymbol{\mu}_y, \boldsymbol{\sigma}_y\mathbb{I}), y = 1, \ldots, K$. The assignments y_i of all $\boldsymbol{z_i}$ to their respective clusters are then distributed according to an RPM, potentially resulting in dependencies between latent vectors z_i. In the following, let $\mathcal{Z} = \{z_1, ..., z_N\}$, and $Y = \{y_1, ..., y_N\}$ contain the respective latent vectors and cluster assignments for each sample of a given dataset $\mathcal{X} = \{x_1, ..., x_N\}$ with N samples. The generative process is summarized as follows: First, we sample the cluster assignments Y from an RPM, i.e., $Y \sim DRPM(\boldsymbol{\omega}, \boldsymbol{s})$. Given Y, we can sample the latent variables $\mathcal{Z}$, where $\boldsymbol{z_i} \sim \mathcal{N}(\boldsymbol{\mu_{y_i}}, \boldsymbol{\sigma_{y_i}^T}\mathbb{I}_l)$, $\boldsymbol{z_i} \in \mathbb{R}^l$. Finally, we sample $\mathcal{X}$, where $\boldsymbol{x_i} \sim \mathcal{N}(\boldsymbol{\mu}(\boldsymbol{z_i}), \boldsymbol{\sigma}(\boldsymbol{z_i})^T\mathbb{I}_d)$ if $\boldsymbol{x_i} \in \mathbb{R}^d$ or $\boldsymbol{x_i} \sim Ber(\boldsymbol{\mu}(\boldsymbol{z_i}))$ if $\boldsymbol{x_i} \in \{0,1\}^d$.

Assuming this generative process, we derive the following evidence lower bound (ELBO) for $p(\mathcal{X})$:

$$\mathcal{L}_{ELBO} = \sum_{\boldsymbol{x} \in \mathcal{X}} \mathbb{E}_{q(\boldsymbol{z}|\boldsymbol{x})}\left[\log p(\boldsymbol{x}|\boldsymbol{z})\right] - \sum_{\boldsymbol{x} \in \mathcal{X}} \mathbb{E}_{q(Y|\mathcal{X})}\left[KL[q(\boldsymbol{z}|\boldsymbol{x})||p(\boldsymbol{z}|Y)]\right] - KL[q(Y|\mathcal{X})||p(Y)]$$

Note that computing $KL[q(Y|\mathcal{X})||p(Y)]$ directly is computationally intractable, and we need to upper bound it using Theorem 4, i.e.

$$KL[q(Y|\mathcal{X})||p(Y)] \leq \mathbb{E}_{q(Y|\mathcal{X})}\left[\log \frac{|\Pi_Y| \cdot q(\boldsymbol{n}; \boldsymbol{\omega}(\mathcal{X}))}{p(\boldsymbol{n}; \boldsymbol{\omega})p(\pi_Y; \boldsymbol{s})}\right] + \log\left(\max_{\pi} q(\pi; \boldsymbol{s}(\mathcal{X}))\right),$$

where π_Y is any $\pi \in \Pi_Y$.

We train and qualitatively evaluate our model on MNIST (18) and present the generative capabilities of our method in Figure 1, where we sample a partition with $n = 8$ from the prior of a trained generative DRPM model. Interestingly, our model seems to pick up on the different digit distributions, learning separate priors for each digit and allowing us to sample from different classes of the distribution in an unsupervised fashion. In practice, this could help with exploring the different modes of a given distribution if we do not have access to labels.

We provide more samples and implementation details in Appendix C.3.

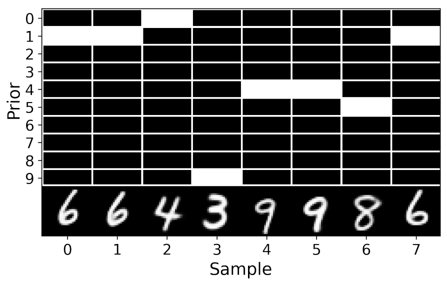


Figure 1: A sample drawn from a trained DRPM generative model. On top is the sampled partition with the cluster assignments, and at the bottom are the corresponding prior samples.

4. Conclusion

In this work, we proposed the DRPM, a novel differentiable random partition model. Its differentiable and reparameterizable formulation enables the integration of RPMs in modern probabilistic gradient-based optimization frameworks. In addition, the proposed two-stage formulation separately controls the number of elements per subset and the assignment of elements to subsets. We show the versatility of the proposed DRPM by applying it to two vastly different experiments, where we could easily use it as a plug-and-play module. In the future, we want to explore whether we can use the conditional generation framework to achieve state-of-the-art clustering performance by investigating the learned approximate posteriors more closely. Further, we want to explore whether we can also use the DRPM as a deterministic module in certain settings.

References

[1] Matej Balog, Nilesh Tripuraneni, Zoubin Ghahramani, and Adrian Weller. Lost relatives of the Gumbel trick. In *International Conference on Machine Learning*, pages 371–379, 2017.

[2] Christopher M Bishop and Markus Svensen. Robust Bayesian Mixture Modelling. *To appear in the Proceedings of ESANN*, page 1, 2004.

[3] Diane Bouchacourt, Ryota Tomioka, and Sebastian Nowozin. Multi-level variational autoencoder: Learning disentangled representations from grouped observations. In *Thirty-Second AAAI Conference on Artificial Intelligence*, 2018.

[4] Jean Chesson. A non-central multivariate hypergeometric distribution arising from biased sampling with application to selective predation. *Journal of Applied Probability*, 13(4):795–797, 1976.

[5] Ronald A Fisher. The logic of inductive inference. *Journal of the royal statistical society*, 98(1):39–82, 1935.

[6] Agner Fog. Calculation methods for Wallenius' noncentral hypergeometric distribution. *Communications in Statistics—Simulation and Computation®*, 37(2):258–273, 2008.

[7] Muhammad Waleed Gondal, Manuel Wuthrich, Djordje Miladinovic, Francesco Locatello, Martin Breidt, Valentin Volchkov, Joel Akpo, Olivier Bachem, Bernhard Schölkopf, and Stefan Bauer. On the Transfer of Inductive Bias from Simulation to the Real World: a New Disentanglement Dataset. In *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc., 2019.

[8] RL Graham, DE Knuth, O Patashnik, S Liu Computers in Physics, and undefined 1989. Concrete mathematics: a foundation for computer science. *aip.scitation.org*, 3(5):165, 1989. doi: 10.1063/1.4822863. URL https://aip.scitation.org/doi/pdf/10.1063/1.4822863.

[9] Aditya Grover, Eric Wang, Aaron Zweig, and Stefano Ermon. Stochastic Optimization of Sorting Networks via Continuous Relaxations. In *International Conference on Learning Representations*, 2019. URL https://openreview.net/forum?id=H1eSS3CcKX.

[10] J A Hartigan. Partition models. *Communications in Statistics - Theory and Methods*, 19(8):2745–2756, 1990. doi: 10.1080/03610929008830345.

[11] I Higgins, L Matthey, A Pal, C Burgess, and X Glorot. beta-vae: Learning basic visual concepts with a constrained variational framework. 2016. URL https://openreview.net/forum?id=Sy2fzU9gl.

[12] Haruo Hosoya. A simple probabilistic deep generative model for learning generalizable disentangled representations from grouped data. *CoRR*, abs/1809.0, 2018.

[13] Eric Jang, Shixiang Gu, and Ben Poole. Categorical reparameterization with gumbel-softmax. *arXiv preprint arXiv:1611.01144*, 2016.

[14] Zhuxi Jiang, Yin Zheng, Huachun Tan, Bangsheng Tang, and Hanning Zhou. Variational Deep Embedding: An Unsupervised and Generative Approach to Clustering. *IJCAI International Joint Conference on Artificial Intelligence*, 0:1965–1972, 11 2016. ISSN 10450823. doi: 10.48550/arxiv.1611.05148. URL https://arxiv.org/abs/1611.05148v3.

[15] Mark E Johnson. *Multivariate statistical simulation: A guide to selecting and generating continuous multivariate distributions*, volume 192. John Wiley \& Sons, 1987.

[16] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[17] Diederik P Kingma and Max Welling. Auto-Encoding Variational Bayes. In *International Conference on Learning Representations*, 2014.

[18] Yann LeCun, Corinna Cortes, and C J Burges. Gradient-based learning applied to document recognition. In *Proceedings of the IEEE*, volume 86, pages 2278–2324, 1998.

[19] Scott W Linderman, Gonzalo E Mena, Hal James Cooper, Liam Paninski, and John P Cunningham. Reparameterizing the Birkhoff Polytope for Variational Permutation Inference. 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 1618–1627. PMLR, 2018. URL http://proceedings.mlr.press/v84/linderman18a.html.

[20] Francesco Locatello, Ben Poole, Gunnar Rätsch, Bernhard Schölkopf, Olivier Bachem, and Michael Tschannen. Weakly-supervised disentanglement without compromises. In *International Conference on Machine Learning*, pages 6348–6359. PMLR, 2020.

[21] R Duncan Luce. *Individual choice behavior: A theoretical analysis.* Courier Corporation, 1959.

[22] J MacQueen. Classification and analysis of multivariate observations. In *5th Berkeley Symp. Math. Statist. Probability*, pages 281–297, 1967.

[23] C Maddison, A Mnih, and Y Teh. The concrete distribution: A continuous relaxation of discrete random variables. In *International Conference on Learning Representations*, 2017.

[24] Toufik Mansour and Matthias Schork. *Commutation relations, normal ordering, and Stirling numbers.* CRC Press Boca Raton, 2016.

[25] Marvin Marcus. Some Properties and Applications of Doubly Stochastic Matrices. *The American Mathematical Monthly*, 67(3):215–221, 1960. ISSN 00029890, 19300972. URL http://www.jstor.org/stable/2309679.

[26] Gonzalo E Mena, David Belanger, Scott W Linderman, and Jasper Snoek. Learning Latent Permutations with Gumbel-Sinkhorn Networks. In *6th International Conference*

on Learning Representations, {ICLR} 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Conference Track Proceedings. OpenReview.net, 2018. URL https://openreview.net/forum?id=Byt3oJ-0W.

[27] Felix Petersen, Christian Borgelt, Hilde Kuehne, and Oliver Deussen. Monotonic Differentiable Sorting Networks. In *International Conference on Learning Representations*, 2021.

[28] Jim Pitman. Some Developments of the Blackwell-Macqueen URN Scheme. *Lecture Notes-Monograph Series*, 30:245–267, 1996. ISSN 07492170. URL http://www.jstor.org/stable/4355949.

[29] Robin L Plackett. The analysis of permutations. *Journal of the Royal Statistical Society: Series C (Applied Statistics)*, 24(2):193–202, 1975.

[30] Sebastian Prillo and Julian Eisenschlos. SoftSort: A Continuous Relaxation for the argsort Operator. 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 7793–7802. PMLR, 2020. URL http://proceedings.mlr.press/v119/prillo20a.html.

[31] Gian-Carlo Rota. The Number of Partitions of a Set. *The American Mathematical Monthly*, 71(5):498, 5 1964. ISSN 00029890. doi: 10.2307/2312585.

[32] Thomas M Sutter, Laura Manduchi, Alain Ryser, and Julia E Vogt. Learning Group Importance using the Differentiable Hypergeometric Distribution. 2022.

[33] Louis L Thurstone. A law of comparative judgment. In *Scaling*, pages 81–92. Routledge, 1927.

[34] Kenneth T Wallenius. Biased sampling; the noncentral hypergeometric probability distribution. Technical report, Stanford Univ Ca Applied Mathematics And Statistics Labs, 1963.

[35] Sang Michael Xie and Stefano Ermon. Reparameterizable subset sampling via continuous relaxations. *arXiv preprint arXiv:1901.10517*, 2019.

[36] John I Yellott Jr. The relationship between Luce's choice axiom, Thurstone's theory of comparative judgment, and the double exponential distribution. *Journal of Mathematical Psychology*, 15(2):109–144, 1977.

Appendix A. Preliminaries

A.1. Probability distribution over subset sizes

The multivariate non-central hypergeometric distribution (MVHG) describes sampling without replacement and allows to skew the importance of groups with an additional importance parameter $\boldsymbol{\omega}$ (5; 34; 4). The MVHG is an urn model and is described by the number of different groups $K \in \mathbb{N}$, the number of elements in the urn of every group $\boldsymbol{m} = [m_1, \dots, m_K] \in \mathbb{N}^K$, the total number of elements in the urn $\sum_{k=1}^{K} m_k \in \mathbb{N}$, the number of samples to draw from the urn $n \in \mathbb{N}_0$, and the importance factor for every group $\boldsymbol{\omega} = [\omega_1, \dots, \omega_K] \in \mathbb{R}_{0+}^K$ (15).

$$p(\boldsymbol{n}; \boldsymbol{\omega}, \boldsymbol{m}) = \frac{1}{P_0} \prod_{k=1}^{K} \binom{m_k}{n_k} \omega_k^{n_k} \tag{7}$$

$$\text{where} \quad P_0 = \sum_{(\eta_1, \dots, \eta_K) \in S} \prod_{\kappa=1}^{K} \binom{m_\kappa}{\eta_\kappa} \omega_\kappa^{\eta_\kappa} \tag{8}$$

Here, $S = \{\boldsymbol{n} \in \mathbb{N}_0^K : \forall k \quad n_k \leq m_k, \sum_{k=1}^{K} n_k = n\}$ denotes the support S of the probability mass function.

Sutter et al. (32) introduced a differentiable reparameterization for the MVHG that makes use of the GS trick (13; 23) and enables the learning of group importance $\boldsymbol{\omega}$ using gradient-based optimization. Hence, the differentiable MVHG $p(\boldsymbol{n}; \boldsymbol{\omega}, \boldsymbol{m})$ allows us to model dependencies between different elements of a set since drawing one element from the urn influences the probability of drawing one of the remaining ones, creating interdependence between samples. Note that here, we assume $\forall m_i \in \boldsymbol{m} : m_i = n$ and thus use the shorthand $p(\boldsymbol{n}; \boldsymbol{\omega})$ to denote the density of the MVHG. The following serves as a quick recap on the MVHG and is largely based on Sutter et al. (32).

A.2. Hypergeometric Distribution

Suppose we have an urn with marbles in different colors. Let $K \in \mathbb{N}$ be the number of different classes or groups (e.g. marble colors in the urn), $\boldsymbol{m} = [m_1, \dots, m_K] \in \mathbb{N}^K$ describe the number of elements per class (e.g. marbles per color), $N = \sum_{k=1}^{K} m_K$ be the total number of elements (e.g. all marbles in the urn) and $n \in \{0, \dots, N\}$ be the number of elements (e.g. marbles) to draw. Then, the multivariate hypergeometric distribution describes the probability of drawing $\boldsymbol{n} = [n_1, \dots, n_K] \in \mathbb{N}^K$ marbles by sampling without replacement such that $\sum_{k=1}^{K} n_k = n$, where n_k is the number of drawn marbles of class k.

In the literature, two different versions of the noncentral hypergeometric distribution exist, Fisher's (5) and Wallenius' (34; 4) distribution. Sutter et al. (32) restrict themselves to Fisher's noncentral hypergeometric distribution due to limitations of the latter (6). Hence, we will also talk solely about Fisher's noncentral hypergeometric distribution.

Definition 5 (Multivariate Fisher's Noncentral Hypergeometric Distribution (5)) *A random vector* $\boldsymbol{X}$ *follows Fisher's noncentral multivariate distribution, if its joint proba-*

bility mass function is given by

$$P(\boldsymbol{N} = \boldsymbol{n}; \boldsymbol{\omega}) = p(\boldsymbol{n}; \boldsymbol{\omega}) = \frac{1}{P_0} \prod_{k=1}^{K} \binom{m_k}{n_k} \omega_k^{n_k} \tag{9}$$

$$\textit{where} \;\; P_0 = \sum_{(\eta_1,\ldots,\eta_K)\in\mathcal{S}} \prod_{k=1}^{K} \binom{m_k}{\eta_k} \omega_k^{\eta_k} \tag{10}$$

The support S of the PMF is given by $S = \{\boldsymbol{n} \in \mathbb{N}^K : \forall k \quad n_k \leq m_k, \sum_{k=1}^{K} n_k = n\}$ and $\binom{n}{k} = \frac{n!}{k!(n-k)!}$.

The class importance $\boldsymbol{\omega}$ is a crucial modeling parameter in applying the noncentral hypergeometric distribution (see (4)).

A.2.1. Differentiable MVHG

Their reparameterizable sampling for the differentiable MVHG consists of three parts:

1. Reformulate the multivariate distribution as a sequence of interdependent and conditional univariate hypergeometric distributions.
2. Calculate the probability mass function of the respective univariate distributions.
3. Sample from the conditional distributions utilizing the Gumbel-Softmax trick.

Following the chain rule of probability, the MVHG distribution allows for sequential sampling over classes k. Every step includes a merging operation, which leads to biased samples compared to groundtruth non-differentiable sampling with equal class weights $\boldsymbol{\omega}$. Given that we intend to use the differentiable MVHG in settings where we want to learn the unknown class weights, we do not expect a negative effect from this sampling procedure. For details on how to merge the MVHG into a sequence of unimodal distributions, we refer to Sutter et al. (32).

The probability mass function calculation is based on unnormalized log-weights, which are interpreted as unnormalized log-weights of a categorical distribution. The interpretation of the class-conditional unimodal hypergeometric distributions as categorical distributions allows applying the Gumbel-Softmax trick (13; 23). Following the use of the Gumbel-Softmax trick, the class-conditional version of the hypergeometric distribution is differentiable and reparameterizable. Hence, the MVHG has been made differentiable and reparameterizable as well. Again, for details we refer to the original paper (32).

A.3. Probability distribution over Permutation Matrices

In addition to the distribution over subset sizes, we are also interested in describing the distribution over permutation matrices $p(\pi) \in \{0,1\}^{n\times n}$ that impose an ordering over a set with n elements. We assume $p(\pi)$ to be parametrized by scores $\boldsymbol{s} \in \mathbb{R}_+^n$, where each element i corresponds to s_i. A sample π from the permutation distribution $p(\pi; \boldsymbol{s})$ then corresponds to the order that results from sorting $\tilde{\boldsymbol{s}}$, which is a perturbed version of $\boldsymbol{s}$ (33). Specifically, sorting $\tilde{\boldsymbol{s}}$ in decreasing order leads to a permutation of the original n elements, which we represent as a permutation matrix π. Hence, resampling scores $\boldsymbol{s}$ enables the resampling of permutation matrices π. The resulting distribution is a Plackett-Luce (PL) distribution (21; 29) if and only if the scores $\boldsymbol{s}$ are perturbed with noise drawn from Gumbel distributions with identical scales (36; 9).

Proposition 6 (Distribution over Permutation Matrices (9)) *For each item i, let g_i denote Gumbel noise $g_i \sim$ Gumbel$(0, \beta)$ for fixed scale β. Let $\tilde{\boldsymbol{s}}$ be the perturbated scores such that $\tilde{s}_i = \beta \log s_i + g_i$. The probability $p(\pi; \boldsymbol{s})$ is then given by*

$$\begin{aligned} p(\pi; \boldsymbol{s}) =& p((\pi\tilde{\boldsymbol{s}})_1 \geq \cdots \geq (\pi\tilde{\boldsymbol{s}})_n) && (11)\\ =& \frac{(\pi\boldsymbol{s})_1}{Z} \frac{(\pi\boldsymbol{s})_2}{Z - (\pi\boldsymbol{s})_1} \cdots \frac{(\pi\boldsymbol{s})_n}{Z - \sum_{j=1}^{n-1}(\pi\boldsymbol{s})_j} && (12)\end{aligned}$$

where π is a permutation matrix and $Z = \sum_{i=1}^n s_i$.

Proof We take the proof from Grover et al. (9), which follows a result from Yellott Jr (36). We only provide a proof sketch and refer the reader to Yellott Jr (36) for more details.

Consider random variables $\{X_i\}_{i=1}^n$ such that $X_i \sim \exp(s_i)$. We may prove by induction a generalization of the memoryless property:

$$\begin{aligned} q(X_1 \leq \cdots \leq X_n \mid x \leq \min_i X_i) = & \int_0^\infty q(x \leq X_1 \leq X_1 + t \mid x \leq \min_i X_i) && (13)\\ & \cdot q(X_2 \leq \cdots \leq X_n \mid x + t \leq \min_{i\geq 2} X_i)dt && (14)\\ = & \int_0^\infty q(0 \leq X_1 \leq t) && (15)\\ & \cdot q(X_2 \leq \cdots \leq X_n \mid x + t \leq \min_{i\geq 2} X_i)dt && (16)\end{aligned}$$

If we assume as inductive hypothesis that

$$q(X_2 \leq \cdots \leq X_n \mid x + t \leq \min_{i\geq 2} X_i) = q(X_2 \leq \cdots \leq X_n \mid t \leq \min_{i\geq 2} X_i),$$

we complete the induction as:

$$\begin{aligned} q(X_1 \leq \cdots \leq X_n \mid x \leq \min_i X_i) = & \int_0^\infty q(0 \leq X_1 \leq t) q(X_2 \leq \cdots \leq X_n \mid t \leq \min_{i\geq 2} X_i)dt && (17)\\ =& q(X_1 \leq X_2 \leq \cdots \leq X_n \mid 0 \leq \min_i X_i) && (18)\end{aligned}$$

It follows from a familiar property of argmin of exponential distributions that:

$$\begin{aligned} q(X_1 \leq \cdots \leq X_n \mid x \leq \min_i X_i) =& q(X_1 \leq \min_i X_i) q(X_2 \leq \cdots \leq X_n \mid X_i \leq \min_i X_i) && (19)\\ =& \frac{s_i}{Z} q(X_2 \leq \cdots \leq X_n \mid X_i \leq \min_i X_i) && (20)\\ =& \frac{s_i}{Z} \int_0^\infty q(X_1 = x) q(X_2 \leq \cdots \leq X_n \mid x \leq \min_{i\geq 2} X_i)dx && (21)\\ =& \frac{s_i}{Z} q(X_2 \leq \cdots \leq X_n) && (22)\end{aligned}$$

and by another induction, we have

$$q(X_1 \leq \dots \leq X_n) = \prod_{i=}^{n} \frac{s_i}{Z - \sum_{j=1}^{i-1} s_j}. \tag{23}$$

Finally, following the argument of Balog et al. (1), we apply the strictly decreasing function $g(x) = -\beta \log x$ to this identity, which from the definition of the Gumbel distribution implies

$$q(\tilde{s}_1 \leq \dots \leq \tilde{s}_n) = \prod_{i=}^{n} \frac{s_i}{Z - \sum_{j=1}^{i-1} s_j}. \tag{24}$$

■

To compute $p(\pi; \boldsymbol{s})$, Grover et al. (9) introduce a differentiable sorting function $f_\pi(\tilde{\boldsymbol{s}}) = \text{sort}(\tilde{\boldsymbol{s}})$ that orders the resampled scores $\tilde{\boldsymbol{s}}$ in descending order.

Here, we summarise the findings from Grover et al. (9) on how to construct such a differentiable sorting operator. As already mentioned in Section 1, there are multiple works on the topic (30; 27; 26), but we restrict ourselves to the work of Grover et al. (9) as we see the differentiable generation of permutation matrices as a tool in our pipeline.

Corollary 7 (Permutation Matrix (9)) *Let $\boldsymbol{s} = [s_1, \dots, s_n]^T$ be a real-valued vector of length n. Let $A_{\boldsymbol{s}}$ denote the matrix of absolute pairwise differences of the elements of $\boldsymbol{s}$ such that $A_{\boldsymbol{s}}[i,j] = |s_i - s_j|$. The permutation matrix π corresponding to sort($\boldsymbol{s}$) is given by:*

$$\pi = \begin{cases} 1 & \text{if } j = \arg\max[(n+1-2i)\boldsymbol{s} - A_{\boldsymbol{s}}\mathbb{1}] \\ 0 & \text{otherwise} \end{cases} \tag{25}$$

where $\not\mathbb{1}$ denotes the column vector of all ones.

As we know, the arg max operator is non-differentiable which prohibits the direct use of Corollary 7 for gradient computation. Hence, Grover et al. (9) propose to replace the arg max operator with softmax to obtain a continuous relaxation $\pi(\tau)$. In particular, the ith row of $\pi(\tau)$ is given by:

$$\pi(\tau)[i,:] = \text{softmax}[(n+1-2i)\boldsymbol{s} - A_{\boldsymbol{s}}\mathbb{1}/\tau] \tag{26}$$

where $\tau > 0$ is a temperature parameter. We adapted this section from Grover et al. (9) and we also refer to their original work for more details on how to generate differentiable permutation matrices.

In this, work we remove the temperature parameter τ to reduce clutter in the notation. Hence, we only write π instead of $\pi(\tau)$, although it is still needed for the generation of the matrix π. For details on how we select the temperature parameter τ in our experiments, we refer to Appendix C.

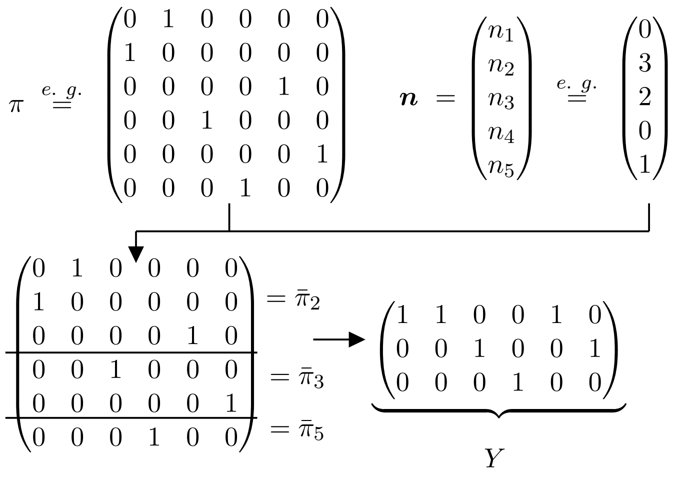

$\pi \overset{e.g.}{=}$
$\boldsymbol{n} = (n_1, n_2, n_3, n_4, n_5)^T \overset{e.g.}{=} (0, 3, 2, 0, 1)^T$
$= \bar{\pi}_2$
$= \bar{\pi}_3$
$= \bar{\pi}_5$
Y

Figure 2: Illustration of the proposed DRPM method. We first sample a permutation matrix π, and a set of subset sizes $\boldsymbol{n}$ separately in two stages. We then use $\boldsymbol{n}$ and π to generate the assignment matrix Y, the matrix representation of a partition ρ.

Appendix B. Method

B.1. Two-Stage Partition Model

B.1.1. Proof of Corollary 2

Proof By definition, every row $\boldsymbol{\pi}_i$ and column $\boldsymbol{\pi}_j$ of π is a one-hot vector, hence $\sum_{n_k} \boldsymbol{\pi}_i$ results in a n_k-hot encoding. Therefore, $\sum_{i=1}^{n_k} \sum_{j=1}^{n} \boldsymbol{\pi}_{ij} = n_k$ follows directly from π being a permutation matrix. Hence, if $\sum_k n_k = n$, every element i is assigned to one and only one $\boldsymbol{y}_k$. Thus, Theorem 1 fulfills $\mathcal{S}_1 \cup \cdots \cup \mathcal{S}_K = [n]$, and $\mathcal{S}_i \cap \mathcal{S}_j = \emptyset \;\; \forall \;\; i, j$ and $i \neq j$. ■

B.2. Computing p(Y)

We calculate the probability $p(Y; \boldsymbol{\omega}, \boldsymbol{s})$ sequentially over the probabilities of subsets

$$p_{\boldsymbol{y}_k} := p(\boldsymbol{y}_k \mid \boldsymbol{y}_{<k}; \boldsymbol{\omega}, \boldsymbol{s}).$$

$p_{\boldsymbol{y}_k}$ itself depends on the probability over subset permutations

$$p_{\bar{\pi}_k} := p(\bar{\pi} \mid n_k, \boldsymbol{y}_{<k}; \boldsymbol{s}),$$

where a subset permutation matrix $\bar{\pi}$ represents an ordering over n_k out of n elements.

Definition 8 (Subset permutation matrix $\bar{\pi}$) *A subset permutation matrix $\bar{\pi} \in \{0,1\}^{n_k \times n}$, where $n_k \leq n$, must fulfill*

$$\forall i \leq n_k : \sum_{j=1}^{n} \bar{\pi}_{ij} = 1 \;\; \textit{and} \;\; \forall j \leq n : \sum_{i=1}^{n_k} \bar{\pi}_{ij} \leq 1.$$

We describe the probability distribution over subset permutation matrices $p_{\bar{\pi}_k}$ using Definition 8 and proposition 6.

Lemma 9 (Probability over subset permutations $p_{\bar{\pi}_k}$) *The probability $p_{\bar{\pi}_k}$ of any subset permutation matrix $\bar{\pi} = [\bar{\boldsymbol{\pi}}_1, \dots, \bar{\boldsymbol{\pi}}_{n_k}]^T \in \{0,1\}^{n_k \times n}$ is given by*

$$p_{\bar{\pi}_k} := p(\bar{\pi} \mid n_k, \boldsymbol{y}_{<k}; \boldsymbol{s}) = \prod_{i=1}^{n_k} \frac{(\bar{\pi}\boldsymbol{s})_i}{Z_k - \sum_{j=1}^{i-1} (\bar{\pi}\boldsymbol{s})_j} \tag{27}$$

where $\boldsymbol{y}_{<k} = \{\boldsymbol{y}_1, ..., \boldsymbol{y}_{k-1}\}$, $Z_k = Z - \sum_{j \in \mathcal{S}_{<k}} \boldsymbol{s}_j$ and $\mathcal{S}_{<k} = \bigcup_{j=1}^{k-1} \mathcal{S}_j$.

Lemma 9, for which we provide the proof in Appendix B.2.1, describes the probability of drawing the elements $i \in \mathcal{S}_k$ in the order described by the subset permutation matrix $\bar{\pi}$ given that the elements in $\mathcal{S}_{<k}$ are already determined. Note that in a slight abuse of notation, we use $p(\bar{\pi} \mid n_k, \boldsymbol{y}_{<k}; \boldsymbol{\omega}, \boldsymbol{s})$ as the probability of a subset permutation $\bar{\pi}$ given that there are n_k elements in $\mathcal{S}_k$ and thus $\bar{\pi} \in \{0,1\}^{n_k \times n}$. Additionally, we condition on the subsets $\boldsymbol{y}_{<k}$ and n_k, the size of subset $\mathcal{S}_k$. In contrast to the distribution over permutations matrices $p(\pi; \boldsymbol{s})$ in proposition 6, we take the product over n_k terms and have a different normalization constant Z_k. Although we induce an ordering over all elements i in Definition 1, the probability $p_{\boldsymbol{y}_k}$ is invariant to intra-subset orderings of elements $i \in \mathcal{S}_k$.

Lemma 10 (Probability distribution $p_{\boldsymbol{y}_k}$) *The probability distribution over subset assignments $p_{\boldsymbol{y}_k}$ is given by*

$$p_{\boldsymbol{y}_k} := p(\boldsymbol{y}_k \mid \boldsymbol{y}_{<k}; \boldsymbol{\omega}, \boldsymbol{s}) = p(n_k \mid n_{<k}; \boldsymbol{\omega}) \sum_{\bar{\pi} \in \Pi_{\boldsymbol{y}_k}} p(\bar{\pi} \mid n_k, \boldsymbol{y}_{<k}; \boldsymbol{s})$$

where $\Pi_{\boldsymbol{y}_k} = \{\bar{\pi} \in \{0,1\}^{n_k \times n} : \boldsymbol{y}_k = \sum_{i=1}^{n_k} \bar{\boldsymbol{\pi}}_i\}$ and $p(\bar{\pi} \mid n_k, \boldsymbol{y}_{<k}; \boldsymbol{s})$ as in Lemma 9.

We provide the proof in Appendix B.2.2. We describe the set of all subset permutations $\bar{\pi}$ of elements $i \in \mathcal{S}_k$ by $\Pi_{\boldsymbol{y}_k}$. Put differently, we make $p(\boldsymbol{y}_k \mid \boldsymbol{y}_{<k}; \boldsymbol{\omega}, \boldsymbol{s})$ invariant to the ordering of elements $i \in \mathcal{S}_k$ by marginalizing over the probabilities of subset permutations $p_{\bar{\pi}_k}$ (35).

We propose the DRPM $p(Y; \boldsymbol{\omega}, \boldsymbol{s})$, a differentiable and reparameterizable two-stage RPM. Since $Y = [\boldsymbol{y}_1, \dots, \boldsymbol{y}_K]^T$, we calculate $p(Y; \boldsymbol{\omega}, \boldsymbol{s})$, the density of the differentiable RPM, sequentially using Lemmas 9 and 10, where we leverage the PL distribution for permutation matrices $p(\pi; \boldsymbol{s})$ to describe the probability distribution over subsets $p(\boldsymbol{y}_k \mid \boldsymbol{y}_{<k}; \boldsymbol{\omega}, \boldsymbol{s})$.

B.2.1. Proof of Lemma 9

Proof We provide the proof for $p_{\bar{\pi}_1}$, but it is equivalent for all other subsets. Without loss of generality, we assume that there are n_1 elements in $\mathcal{S}_1$. Following proposition 6, the probability of a permutation matrix $p(\pi; \boldsymbol{s})$ is given by

$$p(\pi; \boldsymbol{s}) = \frac{(\pi\boldsymbol{s})_1}{Z} \frac{(\pi\boldsymbol{s})_2}{Z - (\pi\boldsymbol{s})_1} \cdots \frac{(\pi\boldsymbol{s})_n}{Z - \sum_{j=1}^{n-1} (\pi\boldsymbol{s})_j} \tag{28}$$

At the moment, we are only interested in the ordering of the first n_1 elements. The probability of the first n_1 is given by marginalizing over the remaining $n - n_1$ elements:

$$p(\bar{\pi} \mid n_1; \boldsymbol{\omega}) = \sum_{\pi \in \Pi_1} p(\pi \mid \boldsymbol{s}) \tag{29}$$

where Π_1 is the set of permutation matrices such that the top n_1 rows select the elements in a specific ordering $\bar{\pi} \in \{0,1\}^{n_1 \times n}$, i.e. $\Pi_1 = \{\pi : [\boldsymbol{\pi}_1, \dots, \boldsymbol{\pi}_{n_1}]^T = \bar{\pi}\}$. It follows

$$\begin{aligned}
p(\bar{\pi} \mid n_1; \boldsymbol{\omega}) &= \sum_{\pi \in \Pi_1} p(\pi \mid \boldsymbol{s}) && (30)\\
&= \sum_{\pi \in \Pi_1} \prod_{i=1}^{n} \frac{(\pi \boldsymbol{s})_i}{Z - \sum_{j=1}^{i-1} (\pi \boldsymbol{s})_j} && (31)\\
&= \prod_{i=1}^{n_1} \frac{(\bar{\pi} \boldsymbol{s})_i}{Z - \sum_{j=1}^{i-1} (\bar{\pi} \boldsymbol{s})_j} \sum_{\pi \in \Pi_1} \prod_{i=1}^{n-n_1} \frac{(\pi \boldsymbol{s})_{n_1+i}}{Z - \sum_{j=1}^{n_1} (\bar{\pi} \boldsymbol{s})_j - \sum_{j=1}^{i-1} (\bar{\pi} \boldsymbol{s})_j} && (32)\\
&= \prod_{i=1}^{n_1} \frac{(\bar{\pi} \boldsymbol{s})_i}{Z - \sum_{j=1}^{i-1} (\bar{\pi} \boldsymbol{s})_j} \sum_{\pi \in \Pi_1} \prod_{i=1}^{n-n_1} \frac{(\pi \boldsymbol{s})_{n_1+i}}{Z_1 - \sum_{j=1}^{i-1} (\bar{\pi} \boldsymbol{s})_j} && (33)
\end{aligned}$$

where $Z_1 = Z - \sum_{j=1}^{n_1} (\bar{\pi} \boldsymbol{s})_j$. It follows

$$p(\bar{\pi} \mid n_1; \boldsymbol{\omega}) = \prod_{i=1}^{n_1} \frac{(\bar{\pi} \boldsymbol{s})_i}{Z - \sum_{j=1}^{i-1} (\bar{\pi} \boldsymbol{s})_j} \quad (34)$$

■

B.2.2. Proof of Lemma 10

Proof We can proof the statement of Lemma 10 as follows:

$$\begin{aligned}
p_{\boldsymbol{y}_k} &= p(\boldsymbol{y}_k \mid \boldsymbol{y}_{<k}; \boldsymbol{\omega}, \boldsymbol{s})\\
&= \sum_{n'_k} p(\boldsymbol{y}_k, n'_k \mid \boldsymbol{y}_{<k}; \boldsymbol{\omega}, \boldsymbol{s}) && (35)\\
&= \sum_{n'_k} p(n'_k \mid \boldsymbol{y}_{<k}; \boldsymbol{\omega}, \boldsymbol{s}) p(\boldsymbol{y}_k \mid n'_k, \boldsymbol{y}_{<k}; \boldsymbol{\omega}, \boldsymbol{s}) && (36)\\
&= \sum_{n'_k} p(n'_k \mid n_{<k}; \boldsymbol{\omega}, \boldsymbol{s}) p(\boldsymbol{y}_k \mid n'_k, \boldsymbol{y}_{<k}; \boldsymbol{s}) && (37)\\
&= p(n_k \mid n_{<k}; \boldsymbol{\omega}, \boldsymbol{s}) p(\boldsymbol{y}_k \mid n_k, \boldsymbol{y}_{<k}; \boldsymbol{s}) && (38)\\
&= p(n_k \mid n_{<k}; \boldsymbol{\omega}) \sum_{\bar{\pi} \in \Pi_{\boldsymbol{y}_k}} p(\bar{\pi} \mid n_k, \boldsymbol{y}_{<k}; \boldsymbol{s}) && (39)
\end{aligned}$$

Equation (35) holds by marginalization, where n'_k denotes the random variable that stands for the size of subset $\mathcal{S}_k$. By Bayese rule, we can then derive Equation (36). The next derivations stem from the fact that we can compute $n_{<k}$ if $\boldsymbol{y}_{<k}$ is given, as the assignments $\boldsymbol{y}_{<k}$ hold information on the size of subsets $\mathcal{S}_{<k}$. More explicitly, $n_i = \sum_{j=1}^{n} y_{ij}$. Further, $\boldsymbol{y}_k$ is independent of $\boldsymbol{\omega}$ if the size n'_k of subset $\mathcal{S}_k$ is given, leading to Equation (37). We further observe that $p(\boldsymbol{y}_k \mid n'_k, \boldsymbol{y}_{<k}; \boldsymbol{s})$ is only non-zero, if $n'_k = \sum_{i=1}^{n} y_{ki} = n_k$. Dropping all zero terms from the sum in Equation (37) thus results in Equation (38). Finally, by Theorem 1,

we know that $\boldsymbol{y}_k = \sum_{i=\nu_k+1}^{\nu_k+n_k} \boldsymbol{\pi}_i$, where $\nu_k = \sum_{\iota=1}^{k-1} n_\iota$ and $\pi \in \{0,1\}^{n\times n}$ a permutation matrix. Hence, in order to get $\boldsymbol{y}_k$ given $\boldsymbol{y}_{<k}$, we need to marginalize over all permutations of the elements of $\boldsymbol{y}_k$ given that the elements in $\boldsymbol{y}_{<k}$ are already ordered, which corresponds exactly to marginalizing over all subset permutation matrices $\bar{\pi}$ such that $\boldsymbol{y}_k = \sum_{i=1}^{n_k} \bar{\boldsymbol{\pi}}_i$, resulting in Equation (39).

■

B.2.3. Proof of Proposition 3

Proof From Lemmas 9 and 10, we write

$$\begin{aligned}
p(Y) =& p(\boldsymbol{y}_1, \ldots, \boldsymbol{y}_K; \boldsymbol{\omega}, \boldsymbol{s}) = p(\boldsymbol{y}_1; \boldsymbol{\omega}, \boldsymbol{s}) \cdots p(\boldsymbol{y}_K \mid \{\boldsymbol{y}_j\}_{j<K}; \boldsymbol{\omega}, \boldsymbol{s}) \\
=& \left(p(n_1; \boldsymbol{\omega}) \sum_{\bar{\pi}_1 \in \Pi_{\boldsymbol{y}_1}} p(\bar{\pi}_1 \mid n_1; \boldsymbol{s}) \right) \cdots \left(p(n_K \mid \{n_j\}_{j<K}; \boldsymbol{\omega}) \sum_{\bar{\pi}_K \in \Pi_{\boldsymbol{y}_K}} p(\bar{\pi}_K \mid \{n_j\}_{j\le K}; \boldsymbol{s}) \right) \\
=& p(n_1; \boldsymbol{\omega}) \cdots p(n_K \mid \{n_K\}_{j<K}; \boldsymbol{\omega}) \cdot \left(\sum_{\bar{\pi}_1 \in \Pi_{\boldsymbol{y}_1}} p(\bar{\pi}_1 \mid n_1; \boldsymbol{s}) \cdots \sum_{\pi_K \in \Pi_{\boldsymbol{y}_K}} p(\bar{\pi}_K \mid \{n_j\}_{j\le K}; \boldsymbol{s}) \right) \\
=& p(\boldsymbol{n}; \boldsymbol{\omega}) \left(\sum_{\bar{\pi}_1 \in \Pi_{\boldsymbol{y}_1}} \cdots \sum_{\pi_K \in \Pi_{\boldsymbol{y}_K}} p(\bar{\pi}_1 \mid n_1; \boldsymbol{s}) \cdots p(\bar{\pi}_K \mid \{n_j\}_{j\le K}; \boldsymbol{s}) \right) \\
=& p(\boldsymbol{n}; \boldsymbol{\omega}) \sum_{\pi \in \Pi_Y} p(\pi \mid \boldsymbol{n}; \boldsymbol{s}) \\
=& p(\boldsymbol{n}; \boldsymbol{\omega}) \sum_{\pi \in \Pi_Y} p(\pi; \boldsymbol{s})
\end{aligned}$$

■

B.3. Approximating p(Y)

B.3.1. Proof of Lemma 4

Proof Since $p(\pi; \boldsymbol{s})$ is a probability we know that $\forall \pi \in \{0,1\}^{n\times n} \;\; p(\pi; \boldsymbol{s}) \ge 0$. Thus, it follows directly that:

$$\forall \pi \in \Pi_Y : \quad p(Y; \boldsymbol{\omega}, \boldsymbol{s}) = p(\boldsymbol{n}; \boldsymbol{\omega}) \sum_{\pi' \in \Pi_Y} p(\pi'; \boldsymbol{s}) \ge p(\boldsymbol{n}; \boldsymbol{\omega}) p(\pi; \boldsymbol{s}),$$

proving Equation (4).

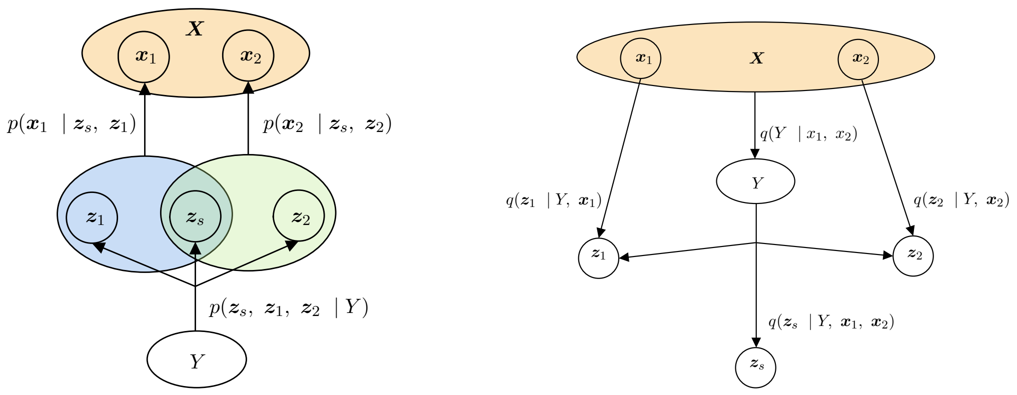
On the other hand, can prove Equation (5) by:

$$\begin{aligned}
p(Y;\boldsymbol{\omega},\boldsymbol{s}) = p(\boldsymbol{n};\boldsymbol{\omega}) \sum_{\pi'\in\Pi_Y} p(\pi';\boldsymbol{s}) \\
\leq & p(\boldsymbol{n};\boldsymbol{\omega}) \sum_{\pi'\in\Pi_Y} \max_{\pi\in\Pi_Y} p(\pi;\boldsymbol{s}) \\
= & p(\boldsymbol{n};\boldsymbol{\omega}) \max_{\pi\in\Pi_Y} p(\pi;\boldsymbol{s}) \sum_{\pi'\in\Pi_Y} 1 \\
= & |\Pi_Y| \cdot p(\boldsymbol{n};\boldsymbol{\omega}) \max_{\pi\in\Pi_Y} p(\pi;\boldsymbol{s}) \\
\leq & |\Pi_Y| \cdot p(\boldsymbol{n};\boldsymbol{\omega}) \max_{\pi} p(\pi;\boldsymbol{s})
\end{aligned}$$

We can compute the maximum probability $\max_\pi p(\pi;\boldsymbol{s})$ with the probability of the permutation matrix $f_\pi(\boldsymbol{s})$, which sorts the unperturbed scores in decreasing order. ■

Appendix C. Experiments

C.1. Weakly-Supervised Learning


$\boldsymbol{X}$
$\boldsymbol{x}_1$
$\boldsymbol{x}_2$
$p(\boldsymbol{x}_1 \mid \boldsymbol{z}_s,\ \boldsymbol{z}_1)$
$p(\boldsymbol{x}_2 \mid \boldsymbol{z}_s,\ \boldsymbol{z}_2)$
$\boldsymbol{z}_1$
$\boldsymbol{z}_s$
$\boldsymbol{z}_2$
$p(\boldsymbol{z}_s,\ \boldsymbol{z}_1,\ \boldsymbol{z}_2 \mid Y)$
Y
$\boldsymbol{x}_1$
$\boldsymbol{X}$
$\boldsymbol{x}_2$
$q(Y \mid x_1,\ x_2)$
Y
$q(\boldsymbol{z}_1 \mid Y,\ \boldsymbol{x}_1)$
$q(\boldsymbol{z}_2 \mid Y,\ \boldsymbol{x}_2)$
$\boldsymbol{z}_1$
$\boldsymbol{z}_2$
$q(\boldsymbol{z}_s \mid Y,\ \boldsymbol{x}_1,\ \boldsymbol{x}_2)$
$\boldsymbol{z}_s$

(a) Generative Model (b) Inference Model

Figure 3: Graphical Models for DRPMVAE models in the weakly-supervised experiment.

C.1.1. Baseline Methods

We compare the proposed DRPMVAE to three methods, which only differ in how they infer shared and latent dimensions. Using a single encoder, all methods independently encode both images to some latent representations, which are used to infer shared latent dimensions. A single decoder independently reconstructs the two views from an aggregated latent vector consisting of a combination of shared and independent factors. While the LabelVAE (3; 12) assumes that the number of independent factors is known, the AdaVAE (20) relies on a heuristic-based approach to infer shared and independent latent factors. Like in Locatello et al. (20) and Sutter et al. (32), we assume a single known factor for LabelVAE in all experiments. HGVAE (32) also relies on the MVHG to model the number of shared and independent factors. Unlike the proposed DRPMVAE approach, HGVAE must rely on a heuristic to assign latent dimensions to shared factors, as the MVHG only allows to model the number of shared and independent factors.

C.1.2. Generative Model

We assume the following generative model for DRPMVAE

$$\begin{aligned} p(\boldsymbol{X}) &= \int_{\boldsymbol{z}} p(\boldsymbol{X}, \boldsymbol{z}) d\boldsymbol{z} && (40) \\ &= \int_{\boldsymbol{z}} p(\boldsymbol{X} \mid \boldsymbol{z}) p(\boldsymbol{z}) d\boldsymbol{z} && (41) \end{aligned}$$

where $\boldsymbol{z} = \{\boldsymbol{z}_s, \boldsymbol{z}_1, \boldsymbol{z}_2\}$. The two frames share an unknown number n_s of generative latent factors $\boldsymbol{z}_s$, and an unknown number, n_1 and n_2, of independent factors $\boldsymbol{z}_1$ and $\boldsymbol{z}_2$. The RPM infers n_k and $\boldsymbol{z}_k$ using Y. Hence, the generative model extends to

$$\begin{aligned} p(\boldsymbol{X}) &= \int_{\boldsymbol{z}} p(\boldsymbol{X} \mid \boldsymbol{z}) \sum_{Y} p(\boldsymbol{z} \mid Y) p(Y) d\boldsymbol{z} \\ &= \int_{\boldsymbol{z}} p(\boldsymbol{x}_1, \boldsymbol{x}_2 \mid \boldsymbol{z}_s, \boldsymbol{z}_1, \boldsymbol{z}_2) \sum_{Y} p(\boldsymbol{z} \mid Y) p(Y) d\boldsymbol{z} \\ &= \int_{\boldsymbol{z}_s, \boldsymbol{z}_1, \boldsymbol{z}_2} p(\boldsymbol{x}_1 \mid \boldsymbol{z}_s, \boldsymbol{z}_1) p(\boldsymbol{x}_2 \mid \boldsymbol{z}_s, \boldsymbol{z}_2) \sum_{Y} p(\boldsymbol{z}_s, \boldsymbol{z}_1, \boldsymbol{z}_2 \mid Y) p(Y) d\boldsymbol{z}_s d\boldsymbol{z}_1 d\boldsymbol{z}_2 && (42) \end{aligned}$$

Figure 3 shows the generative and inference models assumptions in a graphical model.

C.1.3. DRPM ELBO

Following Theorem 4, we are able to optimize DRPMVAE using the following ELBO:

$$\begin{aligned}
\log p(\boldsymbol{X}) \geq & \mathbb{E}_{q(\boldsymbol{z},Y \mid \boldsymbol{X})}\left[\log p(\boldsymbol{X} \mid \boldsymbol{z}, Y) + \log \frac{q(\boldsymbol{z}, Y \mid \boldsymbol{X})}{p(\boldsymbol{z}, Y)}\right] && (43)\\
= & \mathbb{E}_{q(\boldsymbol{z},Y \mid \boldsymbol{X})}\left[\log p(\boldsymbol{X} \mid \boldsymbol{z}) + \log \frac{q(\boldsymbol{z} \mid Y, \boldsymbol{X}) q(Y \mid \boldsymbol{X})}{p(\boldsymbol{z}) p(Y)}\right] && (44)\\
= & \mathbb{E}_{q(\boldsymbol{z},Y \mid \boldsymbol{X})}\left[\log p(\boldsymbol{x}_1, \boldsymbol{x}_2 \mid \boldsymbol{z}) + \log \frac{q(\boldsymbol{z} \mid Y, \boldsymbol{X})}{p(\boldsymbol{z})} - \log \frac{q(Y \mid \boldsymbol{X})}{p(Y)}\right] && (45)\\
= & \mathbb{E}_{q(\boldsymbol{z},Y \mid \boldsymbol{X})}\left[\log p(\boldsymbol{x}_1 \mid \boldsymbol{z}_s, \boldsymbol{z}_1)\right] + \mathbb{E}_{q(\boldsymbol{z},Y \mid \boldsymbol{X})}\left[\log p(\boldsymbol{x}_2 \mid \boldsymbol{z}_s, \boldsymbol{z}_2)\right] \\
& + \mathbb{E}_{q(\boldsymbol{z},Y \mid \boldsymbol{X})}\left[\log \frac{q(\boldsymbol{z}_s, \boldsymbol{z}_1, \boldsymbol{z}_2 \mid Y, \boldsymbol{X})}{p(\boldsymbol{z}_s, \boldsymbol{z}_1, \boldsymbol{z}_2)}\right] + \mathbb{E}_{q(\boldsymbol{z},Y \mid \boldsymbol{X})}\left[\log \frac{q(Y \mid \boldsymbol{X})}{p(Y)}\right] && (46)\\
\geq & \mathbb{E}_{q(\boldsymbol{z},Y \mid \boldsymbol{X})}\left[\log p(\boldsymbol{x}_1 \mid \boldsymbol{z}_s, \boldsymbol{z}_1)\right] + \mathbb{E}_{q(\boldsymbol{z},Y \mid \boldsymbol{X})}\left[\log p(\boldsymbol{x}_2 \mid \boldsymbol{z}_s, \boldsymbol{z}_2)\right] \\
& + \mathbb{E}_{q(\boldsymbol{z},Y \mid \boldsymbol{X})}\left[\log \frac{q(\boldsymbol{z}_s, \boldsymbol{z}_1, \boldsymbol{z}_2 \mid Y, \boldsymbol{X})}{p(\boldsymbol{z}_s, \boldsymbol{z}_1, \boldsymbol{z}_2)}\right] \\
& + \mathbb{E}_{q(\boldsymbol{z},Y \mid \boldsymbol{X})}\left[\log \frac{q(\boldsymbol{n} \mid \boldsymbol{X}; \boldsymbol{\omega}) \cdot |\Pi_Y|}{p(\boldsymbol{n}; \boldsymbol{\omega}_p)}\right] \\
& + \mathbb{E}_{q(\boldsymbol{z},Y \mid \boldsymbol{X})}\left[\log \frac{\max_{\pi \in \Pi_Y} q(\pi \mid \boldsymbol{X}; \boldsymbol{s})}{\max_{\pi \in \Pi_Y} p(\pi \mid \boldsymbol{X}; \boldsymbol{s}_p)}\right] && (47)
\end{aligned}$$

The ELBO $\mathcal{L}(\boldsymbol{X})$ to be optimized can be written as

$$\begin{aligned}
\mathcal{L}(\boldsymbol{X}) = & \mathbb{E}_{q(\boldsymbol{z},Y \mid \boldsymbol{X})}\left[\log p(\boldsymbol{x}_1 \mid \boldsymbol{z}_s, \boldsymbol{z}_1)\right] + \mathbb{E}_{q(\boldsymbol{z},Y \mid \boldsymbol{X})}\left[\log p(\boldsymbol{x}_2 \mid \boldsymbol{z}_s, \boldsymbol{z}_2)\right] \\
& + \beta \cdot \mathbb{E}_{q(\boldsymbol{z},Y \mid \boldsymbol{X})}\left[\log \frac{q(\boldsymbol{z}_s, \boldsymbol{z}_1, \boldsymbol{z}_2 \mid Y, \boldsymbol{X})}{p(\boldsymbol{z}_s, \boldsymbol{z}_1, \boldsymbol{z}_2)}\right] \\
& + \gamma \cdot \left(\mathbb{E}_{q(\boldsymbol{z},Y \mid \boldsymbol{X})}\left[\log \frac{q(\boldsymbol{n} \mid \boldsymbol{X}; \boldsymbol{\omega}) \cdot |\Pi_Y|}{p(\boldsymbol{n}; \boldsymbol{\omega}_p)}\right] + \mathbb{E}_{q(\boldsymbol{z},Y \mid \boldsymbol{X})}\left[\log \frac{\max_{\pi \in \Pi_Y} q(\pi \mid \boldsymbol{X}; \boldsymbol{s})}{\max_{\pi \in \Pi_Y} p(\pi \mid \boldsymbol{X}; \boldsymbol{s}_p)}\right]\right) && (48)
\end{aligned}$$

C.1.4. Implementation and Hyperparameters

In this experiment, we use the `disentanglement_lib` from Locatello et al. (20). We use the same architectures proposed in the original paper for all methods we compare to. The baseline algorithms, LabelVAE (3; 12) and AdaVAE (20) are already implemented in `disentanglement_lib`. For details on the implementation of these methods we refer to the original paper from Locatello et al. (20). HGVAE is implemented in Sutter et al. (32). We did not change any hyperparameters or network details. All experiments were performed using $\beta = 1$ as this is the best performing β (according to Locatello et al. (20). For DRPMVAE we chose $\gamma = 0.25$ for all runs. All models are trained on 5 different random seeds and the reported results are averaged over the 5 seeds. We report mean performance with standard deviations.

We adapted Figure 4 from Sutter et al. (32). It shows the baseline architecture, which is used for all methods. As already stated in the main part of the paper, the methods

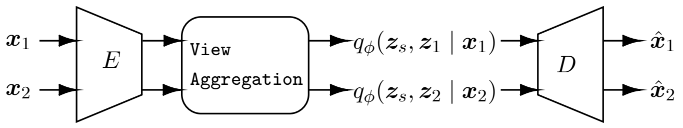


Figure 4: Setup for the weakly-supervised experiment. The three methods differ only in the `View Aggregation` module.

only differ in the `View Aggregation` module, which determines the shared and independent latent factors. Given a subset S of shared latent factors, we have

$$q_\phi(z_i \mid \boldsymbol{x}_j) = avg(q_\phi(z_i \mid \boldsymbol{x}_1), q_\phi(z_i \mid \boldsymbol{x}_2)) \qquad \forall \;\; i \in S \qquad (49)$$

$$q_\phi(z_i \mid \boldsymbol{x}_j) = q_\phi(z_i \mid \boldsymbol{x}_j) \qquad \text{else} \qquad (50)$$

where avg is the averaging function of choice (20; 32) and $j \in \{1, 2\}$. The methods used (i. e. LabelVAE, AdaVAE, HGVAE, DRPMVAE) differ in how to select the subset S.

For DRPMVAE, we infer $\boldsymbol{\omega}$ from the pairwise KL-divergences between the latent vectors of the two views. The KL-divergence values are fed to a single fully-connected layer, which maps from d to K values where $d = 10$ and $K = 2$ in this experiment. d is the total number of latent dimensions and K is the number of groups in the latent space. The scores $\boldsymbol{s}$ are equal to the KL-divergence values.

Similar to the original works, we also anneal the temperature parameter for $p(\boldsymbol{n}; \boldsymbol{\omega})$ and $p(\pi; \boldsymbol{s})$ (9; 32). We use the same annealing function as in the conditional generation experiment (see Appendix C.3). We anneal the temperature τ from 1.0 to 0.5 over the complete training time.

C.2. Additional Results

In Figure 5, we assess the methods with respect to how well they estimate the number of shared factors. In Figure 5, we see that DRPM can accurately estimate the true number of shared generative factors. It matches the performance of HGVAE and outperforms the other two baselines, which consistently overestimate the true number of shared factors.

C.3. Unsupervised Conditional Generation

C.3.1. Additional Samples

In addition to the small partition with eight elements presented in Figure 1, we sample some additional larger partitions of size 32 in Figure 7. Note that despite not cherrypicking any of these partitions, each set of the partition stems from a single digit, which implies that each of the priors learned to represent one of the MNIST digits. To further investigate whether this is true, we repeatedly sample from each of the priors in Figure 8. As demonstrated there, each of the priors nicely represents a digit and also captures some variability that is present in the dataset.

C.3.2. Loss Function

As mentioned in Section 3.2, for a given dataset $\mathcal{X} = \{x_1, ..., x_N\}$ with N samples, let $\mathcal{Z} = \{z_1, ..., z_N\}$, and $Y = \{y_1, ..., y_N\}$ contain the respective latent vectors and cluster

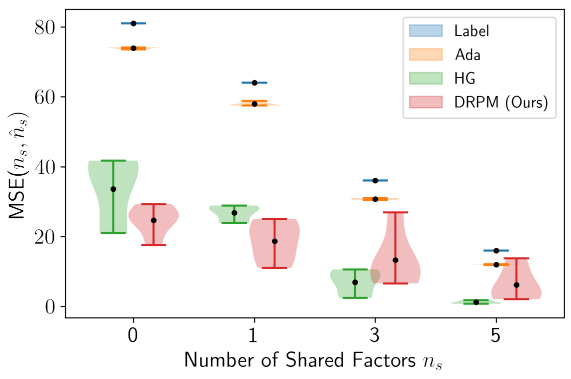


Figure 5: Mean squared error between the estimated number of shared factors $\hat{n}_s$ and the true number of shared factors n_s across five seeds for the LabelVAE, AdaVAE, HGVAE, and DRPMVAE.

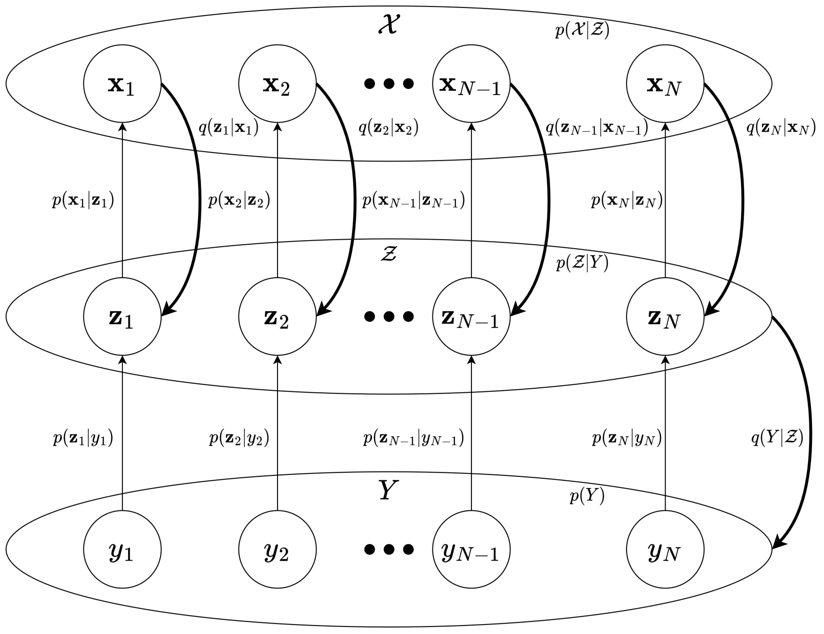

$\mathcal{X}$
$p(\mathcal{X}|\mathcal{Z})$
$\mathbf{x}_1$
$\mathbf{x}_2$
$\mathbf{x}_{N-1}$
$\mathbf{x}_N$
$q(\mathbf{z}_1|\mathbf{x}_1)$
$q(\mathbf{z}_2|\mathbf{x}_2)$
$q(\mathbf{z}_{N-1}|\mathbf{x}_{N-1})$
$q(\mathbf{z}_N|\mathbf{x}_N)$
$p(\mathbf{x}_1|\mathbf{z}_1)$
$p(\mathbf{x}_2|\mathbf{z}_2)$
$p(\mathbf{x}_{N-1}|\mathbf{z}_{N-1})$
$p(\mathbf{x}_N|\mathbf{z}_N)$
$\mathcal{Z}$
$p(\mathcal{Z}|Y)$
$\mathbf{z}_1$
$\mathbf{z}_2$
$\mathbf{z}_{N-1}$
$\mathbf{z}_N$
$p(\mathbf{z}_1|y_1)$
$p(\mathbf{z}_2|y_2)$
$p(\mathbf{z}_{N-1}|y_{N-1})$
$p(\mathbf{z}_N|y_N)$
$q(Y|\mathcal{Z})$
Y
$p(Y)$
y_1
y_2
y_{N-1}
y_N

Figure 6: Generative model of the DRPM conditional generation model. Generative paths are marked with thin arrows, whereas inference is in bold.

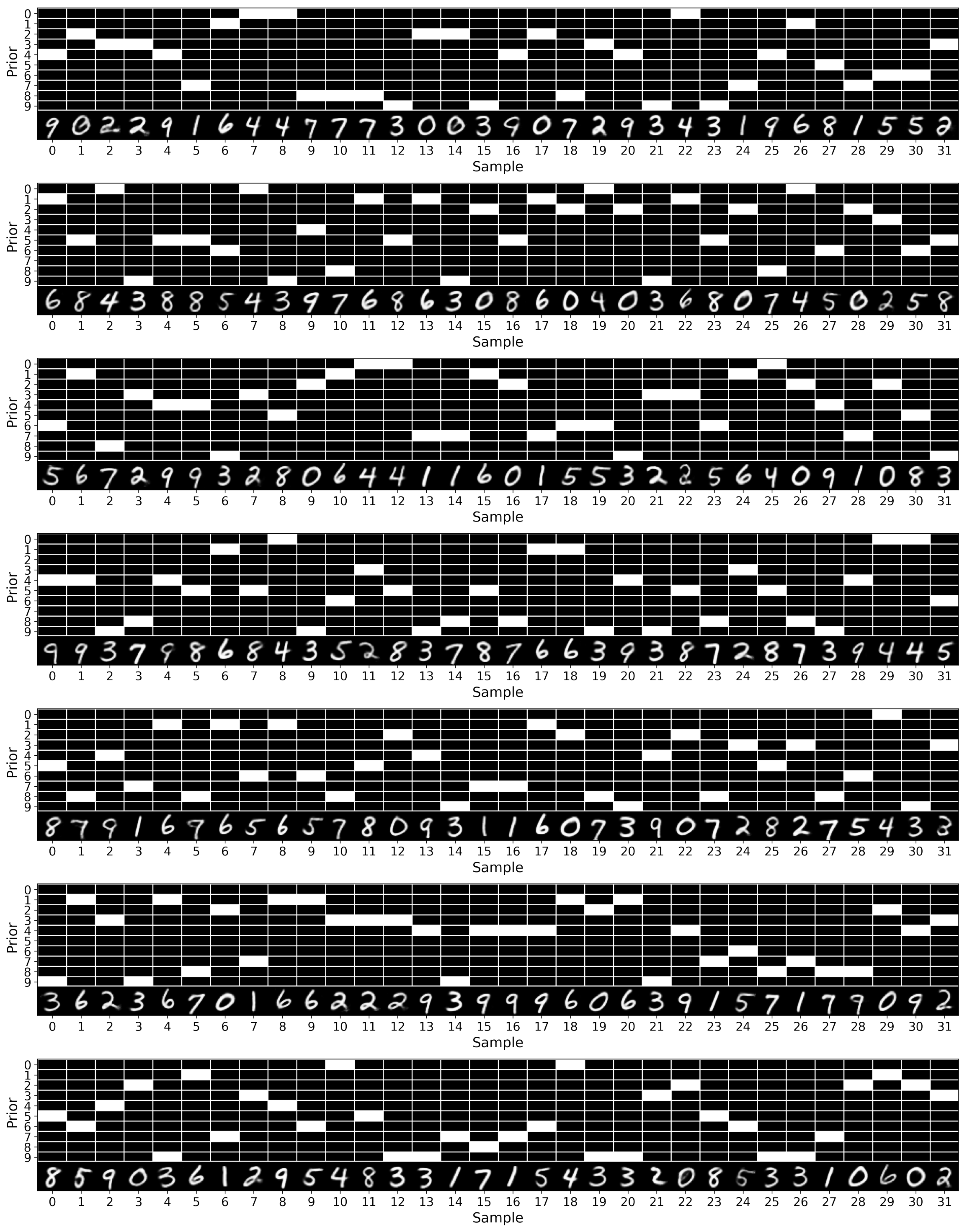


Figure 7: More sampled partitions from the generative DRPM model. The different sets of each partition match each of the digits relatively well also for larger partitions of size 32. Samples for this plot were randomly drawn and not cherry-picked.

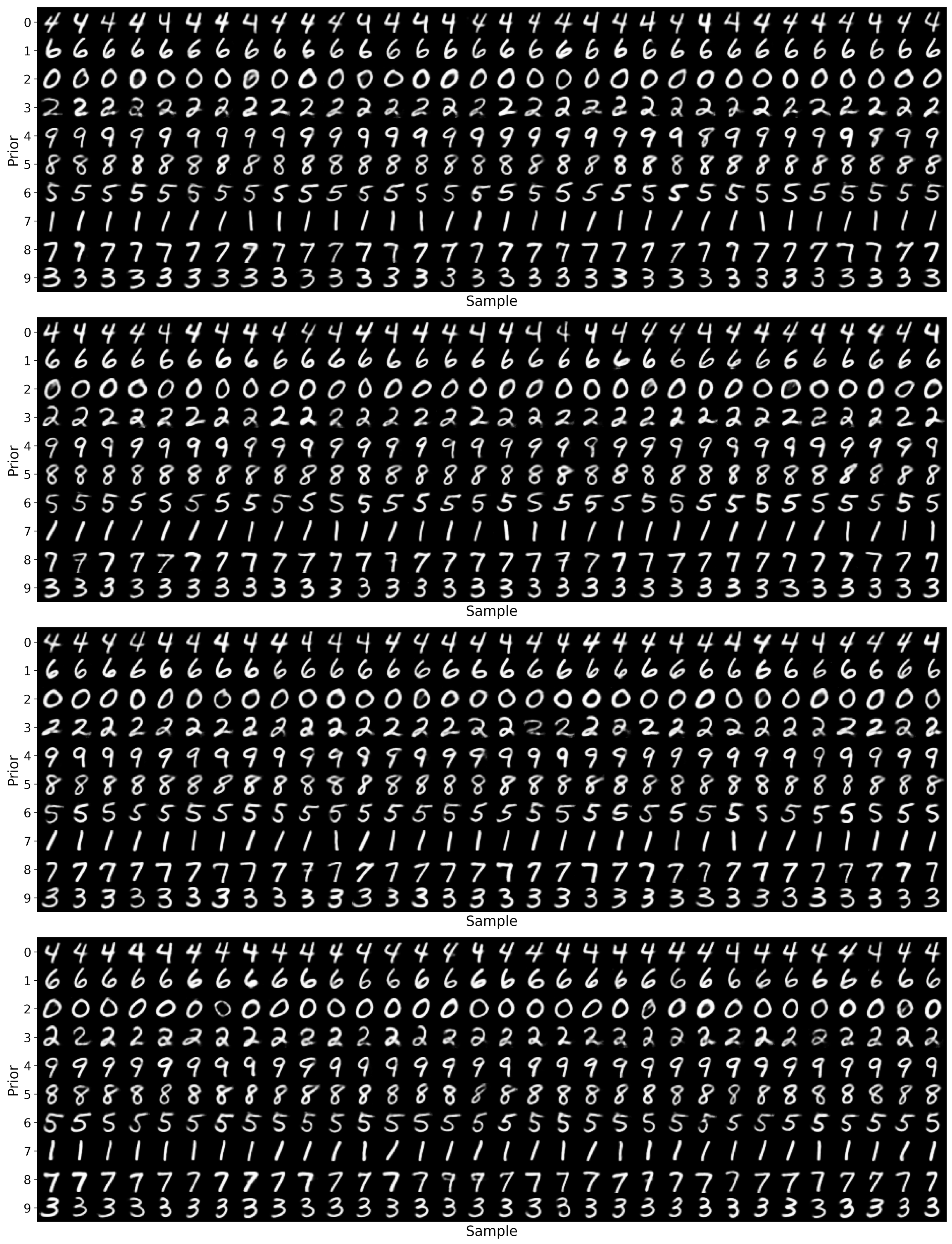


Figure 8: Various samples from each of the generative priors. Each prior learns to represent one of the digits. Samples for this plot were randomly drawn and not cherry-picked.

assignments for each sample. The generative process can then be summarized as follows: First, we sample the cluster assignments Y from an RPM, i.e. $Y \sim DRPM(\boldsymbol{\omega}, \boldsymbol{s})$. Given Y, we can sample the latent variables $\mathcal{Z}$, where $\boldsymbol{z_i} \sim \mathcal{N}(\boldsymbol{\mu_{y_i}}, \boldsymbol{\sigma_{y_i}^T}\mathbb{I}_l)$, $\boldsymbol{z_i} \in \mathbb{R}^l$. Finally, we sample $\mathcal{X}$, where $\boldsymbol{x_i} \sim \mathcal{N}(\boldsymbol{\mu_{z_i}}, \boldsymbol{\sigma_{z_i}^T}\mathbb{I}_d)$ if $\boldsymbol{x_i} \in \mathbb{R}^d$ or $\boldsymbol{x_i} \sim Ber(\boldsymbol{\mu_{z_i}})$ if $\boldsymbol{x_i} \in \{0,1\}^d$. Using Bayes rule and Jensen's inequality we can then derive the following evidence lower bound (ELBO):

$$\begin{aligned}
\log(p(\mathcal{X})) &= \log\left(\int \sum_Y p(\mathcal{X}, Y, \mathcal{Z}) d\mathcal{Z}\right) \\
&\geq \mathbb{E}_{q(\mathcal{Z},Y|\mathcal{X})}\left[\log\left(\frac{p(\mathcal{X}|\mathcal{Z})p(\mathcal{Z}|Y)p(Y)}{q(\mathcal{Z},Y|\mathcal{X})}\right)\right] \\
&:= \mathcal{L}_{ELBO}(X)
\end{aligned}$$

We then assume that we can factorize the approximate posterior as follows:

$$q(\mathcal{Z}, Y|\mathcal{X}) = q(Y|\mathcal{X}) \prod_{\boldsymbol{x}\in\mathcal{X}} q(\boldsymbol{z}|\boldsymbol{x})$$

Note that while we do assume conditional independence between $\boldsymbol{z}$ given its corresponding $\boldsymbol{x}$, we model $q(Y|\mathcal{X})$ with the DRPM and do not have to assume conditional independence between different cluster assignments. This has the advantage that we can leverage dependencies between samples from the dataset. Hence, we can rewrite the ELBO as follows:

$$\begin{aligned}
\mathcal{L}_{ELBO}(X) &= \mathbb{E}_{q(\mathcal{Z}|\mathcal{X})}\left[\log(p(\mathcal{X}|\mathcal{Z}))\right] \\
&- \mathbb{E}_{q(Y|\mathcal{X})}\left[KL[q(\mathcal{Z}|\mathcal{X})||p(\mathcal{Z}|Y)]\right] \\
&- KL[q(Y|\mathcal{X})||p(Y)] \\
&= \sum_{\boldsymbol{x}\in\mathcal{X}} \mathbb{E}_{q(\boldsymbol{z}|\boldsymbol{x})}\left[\log p(\boldsymbol{x}|\boldsymbol{z})\right] \\
&- \sum_{\boldsymbol{x}\in\mathcal{X}} \mathbb{E}_{q(Y|\mathcal{X})}\left[KL[q(\boldsymbol{z}|\boldsymbol{x})||p(\boldsymbol{z}|Y)]\right] \\
&- KL[q(Y|\mathcal{X})||p(Y)]
\end{aligned}$$

See Figure 6 for an illustration of the generative process and the assumed inference model.

C.3.3. Architecture

The model for our unsupervised conditional generation experiment is a relatively simple, fully-connected autoencoder with a structure as seen in Figure 9. We have a fully connected encoder E with four layers mapping the input to 500, 500, and 2000 neurons, respectively. We then compute each parameter by passing the encoder output through a linear layer and mapping to the respective parameter dimension in the last layer. In our experiments, we use a latent dimension size of 10, hence $\mu_x, \sigma_x \in \mathbb{R}^{10}$. In order to learn dependencies between samples, we map the last layer of the $\boldsymbol{\omega_{\mathcal{X}}}$ parameter to K, where K is the number of dimensions. We then apply a softmax activation for each sample, average the resulting vector over the batch, and take the logarithm since we want to model $\log \boldsymbol{\omega_{\mathcal{X}}}$. To compute the score s_x of a sample x, we map the last layer to dimension K and apply a softmax

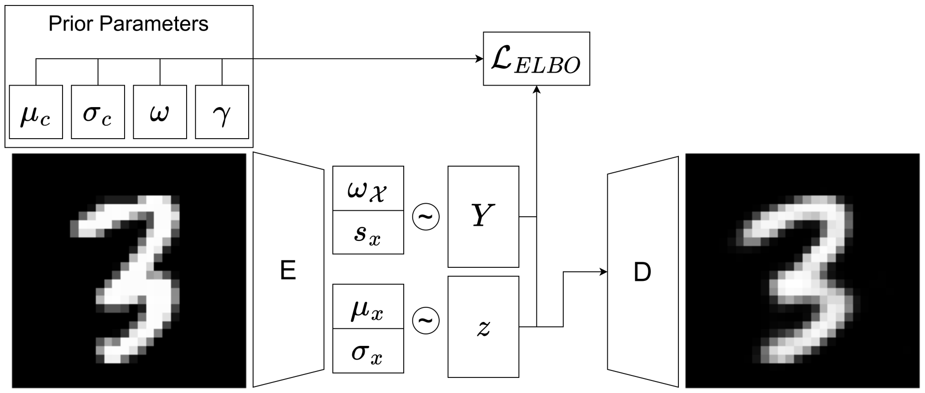


Figure 9: Architecture of the DRPM conditional generation model.

activation to that representation to compute intermediate representation $\boldsymbol{r} \in [0,1]^K$. Since we know that the scores for samples in the same cluster should be approximately equal, we compute s_x by $\log(s_x) = \lambda \sum_{i=1}^{K} i \cdot r_i$, where λ is a learnable parameter that is responsible for scaling the scores to an appropriate magnitude. Note that we thus compute s_x per sample independently of the other samples in the batch. Finally, once we resample $z \sim \mathcal{N}(\mu_x, \sigma_x)$, we pass it through a fully connected decoder D with four layers mapping z to 2000, 500, and 500 neurons in the first three layers and then finally back to the input dimension in the last layer to end up with the reconstructed sample $\hat{x}$.

C.3.4. Training

As described in Section 3.2 and Appendix C.3.2, we use the following loss to train the conditional generation experiment:

$$\begin{aligned}
\mathcal{L}_{ELBO} = &\sum_{\boldsymbol{x}\in\mathcal{X}} \mathbb{E}_{q(\boldsymbol{z}|\boldsymbol{x})}\left[\log p(\boldsymbol{x}|\boldsymbol{z})\right] && (51)\\
&- \sum_{\boldsymbol{x}\in\mathcal{X}} \mathbb{E}_{q(Y|\mathcal{X})}\left[KL[q(\boldsymbol{z}|\boldsymbol{x})||p(\boldsymbol{z}|Y)]\right] && (52)\\
&- \mathbb{E}_{q(Y|\mathcal{X})}\left[\log \frac{|\Pi_Y| \cdot q(\boldsymbol{n};\boldsymbol{\omega}(\mathcal{X}))}{p(\boldsymbol{n};\boldsymbol{\omega})p(\pi_Y;\boldsymbol{s})}\right] && (53)\\
&- \log\left(\max_\pi q(\pi;\boldsymbol{s}(\mathcal{X}))\right) && (54)
\end{aligned}$$

Due to the fact that we applied an upper bound on the KL-divergence $KL[q(Y|X)||p(Y)]$, we need to weight the KL-divergence terms in a similar fashion as in the β-VAE (11). To train the model, we reshuffle the terms in Equations (53) and (54), leading to the following formulation:

$$\begin{aligned}
\mathcal{L}_{ELBO} = &\sum_{\boldsymbol{x}\in\mathcal{X}} \mathbb{E}_{q(\boldsymbol{z}|\boldsymbol{x})}\left[\log p(\boldsymbol{x}|\boldsymbol{z})\right] - \sum_{\boldsymbol{x}\in\mathcal{X}} \mathbb{E}_{q(Y|\mathcal{X})}\left[\beta \cdot KL[q(\boldsymbol{z}|\boldsymbol{x})||p(\boldsymbol{z}|Y)]\right]\\
&- \mathbb{E}_{q(Y|\mathcal{X})}\left[\gamma \cdot \log\left(\frac{|\Pi_Y| \cdot q(\boldsymbol{n};\boldsymbol{\omega}(\mathcal{X}))}{p(\boldsymbol{n};\boldsymbol{\omega})}\right) + \delta \cdot \log\left(\frac{\max_\pi q(\pi;\boldsymbol{s}(\mathcal{X}))}{p(\pi_Y;\boldsymbol{s})}\right)\right]
\end{aligned}$$

This has the advantage that we can balance regularization for balanced clusters with γ and randomness of the permutations through δ. As in vanilla VAEs we can estimate the reconstruction term in Equation (51) with MCMC by applying the reparametrization trick (17) to $q(z|X)$ in order to sample M samples $z^{(i)} \sim q(z|X)$ and compute their reconstruction error to estimate Equation (51). Similarly, as described in Section 2.2, we apply the reparametrization trick to $p(\boldsymbol{n}; \boldsymbol{s})$ and $p(\pi; \boldsymbol{s})$ to attain L reparametrized samples $Y^{(i)} \sim q(Y|\mathcal{X})$. These can then be used to compute the KL divergences in Equations (52) to (54) in closed form. Finally, this leads to the following loss during training:

$$\begin{aligned}\mathcal{L}_{ELBO} = &\sum_{\boldsymbol{x}\in\mathcal{X}} \frac{1}{M}\sum_{i=1}^{M} \log p(\boldsymbol{x}|\boldsymbol{z}^{(i)}) - \sum_{\boldsymbol{x}\in\mathcal{X}} \frac{1}{L}\sum_{i=1}^{L} \beta \cdot KL[q(\boldsymbol{z}|\boldsymbol{x})||p(\boldsymbol{z}|Y^{(i)})] \\ &- \frac{1}{L}\sum_{i=1}^{L}\left(\gamma \cdot \log\left(\frac{|\Pi_{Y^{(i)}}| \cdot q(\boldsymbol{n}^{(i)}; \boldsymbol{\omega}(\mathcal{X}))}{p(\boldsymbol{n}^{(i)}; \boldsymbol{\omega})}\right) + \delta \cdot \log\left(\frac{\max_\pi q(\pi; \boldsymbol{s}(\mathcal{X}))}{p(\pi^{(i)}; \boldsymbol{s})}\right)\right)\end{aligned}$$

In our experiments, we set $M = 1$ as in vanilla VAEs and $L = 100$ since the MVHG, and PL distributions are not concentrated around their mean very well, such that more Monte Carlo samples lead to much better approximations of the expectation terms. In order to resample $\boldsymbol{n}$ and π we need to apply temperature annealing (9; 32). To do this, we applied the exponential schedule that was originally proposed together with the Gumbel-Softmax trick (13), i.e. $\tau = \max(\tau_{final}, exp(-rt))$, where t is the current training step and r is the annealing rate. For our experiments, we choose $r = \frac{\log(\tau_{final}) - \log(\tau_{init})}{100000}$ in order to annealing over 100000 training step. Like Jang et al. (13), we set $\tau_{init} = 1$ and $\tau_{final} = 0.5$. Similarly, we also annealed the weights β, γ, and δ with the same schedule, setting $\beta_{init} = 0.1 \cdot \beta_{final}$, $\beta_{final} = 0.1$, $\gamma_{init} = 0.1 \cdot \gamma_{final}$, $\gamma_{final} = 1$, $\delta_{init} = 0.1 \cdot \delta_{final}$, and $\delta_{final} = 0.001$.

We trained with the Adam (16) optimizer and learning rate 0.0001 with a batch size of 256 for 512 epochs.