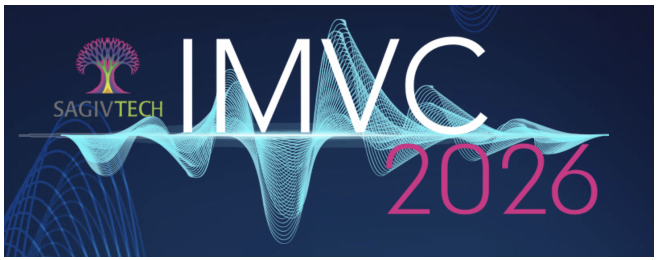
Consistent Amortized Clustering via Generative Flow Networks

Irit Chelly Roy Uziel Oren Freifeld Ari Pakman

Department of Computer Science,
Ben-Gurion University, Israel

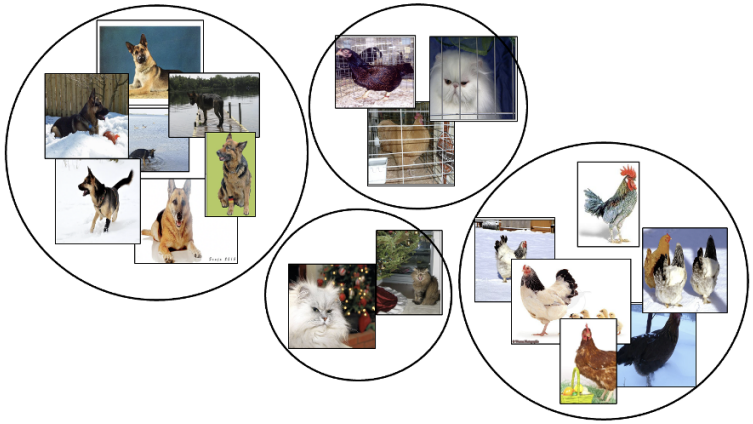
June 2026
IMVC



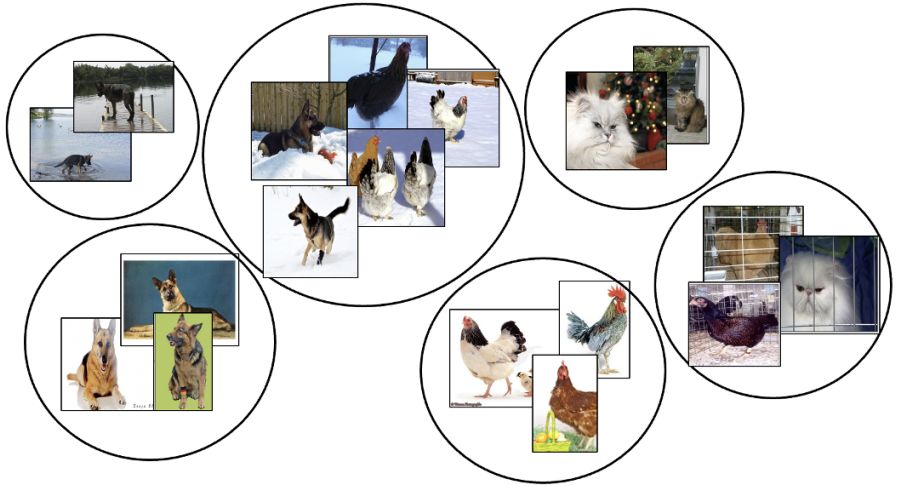
GFNCP: Generative Amortized Clustering

Learn a distribution over clusterings

$P(\boldsymbol{c}|\boldsymbol{x}) = 0.7$



$P(\boldsymbol{c}|\boldsymbol{x}) = 0.2$



GFNCP

A GFlowNet-based model that generates clustering assignments together with their probabilities.

Key properties:

- Multiple clustering hypotheses
- Probabilistic posterior inference
- Permutation-invariant predictions

Amortized Clustering

Goal: learn a model of the clustering posterior $p(c_{1:N} \mid x_{1:N})$ for unseen sets of points.

Key characteristics:

- Each training example is a **set** of points
- Inference is performed using the **entire set**
- Cluster assignments depend on interactions between points
- The model learns how clustering posteriors behave across datasets

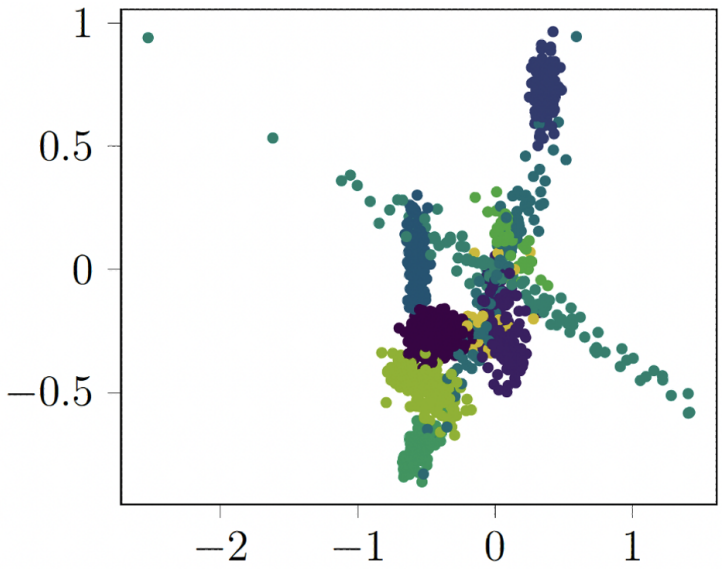
Amortized Inference

Train once, then perform posterior inference for a new set.

New datasets require only a forward pass to infer posterior clusterings.

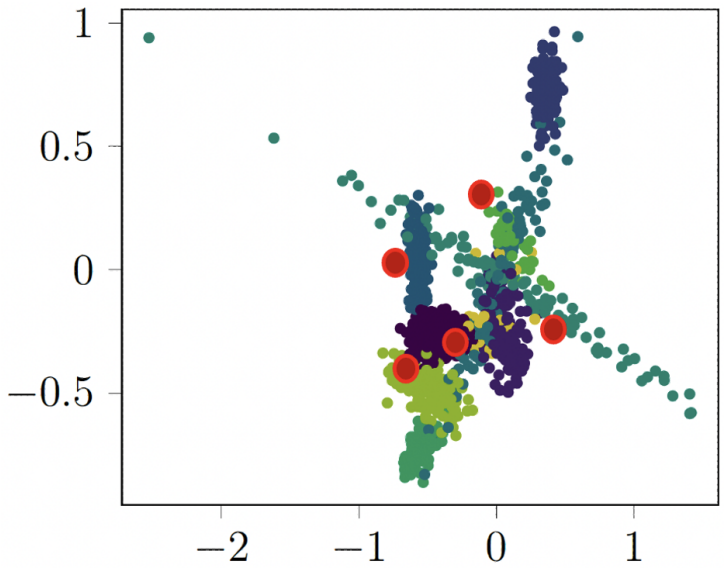
Clustering as Set-Level Inference

Unsupervised Classification



- Learn global cluster structure
- Predict soft assignments independently
- Fixed representation learned during training

Amortized Clustering

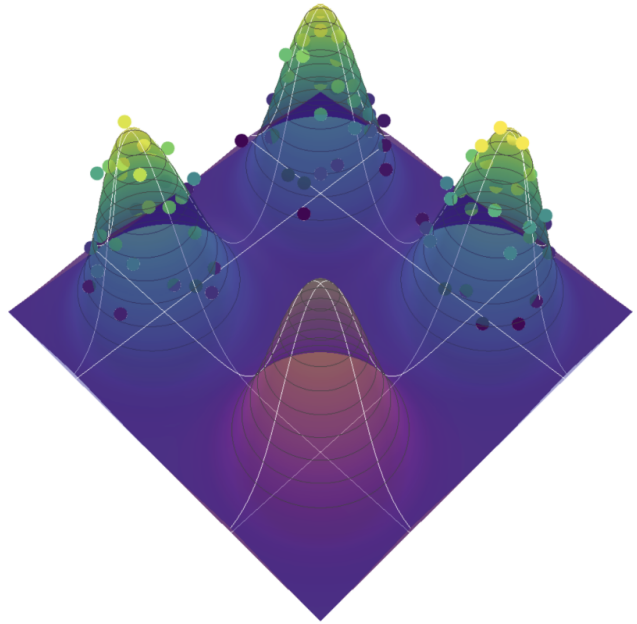


- Model $p(c_{1:N} \mid x_{1:N})$
- Condition on all points jointly
- Infer clustering of a new set
- Can generalize to unseen classes

GFlowNets: Sampling Proportional to Reward

A GFlowNet learns a policy that generates objects according to a reward:

$$p(z) \propto R(z)$$



- Constructs objects **sequentially**
- Naturally captures **multiple modes**
- Generates **diverse high-reward solutions**

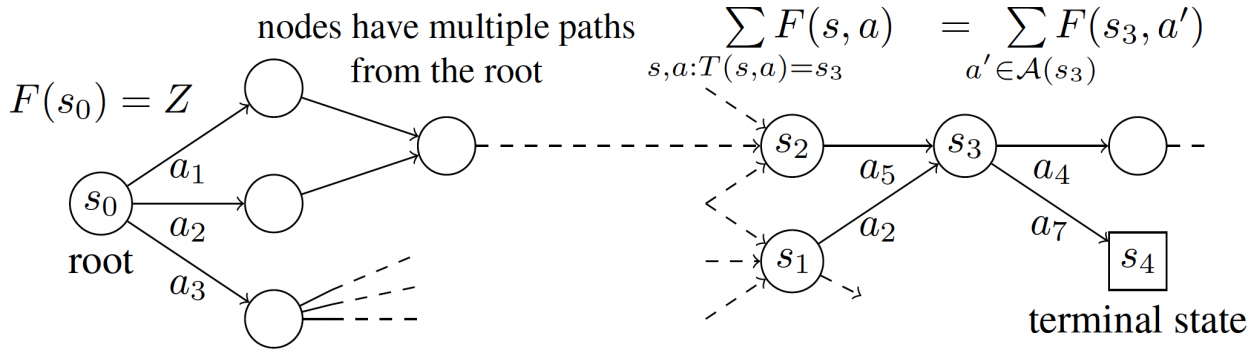
$$s_0 \to s_1 \to \cdots \to z$$

Key idea

Instead of finding a single optimum, GFlowNets learn an entire distribution over good solutions.

GFlowNets: [Bengio *et al.*, NeurIPS '21]

GFlowNets: Flow Consistency



GFlowNets learn flows that satisfy:

$$\sum_{s,a:T(s,a)=s'} F(s,a) = R(s') + \sum_{a' \in \mathcal{A}(s')} F(s',a')$$

- Reward information propagates backward to partial states
- Sampling actions proportionally to flow yields with policy:

$$\tau(a|s) = \frac{F(s,a)}{F(s)}.$$

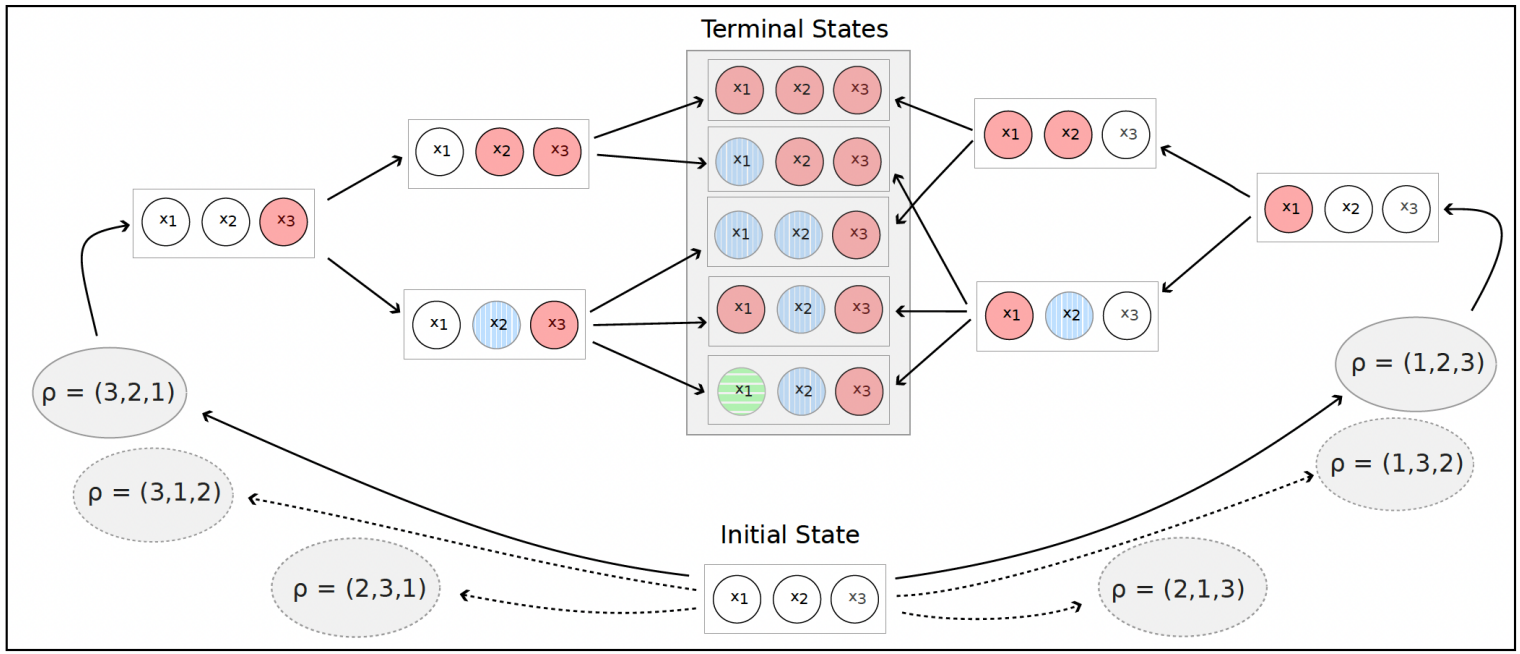
Key consequence

High-reward terminal objects induce high flow through all partial constructions that lead to them.

GFNCP: Clustering as a GFlowNet

We formulate clustering as a sequential construction process:

- State: $s_n = (c_{1:n}, \boldsymbol{x})$, partial assignment of the first n points
- Terminal state: $z = s_N$, a complete clustering assignment
- Reward: $R(z) \propto p_{\boldsymbol{\theta}}(\boldsymbol{c} \mid \boldsymbol{x})$, probability of the final clustering



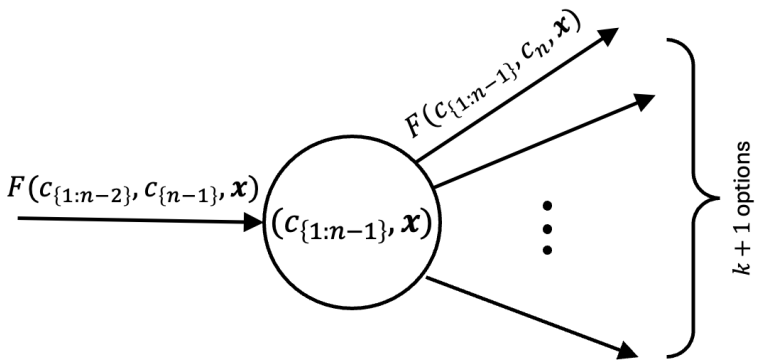
Result

Sampling trajectories generates clusterings from the posterior distribution.

Learning Flow over Cluster Assignments

Our network predicts a flow (energy) for every possible next assignment:

$$F(c_{1:n-1}, c_n, \mathbf{x}) = \log \tilde{p}_\theta(c_{1:n-1}, c_n | \mathbf{x})$$



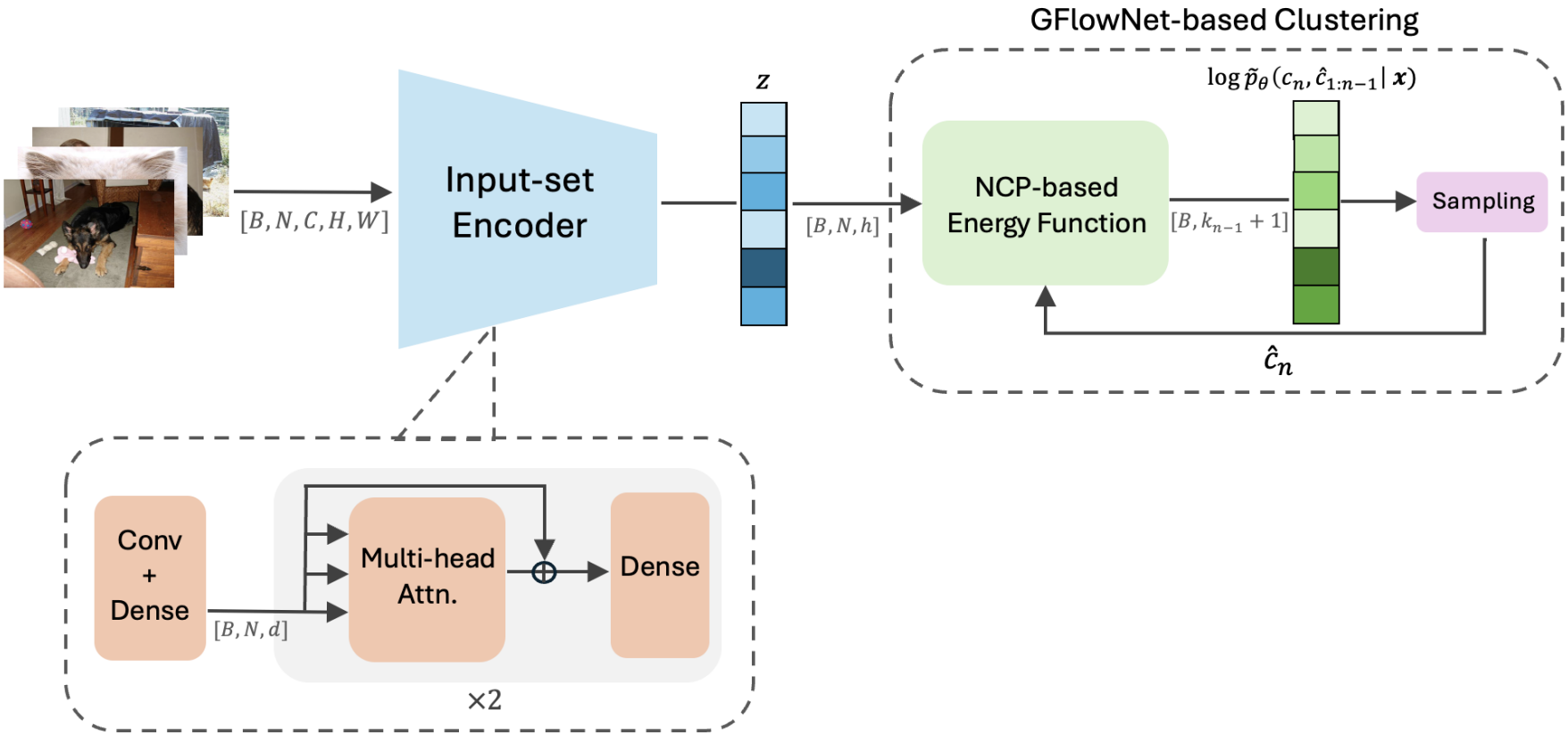
Given a partial assignment $(c_{1:n-1}, \mathbf{x})$:

- Network outputs $K+1$ scores, for each possible action c_n
- Scores define the sampling policy

Interpretation

High flow indicates that extending the current partial clustering is likely to lead to a high-probability solution.

GFNCP Architecture



- Set-based encoder captures interactions between all points
- Network predicts assignment flows for the next point
- Trained end-to-end from clustering rewards
- Reward is learned from data, optimized jointly with the flow equation.

NCP: [Pakman et al., ICML '20]

Flow Matching $\Longleftrightarrow$ Marginal Consistency

Marginal consistency requires

$$p(c_{1:n}|\mathbf{x}) = \sum_{c_{n+1}} p(c_{1:n}, c_{n+1}|\mathbf{x})$$

In our formulation,

$$F(c_{1:n-1}, c_n, \mathbf{x}) = \log \tilde{p}_\theta(c_{1:n-1}, c_n|\mathbf{x})$$

and flow matching becomes

$$\log \tilde{p}_\theta(c_{1:n-1}|\mathbf{x}) = \log \sum_{c_n} \tilde{p}_\theta(c_{1:n-1}, c_n|\mathbf{x})$$

Key observation

Flow consistency in GFlowNets is equivalent to enforcing marginal consistency in clustering.

Why Does GFNCP Become Order Invariant?

Flow matching enforces marginal consistency across all assignment paths.

$$\text{Different assignment orders} \Longrightarrow \text{Same final clustering probability}$$

We prove that when marginal consistency holds,

$$p(\tau) \propto F(c_{1:N})$$

and therefore depends only on the final clustering.

Main consequence

The probability of a clustering is independent of the order in which data points are processed.

Evaluation Setup

Clustering Quality:

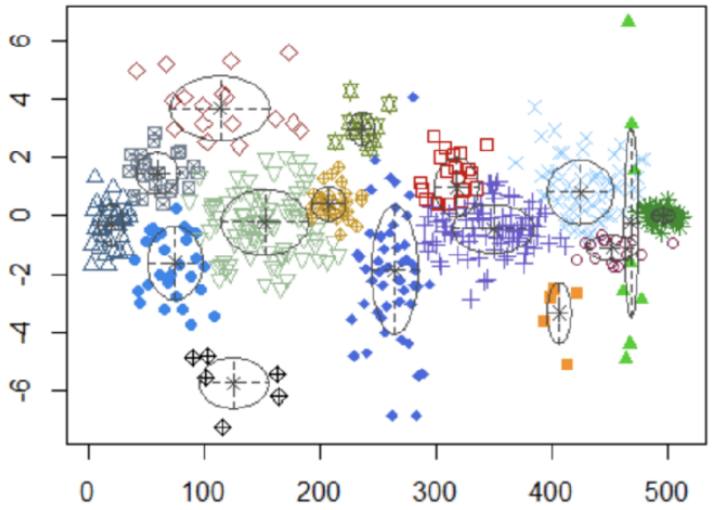
- NMI
- ARI

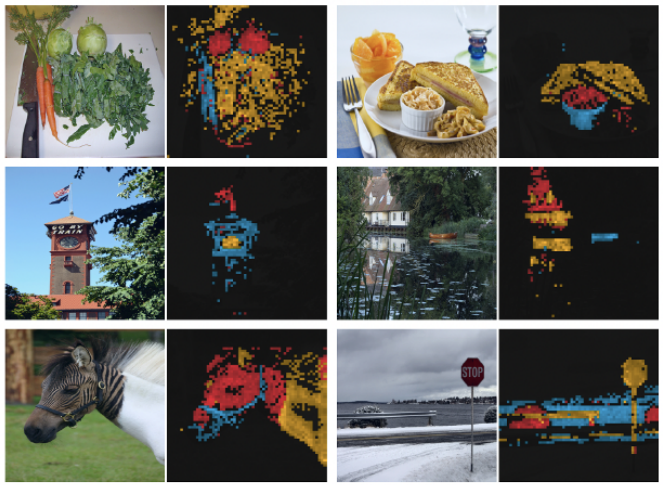
Order Invariance:

- MC Loss
- SDPP (lower is better)

Compared against:

- Set Transformer [Lee *et al.*, ICML '19].
- DAC [Lee *et al.*, arXiv '19]
- NCP [Pakman *et al.*, ICML '20].





Toy Data: Quantitative Results

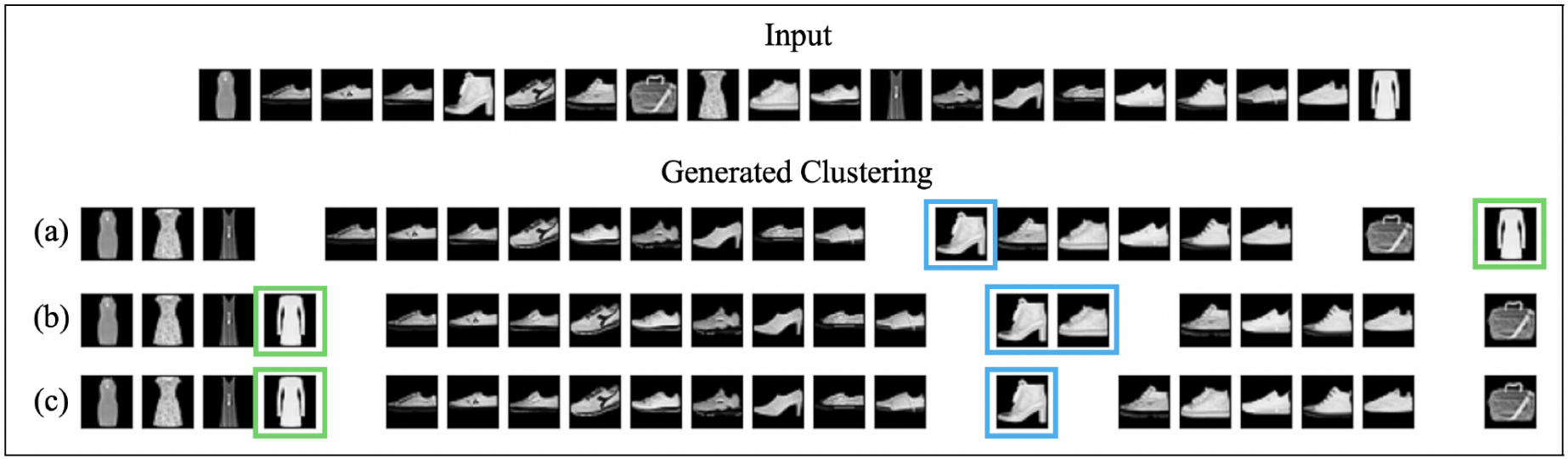
METHOD		DAC	NCP	GFNCP
MoG	NMI	**0.93** ± 0.01	0.92 ± 0.01	**0.93** ± 0.01
	ARI	0.90 ± 0.01	0.89 ± 0.04	**0.91** ± 0.02
	MC ↓	–	50.8 ± 7.1	**41.1** ± 19.8
MNIST	NMI	0.31 ± 0.03	0.65 ± 0.08	**0.72** ± 0.07
	ARI	0.24 ± 0.03	0.64 ± 0.06	**0.74** ± 0.05
	MC ↓	–	39.3 ± 26.9	**16.4** ± 9.3
FMNIST	NMI	0.33 ±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

Observation

GFNCP achieves competitive clustering quality while enforcing order invariance.

Posterior Sampling

Top-3 clustering samples generated by GFNCP:

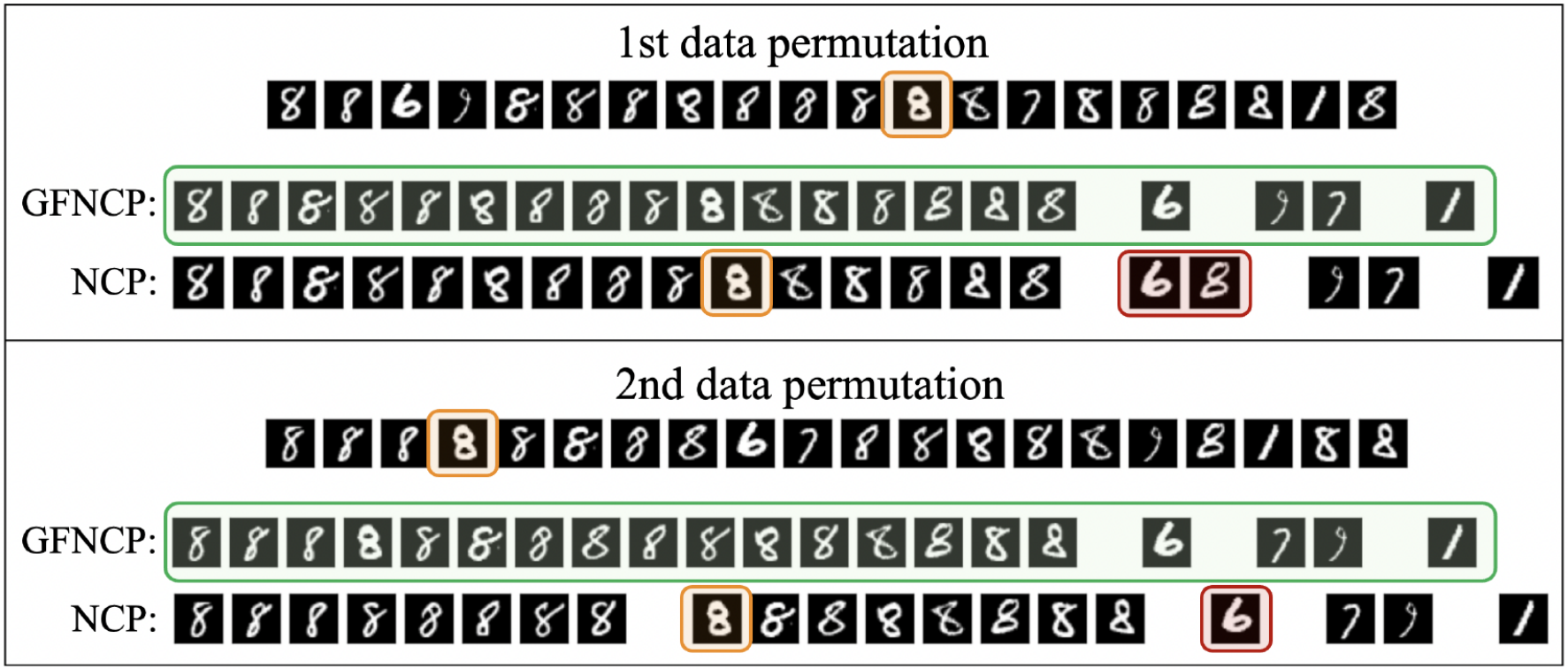


Observation

The model captures multiple plausible clustering modes and its predictions depend on interactions between points.

Order Invariance: Qualitative Results

Most likely clustering under two different permutations of the same group.

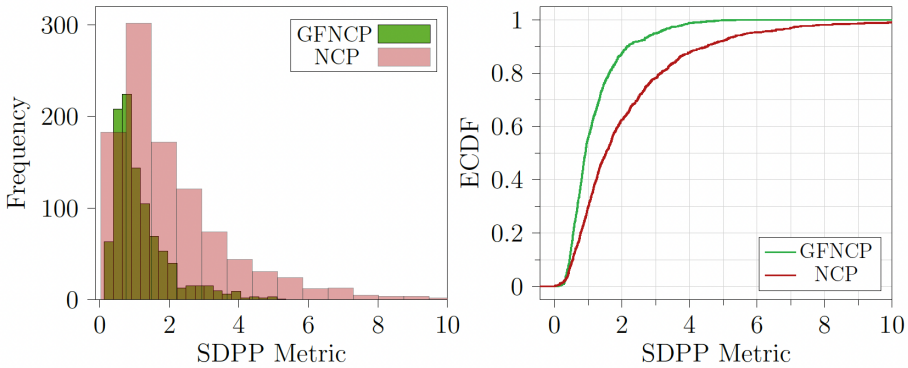


Observation

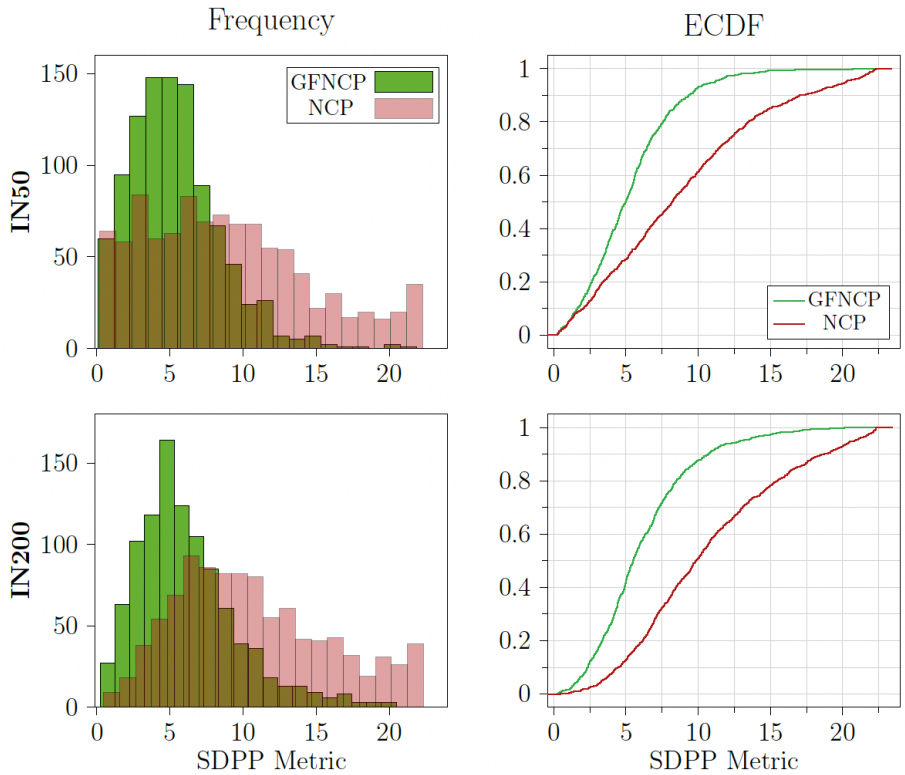
GFNCP produces consistent clusterings, whereas NCP's predictions change with the order of the input points.

Order Invariance: Quantitative Results

MNIST



ImageNet 50/200



Observation

GFNCP substantially reduces SDPP, indicating greater consistency across different input permutations.

Generalization to Unseen Data

ImageNet-50/100/200

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

Observation

GFNCP generalizes to unseen classes and can infer cluster assignments for previously unseen data points.

Takeaways

GFNCP

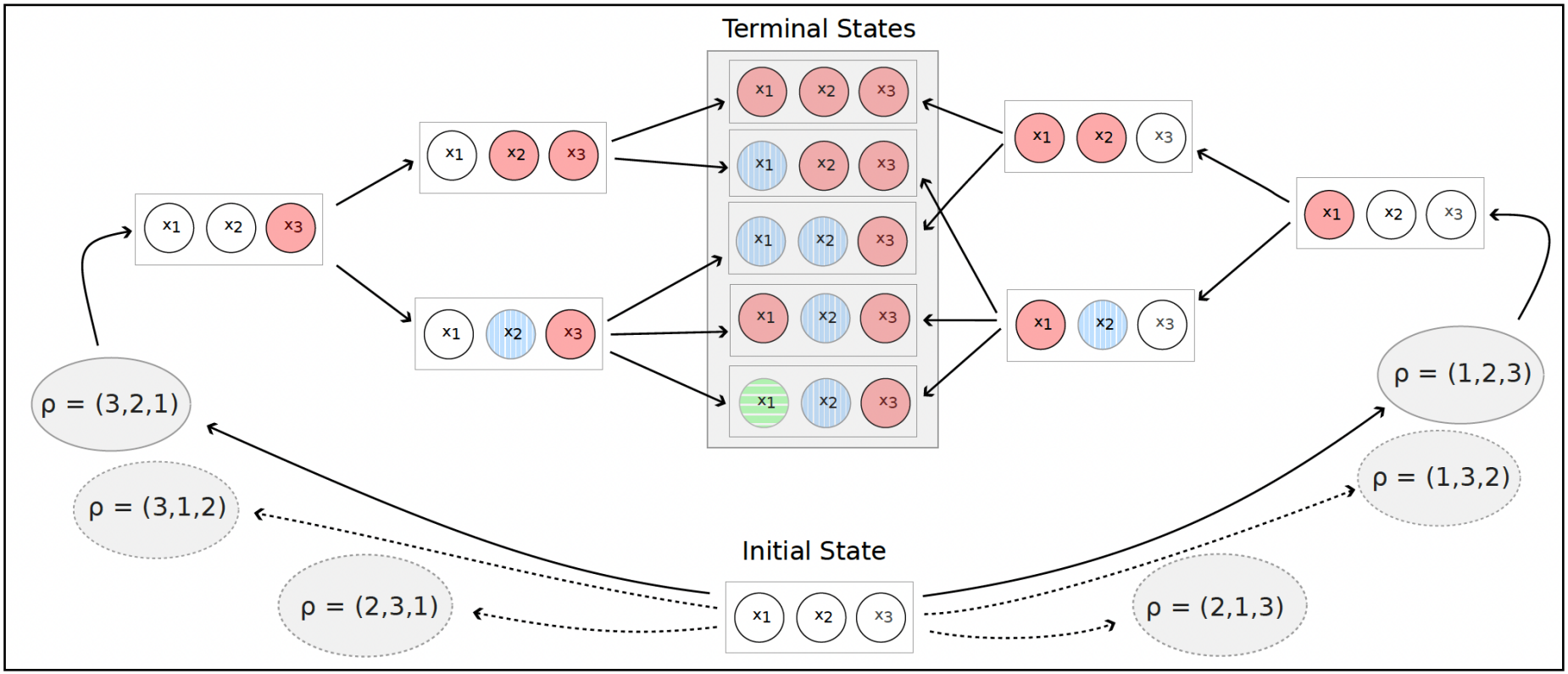
A GFlowNet-based approach for amortized clustering.

- Learns a posterior distribution over clusterings
- Captures interactions between points within a set
- Enforces marginal consistency, yielding order-invariant predictions
- Produces diverse clustering samples with calibrated probabilities
- Achieves strong performance on toy and real-world datasets

Learning how to cluster, not just clustering.

Thank You

Questions?



GFNCP: Consistent Amortized Clustering via Generative Flow Networks

Backup Slides

Challenges

- High reward values: solved by regularization term.
- The output space consists of all possible partitions of the dataset, which grows combinatorially with the number of points.
- Model calibration
- Beam search vs. Greedy decoding
- Computational cost

Objective Function

We train GFNCP end-to-end with the following loss function:

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

where:

$$\mathcal{L}^{\text{mc}}_{\boldsymbol{\theta}}(\boldsymbol{c},\boldsymbol{x}) = \Sigma_{n=2}^{N}(log\ \tilde{p_{\boldsymbol{\theta}}}(c_{1:n-1}|\boldsymbol{x}) - log\Sigma_{c_n}\tilde{p_{\boldsymbol{\theta}}}(c_{1:n-1},c_n|\boldsymbol{x}))^2$$

$$\mathcal{L}^{\text{cd}}_{\boldsymbol{\theta}}(\boldsymbol{c},\boldsymbol{x}) = -\ \lambda(log\ p_{\boldsymbol{\theta}}(c_{1:N}|\boldsymbol{x})) \quad \text{(the reward function)}$$

$$\mathcal{L}^{\text{reg}}_{\boldsymbol{\theta}}(\boldsymbol{c},\boldsymbol{x}) = \delta(log\ \tilde{p_{\boldsymbol{\theta}}}(c_{1:N}|\boldsymbol{x})) \quad \text{(regularize the energy function)}$$

Data Preparation

Each data point in our training and test data is a *set* of images drawn from the original data set, without using the ground-truth labels

Training data generation:

1. Sample a data size $N \sim (N_{\text{min}}, N_{\text{max}})$
2. Generate a clustering mixture $\boldsymbol{c}$ using CRP (let K be the number of clusters)
3. Sample K data points from the dataset (uniformly)
4. Apply instance discrimination to get pseudo-assignments

Data Preparation

Each data point in our training and test data is a *set* of images drawn from the original data set, without using the ground-truth labels

Training data generation:

1. Sample a data size $N \sim (N_{\text{min}}, N_{\text{max}})$
2. Generate a clustering mixture $\boldsymbol{c}$ using CRP (let K be the number of clusters)
3. Sample K data points from the dataset (uniformly)
4. Apply instance discrimination to get pseudo-assignments

We also perform *space exploration* by combining policies sampled from a mixture of $p_{\text{data}}(\boldsymbol{c}|\boldsymbol{x})$ and a uniform random sample of $\boldsymbol{c}$.

Computational Cost

GFNCP has an inference cost of $O(N)$.

Faster models, which sample cluster members in parallel with $O(K)$ cost, introduce latent continuous variables and do not yield sample probabilities, as this requires expensive marginalizations.

Contrastive Divergence

The normalized reward is:

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

where Z is a normalization constant.

To learn this function we minimize the negative log-likelihood,

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

whose contrastive divergence gradient is:

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

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