Consistent Amortized Clustering via Generative Flow Networks

Irit Chelly[1] **Roy Uziel**[1] **Oren Freifeld**[1,3] **Ari Pakman**[2,3]
[1]Department of Computer Science [2]Department of Industrial Engineering and Management
[3]The School of Brain Sciences and Cognition
Ben-Gurion University of the Negev, Beer Sheva, Israel

Abstract

Neural models for amortized probabilistic clustering yield samples of cluster labels given a set-structured input, while avoiding lengthy Markov chain runs and the need for explicit data likelihoods. Existing methods which label each data point sequentially, like the Neural Clustering Process, often lead to cluster assignments highly dependent on the data order. Alternatively, methods that sequentially create full clusters, do not provide assignment probabilities. In this paper, we introduce GFNCP, a novel framework for amortized clustering. GFNCP is formulated as a Generative Flow Network with a shared energy-based parametrization of policy and reward. We show that the flow matching conditions are equivalent to consistency of the clustering posterior under marginalization, which in turn implies order invariance. GFNCP also outperforms existing methods in clustering performance on both synthetic and real-world data.

1 INTRODUCTION

Probabilistic clustering models, also known as mixture models, play a crucial role in many scientific domains and are extensively used in various downstream tasks. These models aim to learn the underlying data structure by grouping similar data points into clusters, with cluster assignments encoded in the posterior distribution of discrete latent variables.

Given a generative model for the clusters, traditional posterior inference methods, such as Markov Chain Monte Carlo (MCMC) (Neal, 2000) and their fast implementations (Chang and Fisher III, 2013; Dinari et al., 2019, 2022), yield samples from the posterior distribution, providing an asymptotically exact solution given enough samples. However, they often struggle with high-dimensional data or large datasets. Conversely, variational inference (Blei and Jordan, 2004) is more suitable for large datasets; however, it incurs a trade-off in accuracy as a result of its reliance on approximations.

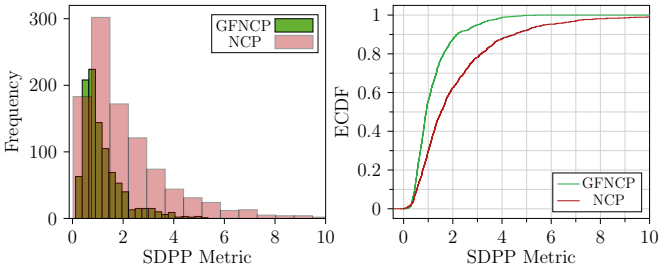

Figure 1: Histogram and Empirical CDF of the SDPP metric (Eq. 32) computed on GFNCP's and NCP's clustering results, trained on MNIST dataset. GFNCP shows a substantial improvement in producing consistent samples across different data orders, as it puts more probability on low SDPP values.

Over the last decade, with the increasing intricacy of data, a variety of clustering solutions have been proposed that utilize deep neural models. A popular family of models, reviewed in Ren et al. (2024) and Zhou et al. (2024), and often referred to as performing *deep clustering*, are unsupervised classifiers trained to discover a finite number of categories. Among these models, methods such as DCN (Yang et al., 2017) and ClusterGAN (Mukherjee et al., 2019) are designed to simultaneously tackle both data representation and clustering tasks. Another approach involves a two-step process in which feature learning and clustering are decoupled, with SCAN (Van Gansbeke et al., 2020) and DDC (Ren et al., 2020) being notable examples. Importantly, at test time, these models are

Proceedings of the 28th International Conference on Artificial Intelligence and Statistics (AISTATS) 2025, Mai Khao, Thailand. PMLR: Volume 258.

limited to generating (soft) assignments for *individual* data points, since they are restricted by their reliance on pre-learned categories and their inability to model point interactions. Note that this is also the case for DeepDPM (Ronen et al., 2022), despite its ability to infer the number of clusters present in the training (but not test) data.

Our focus in this paper is on a different category of models, which address the more ambitious task of *jointly* modeling the clustering posterior for set-structured data of arbitrary size. This task is more challenging, as it involves not only learning the structure of individual data points but also the correlation structure among the points of a dataset of any size. Motivated by this need, there has been consistent progress in recent years in amortized inference methods within the framework of probabilistic clustering, accommodating a set-structured input. Amortized inference (Gershman and Goodman, 2014) refers to the process of training a neural network that can infer the posterior distribution of latent variables based on observations drawn from a generative model. In this setting, Lee et al. (2019a) introduced the Set Transformer (ST), an attention-based network specifically designed to model point interactions within a set. This architecture was used to amortize the inference over the *parameters* of a mixture model, but was restricted to a fixed number of Gaussian components. To avoid these limitations, other models directly amortize the posterior over the joint clustering *labels* of the dataset. DAC (Lee et al., 2019b) and CCP (Pakman et al., 2020a) build on this concept by sequentially generating full clusters, thus enabling more complex prior distributions.

An alternative approach, named Neural Clustering Process (NCP) (Pakman et al., 2020a), sequentially assigns cluster labels to data points. Unlike previous methods, NCP provides assignment probabilities, thereby offering deeper insights into clustering outputs through uncertainty quantification. However, these probabilities are highly sensitive to data-order permutations, failing to preserve a fundamental symmetry of the posterior distribution. The limitations of existing work, as outlined, motivate our current research.

In this paper we propose the GFlowNet-based Clustering Process (GFNCP), a posterior generative clustering model that, given a set-structured input sampled from a (possibly infinite) mixture model, yields clustering assignment samples along with their associated probabilities. In GFNCP, we exploit the framework of GFlowNets (Bengio et al., 2021) to model the clustering task as a sequential generative process in which the final object, a full-data assignment, is constructed by sampling from a learned policy at intermediate states. Notably, we formulate the policy and the learned rewards as energy-based models with shared parameters in an end-to-end framework. Our work differs from previous approaches in several aspects (see Table 1); in particular, unlike NCP, our model encourages invariance under data order permutations (see Fig. 1).

Table 1: Amortized-clustering approaches.

PROPERTY	ST	DAC	CCP	NCP	GFNCP
Data-perm. invariance	✓	✓	✗	✗	✓
Unlimited components	✗	✓	✓	✓	✓
Arbitrary likelihood	✗	✓	✓	✓	✓
Assignment prob.	✗	✗	✗	✓	✓
Well defined posterior	–	✗	✓	✓	✓

Overall, **our contributions are as follows**: (1) We present GFNCP, an amortized clustering method formulated as a GFlowNet sequential generative model, which uses an energy-based joint policy and reward function, allowing an unlimited number of components; (2) We demonstrate that GFNCP surpasses existing methods in clustering performance and also generalizes better to unseen classes; (3) We show that GFNCP exhibits greater consistency across different data orders; (4) Unlike previous works on cluster label amortization, we show that training can be performed without true labels via instance discrimination.[1]

2 RELATED WORK

Amortizing discrete variables. Beyond clustering, models exist for posteriors over permutations (Mena et al., 2018; Pakman et al., 2020b) and network communities (Wang et al., 2024), *inter alia.* When both a generative and an inference model over discrete latents are learned, reparametrization gradients can be used via continuous relaxations (Maddison et al., 2017; Jang et al., 2017). To amortize distributions over discrete factor graphs, Buesing et al. (2020) formulate sampling as a MaxEnt Markov Decision Process. However, this approach fails when there are many ways to generate the same object, as in our setting. Other approaches that leverage nonparametric Bayesian models within neural networks, such as those proposed in (Nalisnick and Smyth, 2017) and (Jiang et al., 2017), develop generative models with latent discrete labels.

Generative Flow Networks (GFlowNets). Introduced in (Bengio et al., 2021) and reviewed in § 3.3, this set of algorithms is designed to train a stochastic policy for sampling composite objects from a tar-

[1]Our code is available at https://github.com/BGU-CS-VIL/GFNCP.

get distribution, following a sequence of actions structured as a directed acyclic graph (DAG). GFlowNets address the challenging setting in which different trajectories in the space of actions can yield the same final state. GFlowNets have connections to variational inference (Malkin et al., 2023) and entropy-regularized reinforcemnt learning (Tiapkin et al., 2024; Deleu et al., 2024), and have been applied to problems such as biological and natural language sequences (Jain et al., 2022; Hu et al., 2024) and combinatorial optimization (Zhang et al., 2023). Models similar to ours, were GFlowNets are conditioned on data, include Deleu et al. (2022); Hu et al. (2023).

3 BACKGROUND

3.1 Generative models of clusters

Consider N data points $\mathbf{x} = \{x_i\}$, and assume they were generated through a probabilistic clustering model of the form

$$\begin{aligned} \alpha_1, \alpha_2 &\sim p(\alpha_1, \alpha_2) \\ N &\sim p(N) \\ c_1 \dots c_N &\sim p(c_{1:N}|\alpha_1) \quad c_i \in \{1 \dots K\} \\ \mu_1 \dots \mu_K | c_{1:N} &\sim p(\mu_{1:K}|\alpha_2) \\ x_i &\sim p(x_i|\mu_{c_i}) \quad i = 1 \dots N. \end{aligned} \tag{1}$$

The generative model introduces discrete random variables c_i, representing the cluster index for each data point x_i. Note that K, the number of clusters, can itself be a random variable. Here α_1, α_2 are hyperparameters and μ_k denotes a parameter vector controlling the distribution of the k-th cluster (e.g., μ_k could include both the mean and covariance of a Gaussian-mixture component). We also assume that the priors $p(c_{1:N}|\alpha_1)$ and $p(\mu_{1:K}|\alpha_2)$ are exchangeable,

$$\begin{aligned} p(c_{1:N}|\alpha_1) &= p(c_{\rho_1} \dots c_{\rho_N}|\alpha_1)\,, \\ p(\mu_{1:K}|\alpha_2) &= p(\mu_{\xi_1} \dots \mu_{\xi_K}|\alpha_2)\,, \end{aligned} \tag{2}$$

where $\{\rho_i\}_{i=1}^N$ and $\{\xi_k\}_{k=1}^K$ are arbitrary permutations over N and K indices, respectively.

Given data points $x_{1:N}$, the central object we aim to model is the posterior $p(c_{1:N}|x_{1:N})$. Of particular interest for us are two properties:

- **Conditional exchangeability:** probabilities should not depend on the order of the data:

$$p(c_{1:N}|x_{1:N}) = p(c_{\rho_1} \dots c_{\rho_N}|x_{\rho_1} \dots x_{\rho_N})\,. \tag{3}$$

- **Marginal consistency:** by definition, the following relationship holds between marginals (for $n = 1 \dots N-1$):

$$p(c_{1:n}|x_{1:N}) = \sum_{c_{n+1}} p(c_{1:n}, c_{n+1}|x_{1:N})\,. \tag{4}$$

3.2 The Neural Clustering Processes

The standard approach to draw samples from the posterior $p(c_{1:N}|x_{1:N})$, MCMC, has two major limitations. First, convergence can be slow and hard to assess. Second, MCMC requires an explicit expression for the generative model $p(x_i|\mu_{c_i})$. A common way out of the latter problem is to assume that $p(x_i|\mu_{c_i})$ follows a Gaussian (perhaps after some data pre-processing, *e.g.*, as proposed in Chang and Fisher III (2013)), a choice that is not justified in general.

To address these limitations, recent advancements have introduced neural amortized models that can generate (approximate) i.i.d. samples from $p(c_{1:N}|x_{1:N})$, without requiring an explicit generative model $p(x_i|\mu_{c_i})$. The Neural Clustering Process (NCP) (Pakman et al., 2020a), is a model for $p(c_{1:N}|x_{1:N})$ that approximates the marginal posteriors

$$p(c_{1:n}|x_{1:N}) = \sum_{c_{(n+1):N}} p(c_{1:N}|x_{1:N})\,, \tag{5}$$

for $n = 1 \dots N-1$, using an energy-based model,

$$p_\theta(c_{1:n}|x_{1:N}) = \frac{e^{-E_n[c_{1:n}, x_{1:N}]}}{Z_n[x_{1:N}]}\,, \tag{6}$$

where $Z_n[x_{1:N}]$ is a normalization constant and θ are the parameters of the energy function E_n. Note that under the generative model (Eq. 1), the marginal posteriors (Eq. 5) depend on all the data points $x_{1:N}$, not just on $x_{1:n}$. Using the approximate marginals (Eq. 6), posterior samples are obtained by fixing a data order and sequentially sampling from each factor in the approximate expansion:

$$p_\theta(\boldsymbol{c}|\boldsymbol{x}) \simeq p(c_1|\boldsymbol{x}) p_\theta(c_2|c_1, \boldsymbol{x}) \dots p_\theta(c_N|c_{1:N-1}, \boldsymbol{x})\,, \tag{7}$$

given by:

$$p(c_1 = 1|\boldsymbol{x}) = 1 \tag{8}$$

$$p_\theta(c_n|c_{1:n-1}, \boldsymbol{x}) = \frac{p_\theta(c_{1:n}|\boldsymbol{x})}{\sum_{c'_n=1}^{K+1} p_\theta(c_1 \dots c'_n|\boldsymbol{x})}, \tag{9}$$

$$= \frac{e^{-E_n[c_{1:n}, x_{1:N}]}}{\sum_{c'_n=1}^{K+1} e^{-E_n[c_1 \dots c'_n, x_{1:N}]}}. \tag{10}$$

Note that the first assignment (Eq. 8) is deterministic. The NCP setup assumes K unique values in $c_{1:n-1}$, so c_n can take $K+1$ values, i.e., x_n can join any existing cluster or form its own new cluster. Interestingly, the normalization constant from Eq. 6 cancels out in the conditionals (Eq. 9-10). This allowed the authors of Pakman et al. (2020a) to train the model via maximum likelihood,

$$\theta = \underset{\theta}{\operatorname{argmax}}\, \mathbb{E}_{p_{\text{data}}(\boldsymbol{c}, \boldsymbol{x})}[\log p_\theta(\boldsymbol{c}|\boldsymbol{x})]\,, \tag{11}$$

without contrastive divergence (Hinton, 2002). A major limitation of the NCP model, however, is that although Eq. 9 assumes the validity of Eq. 4, the latter is not enforced in the architecture or the loss.

Architecture of the energy function. The NCP energy function $E_n[c_{1:n}, x_{1:N}]$ in Eq. 6 is the scalar output of a network that respects the following symmetries of the marginal posteriors $p(c_{1:n}|x_{1:N})$ in Eq. 5:

- Permutations **within a cluster** are preserved by encoding the data points in each cluster as:

$$H_k = \sum_{i:c_i=k} h(x_i) \quad h : \mathbb{R}^{d_x} \to \mathbb{R}^{d_h} \tag{12}$$

- Permutations **between clusters** are preserved, using the cluster invariants H_k, by

$$G_{c_{1:n}} = \sum_{k=1}^{K} g(H_k), \quad g : \mathbb{R}^{d_h} \to \mathbb{R}^{d_g}. \tag{13}$$

- Permutations of the **unassigned data points** are preserved by

$$U_{n+1} = \sum_{i=n+1}^{N} u(x_i)\,, \qquad u : \mathbb{R}^{d_x} \to \mathbb{R}^{d_u}. \tag{14}$$

The functions h, g, u are neural networks. G and U provide distributed, symmetry-invariant representations of the assigned and unassigned data points, respectively, for any N and K. Encodings of this form yield arbitrarily accurate approximations of (partially) symmetric functions (Zaheer et al., 2017; Gui et al., 2021). Using the G, U encodings, the NCP model represents the energy functions in Eq. 6 as

$$E_n[c_{1:n}, x_{1:N}] = f(G_{c_{1:n}}, U_{n+1})\,, \tag{15}$$

where $f : \mathbb{R}^{d_g+d_u} \to \mathbb{R}$ is a neural network.

3.3 Generative Flow Networks

GFlowNets (Bengio et al., 2021, 2023) are a family of models that amortize the cost of sampling over complex discrete objects. Assume we are given a directed acyclic graph (DAG) $(\mathcal{S}, \mathcal{A})$ where $\mathcal{S}$ is a finite set of vertices *(states)* and $\mathcal{A} \subset \mathcal{S} \times \mathcal{S}$ is a set of directed edges *(actions)*, and there is a unique initial state s_0 from which every other state is reachable.

The composite objects of interest are represented by *terminal states* $\mathcal{Z} \subseteq \mathcal{S}$ without outgoing edges. Such objects can be constructed by following a trajectory $\tau = (s_0 \to s_1 \to \ldots \to z)$, where $z \in \mathcal{Z}$. Denoting by $\mathcal{T}$ the set of trajectories, Bengio et al. (2023) define a *trajectory flow* $F : \mathcal{T} \to \mathbb{R}^+$ as an unnormalized probability mass associated to trajectory τ. The *edge flow* is defined as $F(s \to s') = \sum_{\tau : s \to s' \in \tau} F(\tau)$ and the *state flow* as $F(s) = \sum_{\tau : s \in \tau} F(\tau)$.

Markovian GFlowNets treat the generation of a sample in $\mathcal{Z}$ as a Markov sequential decision problem. Samples from $\mathcal{Z}$ are obtained by starting from s_0 and sampling successive actions using a *forward policy*, which represents a distribution over the children of $s \in \mathcal{S} \backslash \mathcal{Z}$ given by

$$P_F(s'|s) = \frac{F(s \to s')}{\sum_{s''} F(s \to s'')}\,, \tag{16}$$

until a terminal state is reached. The goal of training a GFlowNet is to find a policy such that the probability of reaching a terminal state $z \in \mathcal{Z}$ is proportional to a specified or learned *reward* function $R(z) \in \mathbb{R}^+$, i.e.,

$$R(z) \propto \sum_{\tau : z \in \tau} F(\tau)\,. \tag{17}$$

As shown in Bengio et al. (2021), this occurs when using a policy of the form Eq. 16 obtained from edge flows satisfying the flow matching equations

$$\sum_{s' \in \text{ Parent}(s)} F(s' \to s) = \sum_{s'' \in \text{ Children}(s)} F(s \to s'')\,, \tag{18}$$

and whose terminal states satisfy $F(z) \propto R(z)$. Given a parameterized edge flow, the Flow Matching loss (Bengio et al., 2021) imposes equation Eq. 18 in log space. Recent works use other losses such as Detailed Balance (Bengio et al., 2023), Trajectory Balance (Malkin et al., 2022) or SubTB(λ) (Madan et al., 2023), which exploit backward policies over the parents of a state s.

4 GFLOWNET CLUSTERING

Given N data points, we formalize sampling from the clustering posterior using the GFlowNet framework as follows: the initial action uniformly samples a data order, ρ, and assigns the first data point to the first cluster. Then, a sequence of $N-1$ sampled actions allow each successive data point to join an existing cluster or create a new one. A terminal state $z = s_N$ represents a fully clustered dataset. As illustrated in Figure 2, the initial state s_0 edges out into $N!$ branches, all of which meet again in each of the terminal states.

The highly symmetric nature of our setting implies an additional property not present in general GFlowNets. For each terminal state z, there are $N!$ trajectories τ which differ by the data order. However, conditional exchangeability (Eq. 3) implies that all these trajectories are of equal probability. Thus, Eq. 17 becomes:

$$R(z) \propto F(\tau) \qquad \forall z \forall \tau : z \in \tau\,. \tag{19}$$

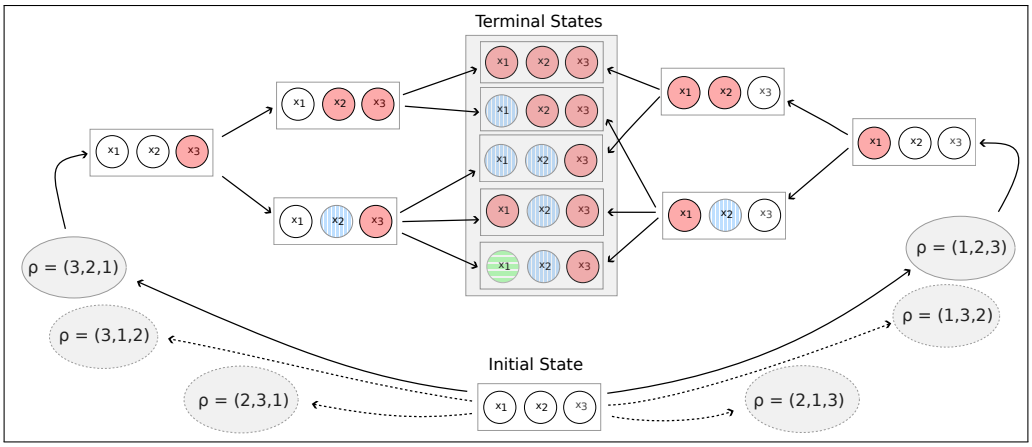

Figure 2: **Clustering via sequential decisions**. Directed Acyclical Graph (DAG) of sequential assignment of cluster labels for a dataset of size $N = 3$. The first action samples uniformly an order ρ for the data points. Each terminal state, corresponding to a fully clustered dataset, receives $N!$ incoming edges, corresponding to all possible orders of the data.

The reward itself, $R(z) \propto p(\boldsymbol{c}|\boldsymbol{x})$, is learned from the data. This setting was studied in Zhang et al. (2022), where an energy-based model for the reward was learned together with the sampling policy. We take a similar approach with one key difference: rather than training separate networks for the policy and the reward, we use a shared parametrization for both. This idea, originally suggested in Bengio et al. (2023), does not appear to have been concretely developed before. To achieve this, we deviate from the standard GFlowNet approach, where the forward policy $P_F(-|s)$ is parameterized as a neural network that takes s as input, and outputs logits for its children's distribution. Instead, we exploit the NCP energy function (Eq. 15) and use the same object to parameterize state flows, edge flows and the reward. Concretely, given N data points, let us define:

$$\hat{E}[c_{1:n}] = \begin{cases} 0 & n = 1, \\ E[c_{1:n}] - \min\limits_{c'_n} E[c_{1:n-1}, c'_n] & 2 \leq n < N, \\ E[c_{1:N}] & n = N, \end{cases}$$

where $E[c_{1:n}] = f(G_{c_{1:n}}, U_{n+1})$ is the energy function given the labels and the data, as described in Eq. 15. We omit indicating the data $x_{1:N}$ in the energy arguments to simplify the notation. We define a state $s_{1 \leq n < N}$ as $(\rho, c_{1:n})$, containing both the data order ρ and the n initial label assignments. The initial transition $s_0 \to s_1$ is special because it uniformly samples a permutation ρ for the data order. We parameterize the initial edge flow as $F[s_0 \to s_1 = (\rho, c_1 = 1)] = 1$, yielding a uniform forward transition (Eq. 16) $P_F[c_1 = 1, \rho | s_0] = \frac{1}{N!}$. For non-initial states, we model both edge and state flows using the *same* function (for $2 \leq n \leq N$):

$$F[c_{1:n-1} \to c_{1:n}, \rho] = F[c_{1:n}, \rho] = e^{-\hat{E}[c_{1:n}]} . \tag{20}$$

This yields the forward transition (Eq. 16)

$$P_F[c_n | c_{1:n-1}, \rho] = \frac{e^{-\hat{E}[c_{1:n}]}}{\sum_{c'_n} e^{-\hat{E}[c_{1:n-1}, c'_n]}} , \tag{21}$$

for $2 \leq n \leq N$, and the order-independent reward

$$R[c_{1:N}] \propto F[c_{1:N}] = e^{-E[c_{1:N}]} . \tag{22}$$

4.1 Objective function

To train the model, we consider a loss function over the network parameters $\boldsymbol{\theta}$ that consists of three terms:

Marginal Consistency loss. Considering Eq. 20 and the fact that non-terminal states have a single parent, the flow matching equations (Eq. 18) become in our case

$$F[c_{1:n-1}, \rho] = \sum_{c_n} F[c_{1:n}, \rho], \tag{23}$$

and our goal is to minimize the discrepancy between the two terms. By substituting $F[c_{1:n}, \rho] = e^{-\hat{E}[c_{1:n}]}$ (as shown in Eq. 20), and transitioning to log space, we aim to minimize the following loss function:

$$\mathcal{L}^{\text{mc}}_{\boldsymbol{\theta}}(\boldsymbol{c}, \boldsymbol{x}) = \sum_{n=2}^{N} \left(\hat{E}[c_{1:n-1}] + \log \Sigma_{c_n} e^{-\hat{E}[c_{1:n}]} \right)^2 . \tag{24}$$

This is the Flow Matching loss (Bengio et al., 2021), which we dub *Marginal Consistency* loss. In our model

Algorithm 1 GFNCP training framework

Input: Training distribution $p_{\text{data}}(\boldsymbol{c}, \boldsymbol{x})$ over data $\boldsymbol{x} = \{x_i\}_{i=1}^N$ and assignments $\boldsymbol{c} = \{c_i\}_{i=1}^N$ (using instance discrimination, § 4.3), hyperparameter $\beta \in [0, 1]$.

Initialize u, h, g, f networks with parameters $\boldsymbol{\theta}$
repeat
  $l \sim \text{Bernoulli}(\beta)$
  if $l = 1$ **then** ▷ Data policy
    Sample $(\boldsymbol{c}, \boldsymbol{x}) \sim p_{\text{data}}(\boldsymbol{c}, \boldsymbol{x})$
    Update $\boldsymbol{\theta}$ with gradient $\nabla_{\boldsymbol{\theta}}[\mathcal{L}^{\text{mc}}_{\boldsymbol{\theta}}(\boldsymbol{c}, \boldsymbol{x}) + \delta\mathcal{L}^{\text{reg}}_{\boldsymbol{\theta}}(\boldsymbol{c}, \boldsymbol{x}) + \lambda\mathcal{L}^{\text{cd}}_{\boldsymbol{\theta}}(\boldsymbol{c}, \boldsymbol{x})]$
  else ▷ Space exploration
    Sample $\boldsymbol{x} \sim p_{\text{data}}(\boldsymbol{x})$
    Uniformly sample $\boldsymbol{c}^U$.
    Update $\boldsymbol{\theta}$ with gradient $\nabla_{\boldsymbol{\theta}}[\mathcal{L}^{\text{mc}}_{\boldsymbol{\theta}}(\boldsymbol{c}^U, \boldsymbol{x}) + \delta\mathcal{L}^{\text{reg}}_{\boldsymbol{\theta}}(\boldsymbol{c}^U, \boldsymbol{x})]$
  end if
until convergence criterion

it is preferred over the other losses mentioned in § 3.3 (Detailed Balance, Trajectory Balance and SubTB(λ)) because Eq. 24 directly enforces the consistency of marginal distributions, which is central to the order invariance proof in § 4.2. The other losses would only enforce this condition asymptotically, achieving equivalence only in the limit of large model capacity and training data.

Reward loss. The normalized reward is:

$$p_{\boldsymbol{\theta}}(c_{1:N}|\boldsymbol{x}) = \frac{e^{-E[c_{1:N}]}}{Z}, \tag{25}$$

where Z is a normalization constant. To learn this function we minimize the negative log-likelihood,

$$\mathcal{L}^{\text{cd}}_{\boldsymbol{\theta}}(\boldsymbol{c}, \boldsymbol{x}) = -\ \log p_{\boldsymbol{\theta}}(c_{1:N}|\boldsymbol{x})\,, \tag{26}$$

whose contrastive divergence gradient is:

$$\nabla_\theta \mathcal{L}^{\text{cd}}_{\boldsymbol{\theta}}(\boldsymbol{c}, \boldsymbol{x}) = \nabla_\theta E[c_{1:N}] - \mathbb{E}_{p_{\boldsymbol{\theta}}(\tilde{c}_{1:N}|\boldsymbol{x})} E[\tilde{c}_{1:N}]\,. \tag{27}$$

In the second term we approximate the expectation using a sample $\tilde{c}_{1:N}$ generated by the learned policy.

Regularization. To regularize the value of the energy function we add the following term:

$$\mathcal{L}^{\text{reg}}_{\boldsymbol{\theta}}(\boldsymbol{c}, \boldsymbol{x}) = (E[c_{1:N}])^2. \tag{28}$$

Training data and policy. We create a training distribution $p_{\text{data}}(\boldsymbol{c}, \boldsymbol{x})$ as described below in § 4.3. Since each training data point $(\boldsymbol{c}, \boldsymbol{x})$ contains a full trajectory, we found it convenient to optimize the losses *off-policy* (except for $\tilde{\boldsymbol{c}}$ in Eq. 27). The full objective is:

$$\begin{aligned}\mathcal{L}_{\boldsymbol{\theta}} = \mathbb{E}_{\pi(\boldsymbol{c}|\boldsymbol{x})p_{\text{data}}(\boldsymbol{x})}[\mathcal{L}^{\text{mc}}_{\boldsymbol{\theta}}(\boldsymbol{c}, \boldsymbol{x}) + \delta\mathcal{L}^{\text{reg}}_{\boldsymbol{\theta}}(\boldsymbol{c}, \boldsymbol{x})] \\ + \lambda\mathbb{E}_{p_{\text{data}}(\boldsymbol{c}, \boldsymbol{x})}[\mathcal{L}^{\text{cd}}_{\boldsymbol{\theta}}(\boldsymbol{c}, \boldsymbol{x})],\end{aligned} \tag{29}$$

where δ, λ are hyperparameters and the policy $\pi(\boldsymbol{c}|\boldsymbol{x})$ is a mixture of $p_{\text{data}}(\boldsymbol{c}|\boldsymbol{x})$ and a uniform random sample of $\boldsymbol{c}$, to assure full support for the flow matching equations Eq. 23. See Algorithm 1.

4.2 Order invariance

Lemma: when Eq. 24 is zero for all trajectories, the probability of reaching a final state is independent of the selected data order ρ, and thus Eq. 19 holds.

Proof. The probability of any trajectory τ is

$$\begin{aligned}P(\tau) =& P_F(c_1)P_F(c_2|c_1)\dots P_F(c_N|c_{1:N-1}) \qquad (30)\\ =& \frac{1}{N!}\frac{\cancel{F(c_{1:2}, \rho)}}{\cancel{\sum_{c'_2} F(c_1, c'_2, \rho)}}\cdots\frac{F(c_{1:N})}{\cancel{\sum_{c'_N} F(c_{1:N-1}, c'_N, \rho)}}\\ \propto& F(c_{1:N}) = e^{-E[c_{1:N}]}, \qquad (31)\end{aligned}$$

where in all the pairwise cancellations we used Eq. 23, which holds when Eq. 24 is zero. The proof is completed by noting that $E[c_{1:N}]$ is invariant under identical simultaneous permutations of $x_{1:N}$ and $c_{1:N}$. □

4.3 Self-supervised learning

While traditional amortized clustering models require ground-truth labels (Pakman and Paninski, 2018; Pakman et al., 2020a; Lee et al., 2019b), we propose a self-supervised approach using instance discrimination (Dosovitskiy et al., 2014; Wu et al., 2018; Chen et al., 2020; He et al., 2020), which assigns unique labels to each data point based on data augmentations.

Computational cost. Like NCP, our model has an inference cost of $O(N)$. Faster models, which sample cluster members in parallel with $O(K)$ cost (Pakman et al., 2020a), introduce latent continuous variables and do not yield sample probabilities, as this requires expensive marginalizations.

5 EXPERIMENTS

We present an extensive evaluation of GFNCP on several datasets, covering synthetic and toy data in § 5.3, and real-world data in § 5.4. We compare GFNCP with other notable existing approaches, including Set Transformer (ST) (Lee et al., 2019a), DAC (Lee et al., 2019b) and NCP (Pakman et al., 2020a). We demonstrate that GFNCP outperforms existing methods in clustering performance and maintains invariance to input data order, supported by both quantitative and qualitative evidence. In all of our experiments we report the average values on three runs, selecting the best model based on the lowest loss value.

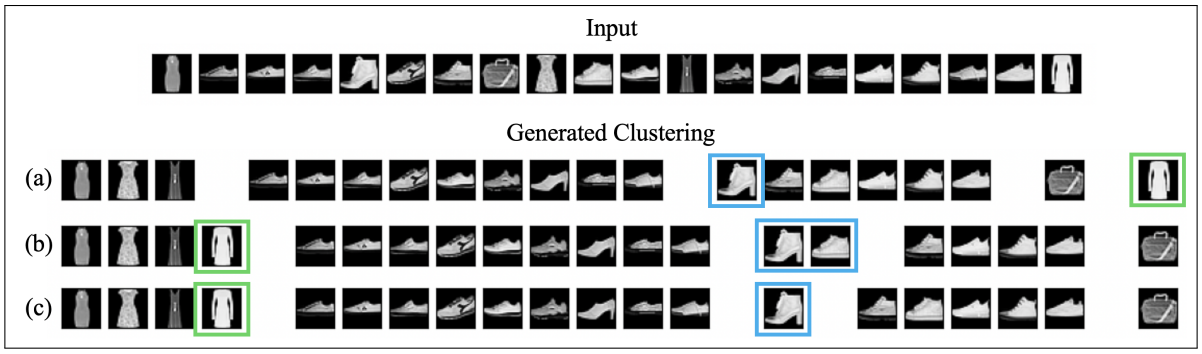

Figure 3: Top-three most-likely clusterings (*i.e.* $p(a) \geq p(b) \geq p(c)$) generated by GFNCP, trained on the Fashion-MNIST dataset. We show that GFNCP's predictions are influenced by point interactions: the model captures the data uncertainty when assigning the dress (green) and the boot (blue), highlighting their distinctiveness within the context of the input set.

5.1 Set-structured input generation

Recall that our method can be trained in an unsupervised manner (§ 4.3). In all our experiments, we generate the training-data distribution $p_{\text{data}}(\boldsymbol{c}, \boldsymbol{x})$ by following Eq. 1 and utilizing a Chinese Restaurant Process (CRP) (Pitman, 2006) for clustering mixtures, where each cluster comprises a sampled data point. Specifically, we adhere to the following steps: (1) the data size N is either fixed or sampled, and a clustering mixture $\boldsymbol{c}$ is drawn using the CRP; (2) for self-supervision, instance discrimination is applied: data for each cluster contains a random data point from the dataset plus its augmentations to match the cluster size. For the test data, we follow the same procedure, except that in step (2), we use the original data points instead of augmentations. See § A.1 for details.

5.2 Data-order invariance metrics

To compare GFNCP with NCP, we introduce two metrics. (i) **Marginal Consistency (MC):** the average of the marginal-consistency loss (Eq. 24) over test pairs $(\boldsymbol{c}, \boldsymbol{x}) \sim p_{\text{data}}(\boldsymbol{c}, \boldsymbol{x})$. (ii) **Standard Deviation over Permutation Probabilities (SDPP):** Given a pair $(\boldsymbol{c}, \boldsymbol{x}) \sim p_{\text{data}}(\boldsymbol{c}, \boldsymbol{x})$, we evaluate the probabilities (Eq. 30) for 500 different permutations $(\boldsymbol{c}_\rho, \boldsymbol{x}_\rho)$; we then compute the standard deviation (SD) of these probabilities and divide by their mean (M), to eliminate scaling bias. The SDPP metric for $(\boldsymbol{c}, \boldsymbol{x})$ is

$$\text{SDPP}(\boldsymbol{c}, \boldsymbol{x}) = \frac{SD(\{p(\boldsymbol{c}_\rho|\boldsymbol{x}_\rho)\}_{\rho \in S^N})}{M(\{p(\boldsymbol{c}_\rho|\boldsymbol{x}_\rho)\}_{\rho \in S^N})} \,. \tag{32}$$

Lower MC and SDPP metrics indicate greater consistency of the model. Note that ST and DAC are excluded from this comparison since they do not produce probabilities for clustering assignments.

Table 2: Clustering results on test sets sampled from MoG, MNIST and Fashion-MNIST (FMNIST), using **a fixed number of six clusters.**

METHOD		ST	DAC	NCP	GFNCP
MoG	NMI	0.91 ± 0.03	**0.96** ± 0.01	0.95 ± 0.01	**0.96** ± 0.00
	ARI	0.91 ± 0.04	0.97 ± 0.01	0.95 ± 0.01	**0.98** ± 0.01
	MC ↓	–	–	10.3 ± 7.7	**3.7** ± 2.5
MNIST	NMI	0.41 ± 0.01	0.27 ± 0.07	0.68 ± 0.24	**0.79** ± 0.08
	ARI	0.24 ± 0.01	0.23 ± 0.06	0.71 ± 0.23	**0.80** ± 0.09
	MC ↓	–	–	34.3 ± 28.1	**12.3** ± 10.5
FMNIST	NMI	0.38 ± 0.01	0.34 ± 0.01	0.50 ± 0.05	**0.53** ± 0.02
	ARI	0.25 ± 0.00	0.20 ± 0.02	0.44 ± 0.07	**0.48** ± 0.04
	MC ↓	–	–	60.9 ± 15.2	**42.6** ± 18.2

Table 3: Clustering results on test sets sampled from MoG, MNIST and Fashion-MNIST (FMNIST), with **an arbitrary number of clusters.**

METHOD		DAC	NCP	GFNCP
MoG	NMI	**0.93** ± 0.01	0.92 ± 0.01	**0.93** ± 0.01
	ARI	0.90 ± 0.01	0.89 ± 0.04	**0.91** ± 0.02
	MC ↓	–	50.8 ± 7.1	**41.1** ± 19.8
MNIST	NMI	0.31 ± 0.03	0.65 ± 0.08	**0.72** ± 0.07
	ARI	0.24 ± 0.03	0.64 ± 0.06	**0.74** ± 0.05
	MC ↓	–	39.3 ± 26.9	**16.4** ± 9.3
FMNIST	NMI	0.33 ± 0.01	0.51 ± 0.10	**0.57** ± 0.03
	ARI	0.18 ± 0.02	0.48 ± 0.11	**0.51** ± 0.07
	MC ↓	–	62.2 ± 18.4	**41.1** ± 13.1

5.3 Toy data

We first demonstrate GFNCP's performance on a two-dimensional Gaussian-Mixture-Model (MoG), MNIST and Fashion-MNIST (FMNIST) (Xiao et al., 2017) datasets, using either fixed or arbitrary number of clusters K. For data generation we follow the procedure in § 5.1. We use $N \sim (100, 1000)$ for set size, and a batch size of 64. For the fixed-clusters experiment we condition the CRP prior on $K = 6$. As the ST

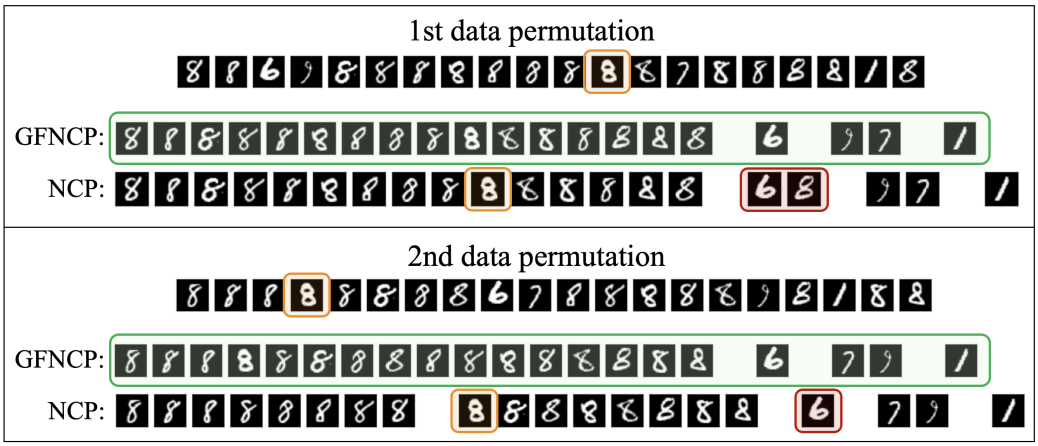

Figure 4: Most-likely assignments generated by GFNCP and NCP, given two different data orders of the same input. GFNCP demonstrates consistent predictions (highlighted in green), whereas NCP's results vary based on the order of the data (with an example shown in red). The digits marked in orange are discussed in the text.

method is mainly designed for embedded images, we utilize a data encoder (similar to the one used in our architecture) for these experiments, to enhance its performance. We evaluate the models on $10,000$ input sets drawn from the test data for MNIST and FMNIST, and on $3,000$ newly-generated input sets for MoG, where each set is of size $N = 300$. We report NMI and ARI metrics (Fahad et al., 2014) for clustering evaluation. For NCP and GFNCP, these metrics are calculated on assignments generated via greedy decoding, where each assignment c_n in the sequence is selected according to the highest forward probability Eq. 21. See Appendix A for more details on the experimental setup and data generation. We also evaluate GFNCP using the average-score approach, where metrics are derived from multiple assignment samples (see Appendix B for more details).

Table 2 and Table 3 present results for fixed and varying number of clusters, respectively. GFNCP consistently outperforms existing methods on different datasets and settings. We observe that DAC's performance deteriorates when the data mixture is less balanced, a factor influenced by the CRP hyperparameter. Note that ST is excluded from Table 3 because it is not designed for varying K. In Fig. 3 we illustrate the top-three most probable clustering generated by GFNCP, highlighting its capability to leverage point interactions in assignment predictions. In Fig. 4 we showcase GFNCP's invariance to input-data order, *e.g.*, when the digit '8' (highlighted in orange) appears earlier in the data sequence, NCP generates an additional cluster for the thicker '8' instances, whereas GFNCP produces consistent results. The gap in order invariance is also shown in Fig. 1, where we present the histogram and empirical CDF (ECDF) of the SDPP metric for both GFNCP and NCP on MNIST.

Table 4: Comparison of GFNCP with existing methods on ImageNet-50/100/200 (IN50/IN100/IN200) using an arbitrary number of clusters.

METHOD		DAC	NCP	GFNCP
IN50	NMI	0.53 ± 0.01	0.58 ± 0.08	**0.62** ± 0.08
	ARI	0.27 ± 0.07	0.43 ± 0.12	**0.48** ± 0.06
	MC ↓	–	65.6 ± 13.9	**60.7** ± 17.8
IN100	NMI	0.53 ± 0.02	0.50 ± 0.10	**0.60** ± 0.05
	ARI	0.13 ± 0.03	0.35 ± 0.07	**0.40** ± 0.05
	MC ↓	–	67.9 ± 15.5	**53.0** ± 13.5
IN200	NMI	0.57 ± 0.03	0.46 ± 0.09	**0.58** ± 0.04
	ARI	0.25 ± 0.01	0.28 ± 0.07	**0.37** ± 0.02
	MC ↓	–	64.4 ± 20.5	**41.6** ± 8.2

5.4 Real-world data

We evaluate GFNCP's clustering performance on images sampled from ImageNet-50/100/200, which are subsets of 50/100/200 classes from ImageNet (Russakovsky et al., 2015), using arbitrary K. Since the models we consider assume that the cluster parameters, μ_k (Eq. 1), have been integrated out, it is natural to explore their ability to generalize to unseen cluster classes. We thus sample training sets from half of the classes, and use the other half to form the test sets. Each image is represented by a 384-dim vector obtained from DINO (Caron et al., 2021). Similar to the toy-data experiments, we follow the procedure in § 5.1 for data generation. In Table 4 we show a significant advantage to GFNCP. We report NMI, ARI and MC metrics, computed on the results over 2500 test sets. In Fig. 5 we present the histogram and ECDF

of the SDPP metric for GFNCP and NCP. More information on the experimental setup can be found in Appendix A.

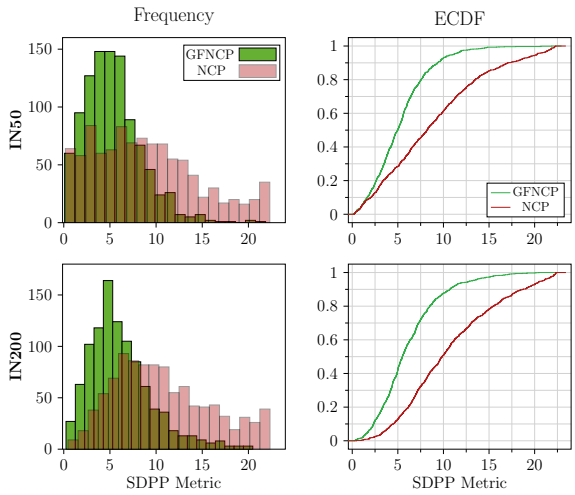

Figure 5: Histogram and Empirical CDF (ECDF) of the SDPP metric, computed on GFNCP's and NCP's clustering results, trained on IN50/200 datasets.

In addition to the mentioned experiments, we evaluate GFNCP in an "online mode". Unlike the primary mode, where the forward transition relies on the full dataset $x_{1:N}$, the model here sequentially predicts assignments for new data points based only on prior points and their cluster assignments, without access to future data. More details about this experiment are available in Appendix C.

6 CONCLUSION

In this paper, we presented GFNCP, a novel approach to amortized clustering that generates data assignments and their associated probabilities while accounting for point interactions and maintaining invariance to the order of the input data. We showed that GFNCP outperforms existing methods in various tasks, in both clustering and consistency metrics.

Acknowledgments

This work was supported in part by the Lynn and William Frankel Center at BGU CS, by Israel Science Foundation Personal Grant #360/21, and by the Israeli Council for Higher Education (CHE) via the Data Science Research Center at BGU. A.P. was supported by the Israel Science Foundation (grant No. 1138/23). I.C. was also funded in part by the Kreitman School of Advanced Graduate Studies, by BGU's Hi-Tech Scholarship, and by the Israel's Ministry of Technology and Science Aloni Scholarship.

References

E. Bengio, M. Jain, M. Korablyov, D. Precup, and Y. Bengio. Flow network based generative models for non-iterative diverse candidate generation. *Advances in Neural Information Processing Systems*, 34:27381–27394, 2021. 2, 4, 5

Y. Bengio, S. Lahlou, T. Deleu, E. J. Hu, M. Tiwari, and E. Bengio. GFlowNet Foundations. *The Journal of Machine Learning Research*, 24(1):10006–10060, 2023. 4, 5

D. M. Blei and M. I. Jordan. Variational Methods for the Dirichlet Process. In *Proceedings of the Twenty-first International Conference on Machine Learning*, ICML, 2004. 1

L. Buesing, N. Heess, and T. Weber. Approximate inference in discrete distributions with Monte Carlo tree search and value functions. In *International Conference on Artificial Intelligence and Statistics*, pages 624–634. PMLR, 2020. 2

M. Caron, H. Touvron, I. Misra, H. Jégou, J. Mairal, P. Bojanowski, and A. Joulin. Emerging properties in self-supervised vision transformers. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 9650–9660, 2021. 8

J. Chang and J. W. Fisher III. Parallel sampling of DP mixture models using sub-cluster splits. In *NIPS*, 2013. 1, 3

T. Chen, S. Kornblith, M. Norouzi, and G. Hinton. A simple framework for contrastive learning of visual representations. In *International conference on machine learning*, pages 1597–1607. PMLR, 2020. 6

T. Deleu, A. Góis, C. Emezue, M. Rankawat, S. Lacoste-Julien, S. Bauer, and Y. Bengio. Bayesian structure learning with generative flow networks. In *Uncertainty in Artificial Intelligence*, pages 518–528. PMLR, 2022. 3

T. Deleu, P. Nouri, N. Malkin, D. Precup, and Y. Bengio. Discrete probabilistic inference as control in multi-path environments. In *The 40th Conference on Uncertainty in Artificial Intelligence*, 2024. 3

O. Dinari, A. Yu, O. Freifeld, and J. Fisher. Distributed MCMC inference in Dirichlet process mixture models using Julia. In *2019 19th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing (CCGRID)*, pages 518–525. IEEE, 2019. 1

O. Dinari, R. Zamir, J. W. Fisher III, and O. Freifeld. Cpu-and gpu-based distributed sampling in dirichlet process mixtures for large-scale analysis. *arXiv preprint arXiv:2204.08988*, 2022. 1

A. Dosovitskiy, J. T. Springenberg, M. Riedmiller, and T. Brox. Discriminative unsupervised feature learning with convolutional neural networks. *Advances in neural information processing systems*, 27, 2014. 6

A. Fahad, N. Alshatri, Z. Tari, A. Alamri, I. Khalil, A. Y. Zomaya, S. Foufou, and A. Bouras. A survey of clustering algorithms for big data: Taxonomy and empirical analysis. *IEEE transactions on emerging topics in computing*, 2(3):267–279, 2014. 8

S. Gershman and N. Goodman. Amortized inference in probabilistic reasoning. In *Proceedings of the annual meeting of the cognitive science society*, volume 36, 2014. 2

S. Gui, X. Zhang, P. Zhong, S. Qiu, M. Wu, J. Ye, Z. Wang, and J. Liu. Pine: Universal deep embedding for graph nodes via partial permutation invariant set functions. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 44(2):770–782, 2021. 4

K. He, H. Fan, Y. Wu, S. Xie, and R. Girshick. Momentum contrast for unsupervised visual representation learning. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 9729–9738, 2020. 6

G. E. Hinton. Training products of experts by minimizing contrastive divergence. *Neural computation*, 14(8):1771–1800, 2002. 4

E. J. Hu, N. Malkin, M. Jain, K. E. Everett, A. Graikos, and Y. Bengio. Gflownet-em for learning compositional latent variable models. In *International Conference on Machine Learning*, pages 13528–13549. PMLR, 2023. 3

E. J. Hu, M. Jain, E. Elmoznino, Y. Kaddar, G. Lajoie, Y. Bengio, and N. Malkin. Amortizing intractable inference in large language models. In *The Twelfth International Conference on Learning Representations*, 2024. 3

M. Jain, E. Bengio, A. Hernandez-Garcia, J. Rector-Brooks, B. F. Dossou, C. A. Ekbote, J. Fu, T. Zhang, M. Kilgour, D. Zhang, et al. Biological sequence design with gflownets. In *International Conference on Machine Learning*, pages 9786–9801. PMLR, 2022. 3

E. Jang, S. Gu, and B. Poole. Categorical reparametrization with gumble-softmax. In *International Conference on Learning Representations (ICLR 2017)*. International Conference on Learning Representations, 2017. 2

Z. Jiang, Y. Zheng, H. Tan, B. Tang, and H. Zhou. Variational deep embedding: An unsupervised generative approach to clustering. In *26th International Joint Conference on Artificial Intelligence, IJCAI 2017*, pages 1965–1972. International Joint Conferences on Artificial Intelligence, 2017. 2

J. Lee, Y. Lee, J. Kim, A. Kosiorek, S. Choi, and Y. W. Teh. Set transformer: A framework for attention-based permutation-invariant neural networks. In *International conference on machine learning*, pages 3744–3753. PMLR, 2019a. 2, 6

J. Lee, Y. Lee, and Y. W. Teh. Deep Amortized Clustering. *arXiv:1909.13433*, 2019b. 2, 6

K. Madan, J. Rector-Brooks, M. Korablyov, E. Bengio, M. Jain, A. C. Nica, T. Bosc, Y. Bengio, and N. Malkin. Learning gflownets from partial episodes for improved convergence and stability. In *International Conference on Machine Learning*, pages 23467–23483. PMLR, 2023. 4

C. Maddison, A. Mnih, and Y. Teh. The concrete distribution: A continuous relaxation of discrete random variables. In *Proceedings of the international conference on learning Representations*. International Conference on Learning Representations, 2017. 2

N. Malkin, M. Jain, E. Bengio, C. Sun, and Y. Bengio. Trajectory balance: Improved credit assignment in gflownets. *Advances in Neural Information Processing Systems*, 35:5955–5967, 2022. 4

N. Malkin, S. Lahlou, T. Deleu, X. Ji, E. Hu, K. Everett, D. Zhang, and Y. Bengio. Gflownets and variational inference. In *The Eleventh International Conference on Learning Representations*, 2023. 3

G. Mena, D. Belanger, S. Linderman, and J. Snoek. Learning latent permutations with gumbel-sinkhorn networks. In *International Conference on Learning Representations*, 2018. 2

S. Mukherjee, H. Asnani, E. Lin, and S. Kannan. Clustergan: Latent space clustering in generative adversarial networks. In *Proceedings of the AAAI conference on artificial intelligence*, volume 33, pages 4610–4617, 2019. 1

E. Nalisnick and P. Smyth. Stick-breaking variational autoencoders. In *International Conference on Learning Representations*, 2017. 2

R. M. Neal. Markov chain sampling methods for Dirichlet process mixture models. *Journal of computational and graphical statistics*, 9(2):249–265, 2000. 1

A. Pakman and L. Paninski. Amortized Bayesian inference for clustering models. *BNP@NeurIPS 2018 Workshop All of Bayesian Nonparametric*, 2018. 6

A. Pakman, Y. Wang, C. Mitelut, J. Lee, and L. Paninski. Neural Clustering Processes. In *Inter-*

national Conference on Machine Learning, 2020a. 2, 3, 6, 13

A. Pakman, Y. Wang, and L. Paninski. Neural permutation processes. In *Symposium on Advances in Approximate Bayesian Inference*, pages 1–7. PMLR, 2020b. 2

J. Pitman. *Combinatorial stochastic processes*. Lecture Notes in Mathematics. Springer-Verlag, Berlin, 2006. 7

Y. Ren, N. Wang, M. Li, and Z. Xu. Deep density-based image clustering. *Knowledge-Based Systems*, 197:105841, 2020. 1

Y. Ren, J. Pu, Z. Yang, J. Xu, G. Li, X. Pu, S. Y. Philip, and L. He. Deep clustering: A comprehensive survey. *IEEE Transactions on Neural Networks and Learning Systems*, 2024. 1

M. Ronen, S. E. Finder, and O. Freifeld. Deepdpm: Deep clustering with an unknown number of clusters. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 9861–9870, 2022. 2

O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, et al. Imagenet large scale visual recognition challenge. *IJCV*, 2015. 8

D. Tiapkin, N. Morozov, A. Naumov, and D. P. Vetrov. Generative flow networks as entropy-regularized RL. In *International Conference on Artificial Intelligence and Statistics*, pages 4213–4221. PMLR, 2024. 3

W. Van Gansbeke, S. Vandenhende, S. Georgoulis, M. Proesmans, and L. Van Gool. Scan: Learning to classify images without labels. In *European conference on computer vision*, pages 268–285. Springer, 2020. 1

Y. Wang, Y. Lee, P. Basu, J. Lee, Y. W. Teh, L. Paninski, and A. Pakman. Amortized probabilistic detection of communities in graphs. *Structured Probabilistic Inference & Generative Modeling workshop at ICML*, 2024. 2

Z. Wu, Y. Xiong, S. X. Yu, and D. Lin. Unsupervised feature learning via non-parametric instance discrimination. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3733–3742, 2018. 6

H. Xiao, K. Rasul, and R. Vollgraf. Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms. *arXiv preprint arXiv:1708.07747*, 2017. 7

B. Yang, X. Fu, N. D. Sidiropoulos, and M. Hong. Towards k-means-friendly spaces: Simultaneous deep learning and clustering. In *international conference on machine learning*, pages 3861–3870. PMLR, 2017. 1

M. Zaheer, S. Kottur, S. Ravanbakhsh, B. Póczos, R. Salakhutdinov, and A. J. Smola. Deep sets. In *Advances in neural information processing systems*, 2017. 4

D. Zhang, N. Malkin, Z. Liu, A. Volokhova, A. Courville, and Y. Bengio. Generative flow networks for discrete probabilistic modeling. In *International Conference on Machine Learning*, pages 26412–26428. PMLR, 2022. 5

D. Zhang, H. Dai, N. Malkin, A. C. Courville, Y. Bengio, and L. Pan. Let the flows tell: Solving graph combinatorial problems with GFlowNets. *Advances in neural information processing systems*, 36:11952–11969, 2023. 3

S. Zhou, H. Xu, Z. Zheng, J. Chen, Z. Li, J. Bu, J. Wu, X. Wang, W. Zhu, and M. Ester. A comprehensive survey on deep clustering: Taxonomy, challenges, and future directions. *ACM Computing Surveys*, 2024. 1

Checklist

1. For all models and algorithms presented, check if you include:
   (a) A clear description of the mathematical setting, assumptions, algorithm, and/or model. [Yes]
   (b) An analysis of the properties and complexity (time, space, sample size) of any algorithm. [Yes]
   (c) (Optional) Anonymized source code, with specification of all dependencies, including external libraries. [Not Applicable]

2. For any theoretical claim, check if you include:
   (a) Statements of the full set of assumptions of all theoretical results. [Yes]
   (b) Complete proofs of all theoretical results. [Yes]
   (c) Clear explanations of any assumptions. [Yes]

3. For all figures and tables that present empirical results, check if you include:
   (a) The code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL). [Yes]
   (b) All the training details (e.g., data splits, hyperparameters, how they were chosen). [Yes]
   (c) A clear definition of the specific measure or statistics and error bars (e.g., with respect to the random seed after running experiments multiple times). [Yes]
   (d) A description of the computing infrastructure used. (e.g., type of GPUs, internal cluster, or cloud provider). [Yes]

4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets, check if you include:
   (a) Citations of the creator If your work uses existing assets. [Yes]
   (b) The license information of the assets, if applicable. [Not Applicable]
   (c) New assets either in the supplemental material or as a URL, if applicable. [Not Applicable]
   (d) Information about consent from data providers/curators. [Not Applicable]
   (e) Discussion of sensible content if applicable, e.g., personally identifiable information or offensive content. [Not Applicable]

5. If you used crowdsourcing or conducted research with human subjects, check if you include:
   (a) The full text of instructions given to participants and screenshots. [Not Applicable]
   (b) Descriptions of potential participant risks, with links to Institutional Review Board (IRB) approvals if applicable. [Not Applicable]
   (c) The estimated hourly wage paid to participants and the total amount spent on participant compensation. [Not Applicable]

A DATA GENERATION AND EXPERIMENTAL SETUP

Here we provide technical details about the experiments mentioned in the paper, including input-data generation and training procedures. Recall that our model consists of four functions, h, g, u and f. We use for them architectures similar to the ones used in the NCP model (Pakman et al., 2020a).

A.1 Set-structured Input Generation

MNIST, Fashion-MNIST and ImageNet. Each data point in our training and test data is a *set* of images or image representations drawn from the original data set, without using the ground-truth labels. We apply self-labeling, as outlined below in step 4, for methods that require labeled data. We use the training split as the data source for generating datasets during training, and the validation split for testing.

The training data is generated as follows:

1. A data size $N \sim (N_{\min}, N_{\max})$ is sampled.
2. A clustering mixture $\boldsymbol{c}$ is generated using the Chinese Restaurant Process (CRP), which determines the number of clusters, K, and the distribution of data points across each group.
3. We then sample K data points from the dataset.
4. Instance discrimination is applied, where each cluster contains the sampled data point and a number of augmentations to the fill up each cluster size, according to the sampled mixture $\boldsymbol{c}$.

For these experiments, we set $(N_{\min}, N_{\max}) = (100, 1000)$, and choose $\alpha = 1$, the hyperparameter controlling the data distribution in the CRP.

Test data is generated by setting $N = 300$ for the data size, following steps 2 and 3, and then sampling original images from the dataset's categories based on the sampled mixture.

As explained in the paper (§ 4.1 and Algorithm 1), during training our model performs space exploration by using policies sampled from a mixture of $p_{\text{data}}(\boldsymbol{c}|\boldsymbol{x})$ and a uniform random sample of $\boldsymbol{c}$, using $l \sim \text{Bernoulli}(\beta)$ to control that mixture. For MNIST and FMNIST we set $\beta = 0.99$, while for datasets with fewer images per category, such as in IN50/100/200, we set $\beta = 0.999$.

2D Mixure of Gaussians (MoG). We create a synthetic dataset using the following procedure for both training and testing data. Similar to the image datasets, for a fixed or sampled data size N, a clustering mixture is generated via the CRP, which determines the number of clusters, K, and the data distribution. We then sample K centroids from the Normal distribution, where each centroid is generated as follows:

$$\mu_k \sim \mathcal{N}([0]_{2\times 1}, \sigma I_{2\times 2}), \tag{33}$$

followed by 2D data-point generation based on the CRP mixture:

$$x_i \sim \mathcal{N}(\mu_k, I_{2\times 2}), \tag{34}$$

using $\sigma = 10$.

For data size values, we set $(N_{\min}, N_{\max}) = (100, 1000)$ for training and $N = 300$ for testing. Additionally, we set $\alpha = 6$ for the CRP, and $\beta = 0.999$ for space exploration.

A.2 Training Procedure

For both the toy-data and real-data experiments we train GFNCP with $5K$ iterations using Adam optimizer, a batch size of 64, no weight decay and a cosine scheduler with an initial learning rate of $5 \cdot 10^{-4}$ and a minimum learning rate of $1 \cdot 10^{-6}$. During training, each data point in the batch is a set-structured input, generated by following the procedure outlined in § A.1. During testing, we calculate the average NMI, ARI, and MC metrics (when applicable) across M input sets drawn from the test data source For MoG we set $M = 3K$; for MNIST and Fashion-MNIST, M is set to $10K$. For ImageNet, we set $M = 2.5K$ due to the lower number of data

points in each category. As for the competitor's training (DAC, Set Transformer and NCP), we set the default hyper-parameters used in their published code or mentioned in their papers.

Each model was trained until convergence based to its objective function. For each experiment we report the metrics' average and standard deviation over three different runs (using three different seeds), selecting the best model based on the lowest loss value.

B AVERAGE SCORE VS. GREEDY DECODING

In this section, we evaluate GFNCP using the average-score approach, where metrics are derived from multiple assignment samples, and compare it to the greedy-decoding approach introduced in the paper. In the average-score approach, for each input set sampled from the test data, we generate assignment predictions by selecting the next assignment at each step in the sequence through sampling, instead of choosing the most probable assignment. Next, we compute the NMI/ARI metrics based on the top-100 assignments, sorted according to their predicted probabilities. We utilize the same models trained and evaluated in the Experiment section of the paper on Mixture of Gaussians (MoG), MNIST, and ImageNet-50/100/200 (IN50/IN100/IN200) datasets, maintaining the same test set size for consistency. Results are presented in Table 5. We observe that GFNCP is still equal or better than NCP across all datasets, and that greedy decoding yields slightly better results in most cases.

Table 5: Comparison of GFNCP and NCP using both greedy decoding and average-score approaches across the MoG, MNIST, and ImageNet-50/100/200 (IN50/IN100/IN200) datasets, with an arbitrary number of clusters.

Dataset	NCP		GFNCP	
	NMI	ARI	NMI	ARI
MoG	**0.92**	**0.89**	**0.92**	**0.89**
MNIST	0.71	0.69	**0.74**	**0.75**
IN50	0.53	0.42	**0.60**	**0.45**
IN100	0.48	0.33	**0.54**	**0.39**
IN200	0.24	0.15	**0.48**	**0.28**

C PREDICTING POSTERIOR FOR UNSEEN DATA

During inference, given an input set of N data points, the forward transition $P_F[c_n|c_{1:n-1}, x_{1:N}]$ (as formulated in Eq. 1) predicts the assignment for x_n based on the previously predicted assignments $c_{1:n-1}$ and the complete set of data points $x_{1:N}$. While this is the primary formulation, GFNCP has also an "online mode", in which it is capable of predicting assignments for previously unseen data points, enabling it to perform the forward transition $P_F[c_n|c_{1:n-1}, x_{1:n-1}]$. This is achieved by setting $U = 0$ (see Eq. 14), thus eliminating the encoding of yet unlabeled points.

To illustrate this functional modality, we performed several experiments where we trained and evaluated GFNCP in online mode (i.e., with $U = 0$) on Mixture of Gaussians (MoG), MNIST, and ImageNet-50/100/200 (IN50/IN100/IN200) datasets. In all of these experiments, we followed the exact same training procedure as described in the paper. In Table 6 we show the performance gap between the online mode and the primary mode (reproduced from Table 4 in the paper). We note that the degradation in performance is negligible.

Table 6: Comparison of GFNCP's online and primary modes on MoG, MNIST and ImageNet-50/100/200 (IN50/IN100/IN200), using an arbitrary number of clusters.

Dataset	**Online Mode**			**Primary Mode**		
	NMI	ARI	MC ↓	NMI	ARI	MC ↓
MoG	0.92	0.90	30.5	0.93 ± 0.01	0.91 ± 0.02	41.1 ± 19.8
MNIST	0.70	0.66	33.8	0.72 ± 0.07	0.74 ± 0.05	16.4 ± 9.3
IN50	0.59	0.46	23.9	0.62 ± 0.08	0.48 ± 0.06	60.7 ± 17.8
IN100	0.55	0.40	54.4	0.60 ± 0.05	0.40 ± 0.05	53.0 ± 13.5
IN200	0.62	0.43	60.0	0.58 ± 0.04	0.37 ± 0.02	41.6 ± 8.2