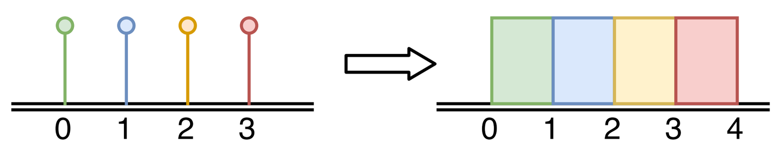
Categorical Normalizing Flows via Continuous Transformations

Phillip Lippe, Efstratios Gavves
University of Amsterdam

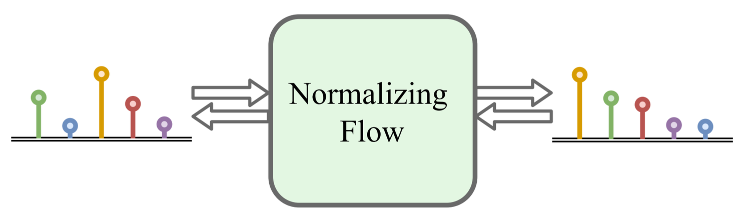
Paper and code

Introduction

- How can we model categorical data with Normalizing Flows?



(Variational) Dequantization
(Theis et al., 2016; Ho et al., 2019)
- Categories ≠ quantized values



Discrete Normalizing Flows
(Tran et al., 2019)
- Difficult optimization
- Limited to permutations

➢ Model categorical distributions by Normalizing Flow in continuous space

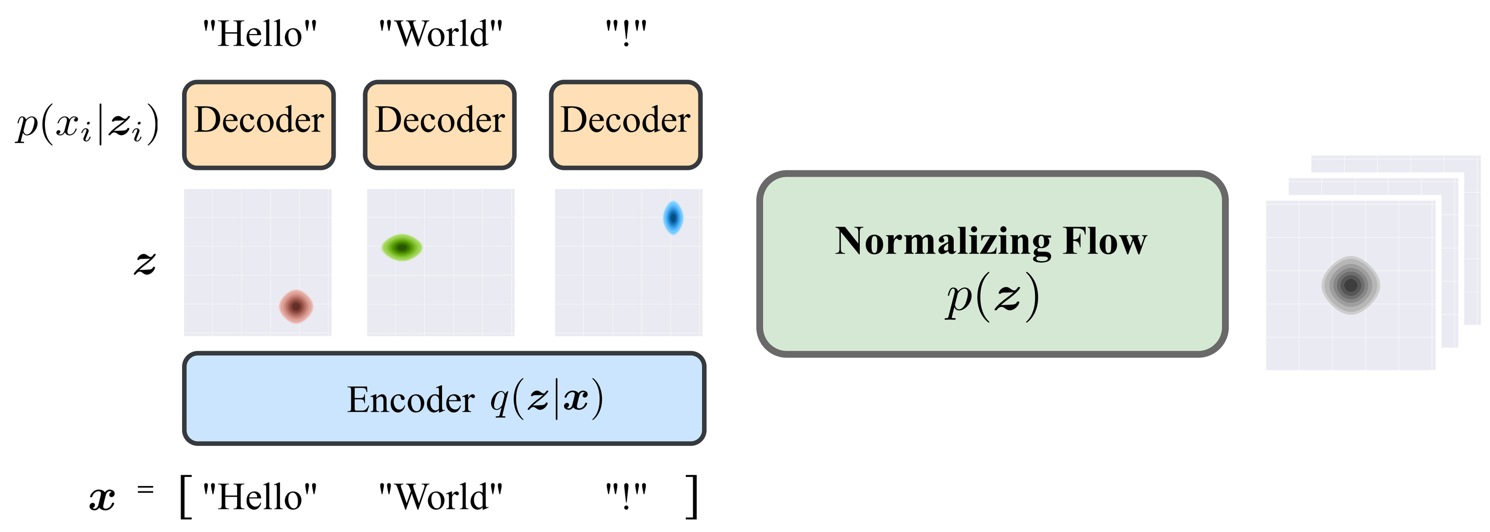
Categorical Normalizing Flows

- Desired properties of an encoding function:
  No loss of information, learnable, smooth, support for higher dimension

➢ Variational Inference with factorized decoder:

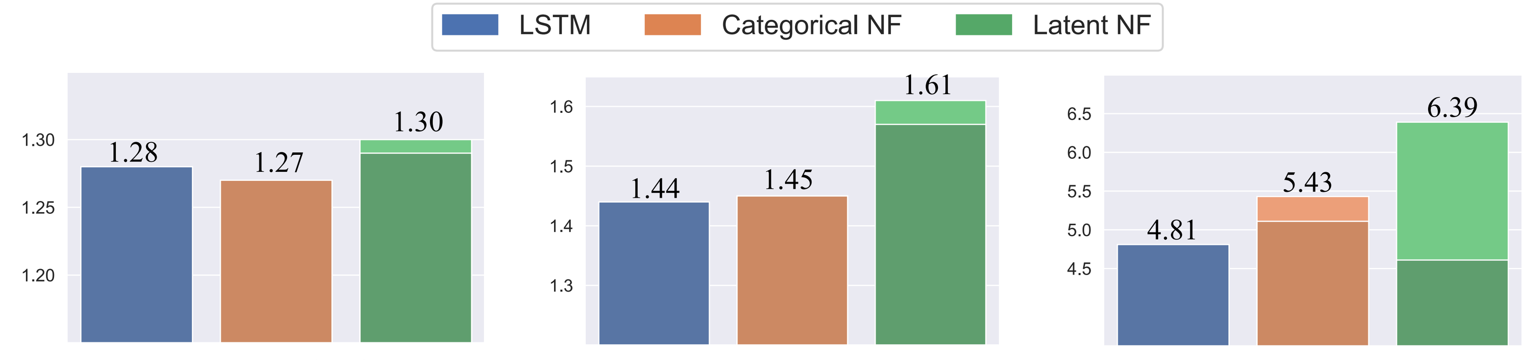
$$p(\boldsymbol{x}) \geq \mathbb{E}_{\boldsymbol{z}\sim q(\cdot|\boldsymbol{x})}\left[\frac{\prod_i p(x_i|\boldsymbol{z}_i)}{q(\boldsymbol{z}|\boldsymbol{x})}p(\boldsymbol{z})\right]$$

- All model complexity in flow, variational inference only used for encoding
- Best encoding function: simple mixture model with exact decoder



Experiments – Language modeling

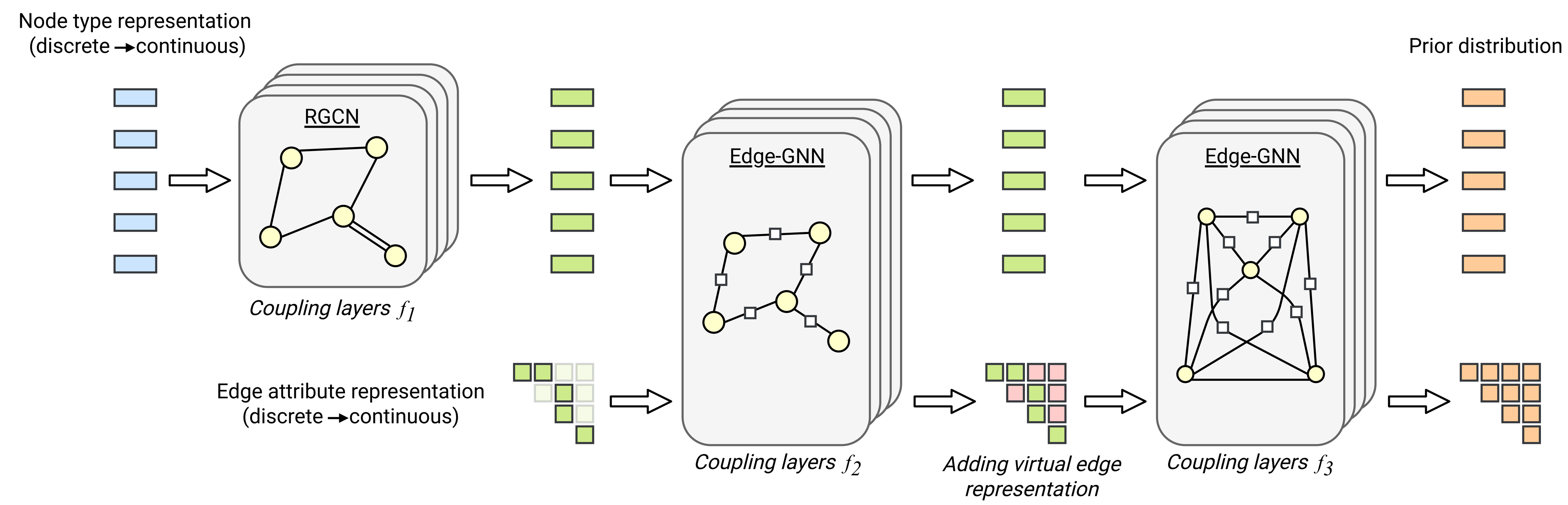
- Considerable better than joint decoder models (LatentNF, Ziegler et al., 2019), especially for complex datasets



Metric: bits per character/word

GraphCNF – Permutation-invariant graph NF

- Stepwise modeling of graph to latent space: node types, edge types, virtual edges
- Permutation-invariant by deploying GNNs, mapping input to a fully-connected graph
- Efficient yet powerful generative model



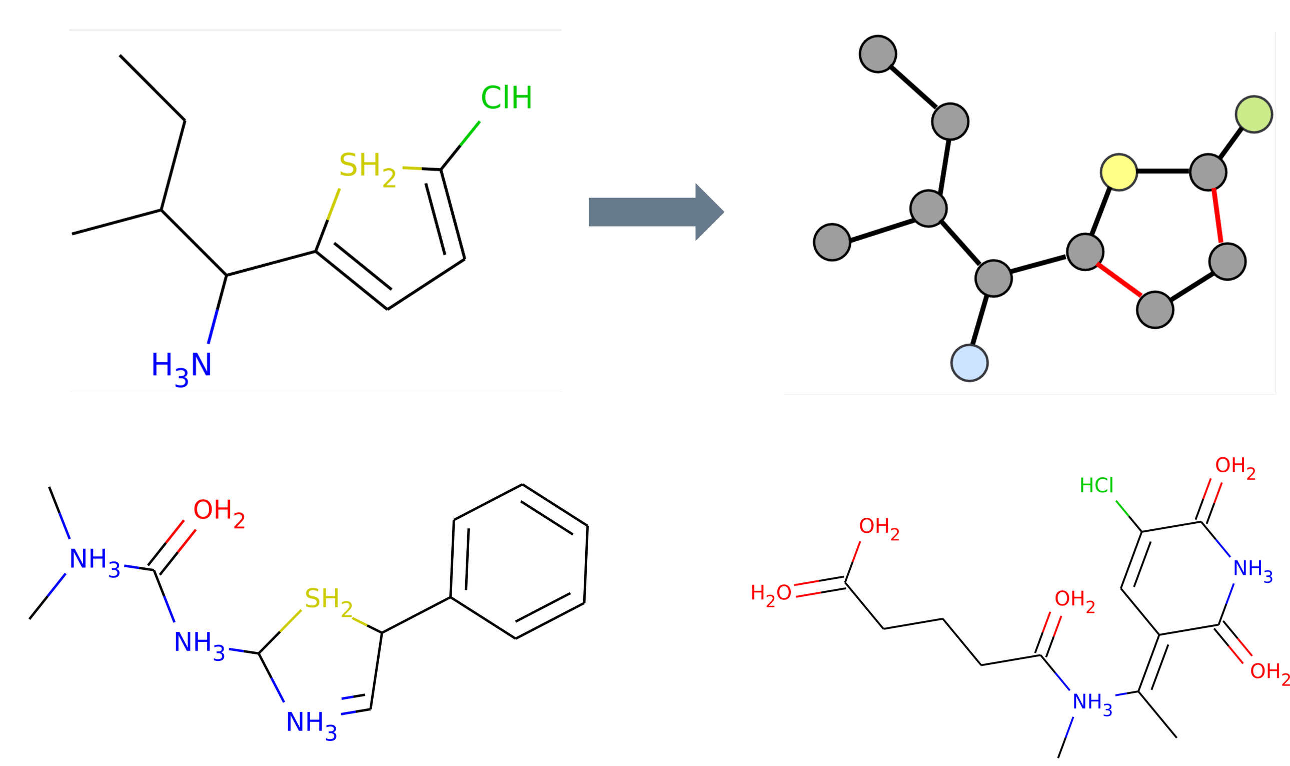
Given a graph $G = (V, E)$:

$$\begin{aligned} z_1^{(V)} &= f_1\left(z_0^{(V)}; E, E_{attr}\right) \\ z_2^{(V)}, z_1^{(E_{attr})} &= f_2\left(z_1^{(V)}, z_0^{(E_{attr})}; E\right) \\ z_3^{(V)}, z_1^{(E)} &= f_3\left(z_2^{(V)}, z_0^{(E)}\right) \end{aligned}$$

Experiments – Molecule generation

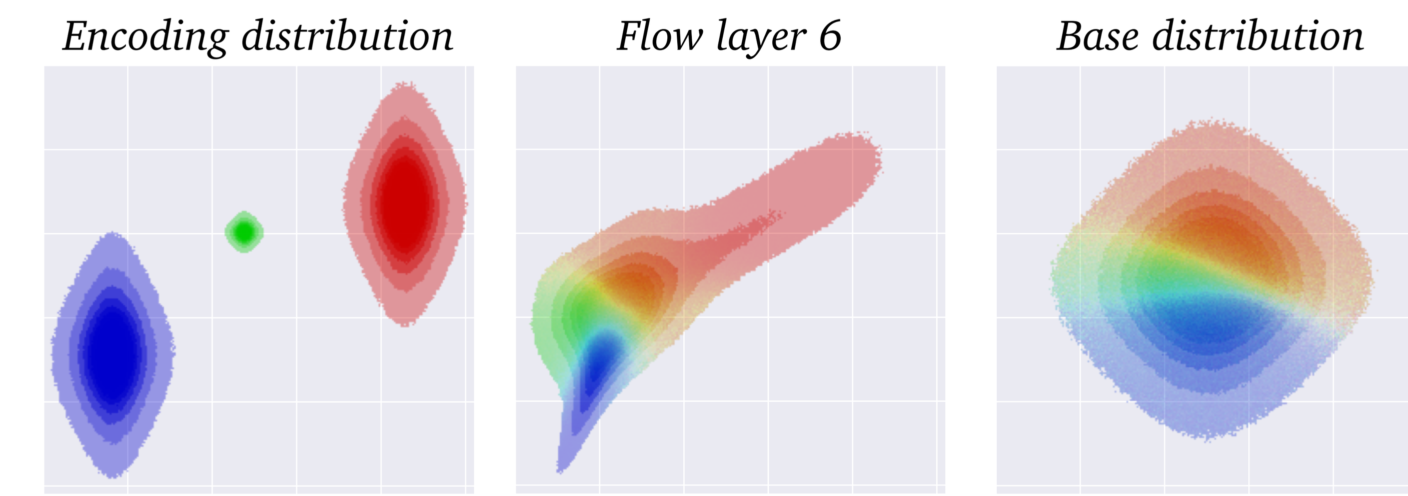
- Molecules: nodes are atoms, edges are bonds
- SoTA for flows without manually encoded rules
- Most common failure case: generating two unconnected valid graphs.

Method	Valid	Unique	Novel	Parallel	Manual
JT-VAE	100%	100%	100%	✗	✓
GraphAF	68%	99.10%	100%	✗	✗
R-VAE	34.9%	100%	-	✓	✗
GraphNVP	42.60%	94.80%	100%	✓	✗
GraphCNF	83.41%	99.99%	100%	✓	✗
+ Sub-graphs	96.35%	99.98%	99.98%	✓	✗



Experiments – Graph Coloring

- Assign color to each node, neighbors need different colors
- Tested different node ordering for GraphRNN based on heuristics
- GraphCNF competitive and faster than best GraphRNN



References

Jonathan Ho et al. (2019). “Flow++: Improving Flow-Based Generative Models with Variational Dequantization and Architecture Design.” In: Proceedings of the 36th International Conference on Machine Learning, pages 2722–2730, Long Beach, California, USA. PMLR.

Lucas Theis et al. (2016). “A note on the evaluation of generative models.” 4th International Conference on Learning Representations, ICLR 2016 - Conference Track Proceedings.

Dustin Tran et al. (2019). “Discrete Flows: Invertible Generative Models of Discrete Data.” In: Advances in Neural Information Processing Systems, pages 14692–14701.

Zachary M. Ziegler et al. (2019). “Latent Normalizing Flows for Discrete Sequences.” In: Proceedings of the 36th International Conference on Machine Learning, pages 7673–7682, Long Beach, California, USA. PMLR.