Autoregressive Diffusion Models with non-Uniform Generation Order

Filip Ekström Kelvinius [1] **Fredrik Lindsten** [1]

Abstract

Diffusion models for discrete data have gained increasing interest lately. Recent methods use an autoregressive formulation, but where the generation order is random. In this work, we turn our attention to the distribution of the generation order. Instead of using a uniform distribution over all possible orders, we propose to limit the distribution for facilitating learning the generative model, while still keeping the benefit of not having to rely on a fixed generation order. We empirically show how limiting the generation order can improve the generative performance in generating molecular graphs.

1. Introduction

Diffusion models (Sohl-Dickstein et al., 2015; Ho et al., 2020; Song & Ermon, 2019) have undoubtedly received much attention lately. Although initial work mostly focused on continuous data, recent work has also considered diffusion models for discrete data (Hoogeboom et al., 2021; Austin et al., 2021; Hoogeboom et al., 2022a).

The recently proposed Autoregressive Diffusion Model (ARDM) (Hoogeboom et al., 2022a) essentially generate data in an autoregressive manner. However, the order in which data is generated is not fixed, but drawn uniformly for all possible permutations of the data elements. This means that the model has to learn all these potential orderings at the same time, which could be argued to be a drawback (Shih et al., 2022). On the other hand, for some types of data, such as graphs, there is not a straightforward way to define a fixed order. In our work, we therefore propose to use a non-uniform distribution over the permutation of the data elements, with the intention of reducing the number of generation orders that the model has to learn, while keeping the property of not having to rely on finding a fixed order for generation. The idea of not using a completely random order is explored also with MAC (Shih et al., 2022), but we have a different focus: while MAC focuses on order agnostic likelihood evaluation, we focus on generative capabilities. Additionally, MAC still relies on the existence of some form of canonical order, which is then maximally reduced under the constraint that all marginals should still be attainable. Contrary to this, we further restrict the support over possible permutations to cut away those that are deemed to be unfavorable from a generative perspective.

To test this new approach, we turn to the task of generating graphs (Li et al., 2018; Chen et al., 2021; Vignac et al., 2023). We apply an implicit distribution over the ordering by defining a generative process where a new node can only be generated once edges between all generated nodes have been assigned, and we show how this approach improves over a uniform distribution over orderings in generating molecular graphs.

2. Background

Let $\mathbf{x} \in \{0, 1, \dots\}^D$ be a D-dimensional discrete random vector with probability mass function $p(\mathbf{x})$. In an autoregressive setting, the log-likelihood factorizes as

$$\log p(\mathbf{x}) = \sum_{t=1}^{D} \log p(x_t | \mathbf{x}_{<t}), \tag{1}$$

where $\mathbf{x}_{<t} = (x_1, x_2, \dots, x_{t-1})$. We can easily sample from the model using D steps of ancestral sampling.

Order Agnostic Autoregressive Models (OA-ARM) (Uria et al., 2014) generate data in an autoregressive manner. However, the order is not fixed, but a random variable which is drawn from the set of all permutations S_D of the indices $\{1, \dots, D\}$. The log-likelihood of a sample, given the permutation σ, is written

$$\log p(\mathbf{x}|\sigma) = \sum_{t=1}^{D} \log p(x_{\sigma(t)} | \mathbf{x}_{\sigma(<t)}), \tag{2}$$

where $\sigma(< t) \coloneqq \{\sigma(i) : i < t\}$. This is analogous to Equation (1), but the order of elements is determined by σ. In this setting, the (unconditional) log-likelihood $\log p(\mathbf{x})$

[1]Division of Statistics and Machine Learning, Department of Computer and Information Science, Linköping University, Linköping, Sweden. Correspondence to: Filip Ekström Kelvinius <filip.ekstrom@liu.se>.

Accepted to ICML workshop on Structured Probabilistic Inference & Generative Modeling

simply becomes an expectation over all permutations,

$$\log p(\mathbf{x}) = \log \mathbb{E}_\sigma \left[p(\mathbf{x}|\sigma) \right]. \tag{3}$$

Autoregressive Diffusion Models (ARDM) (Hoogeboom et al., 2022a) put the OA-ARM in a diffusion model setting. In principle, these models are trained by sampling a permutation σ, a timestep t and then masking out all elements at indices $k \in \sigma(\geq t)$ (e.g, assigning them to a new class "mask"). The model is then trained simultaneously for all permutations that share the same first t indices by having the model predict all the potential next elements $x_{\sigma(t)}$. Mathematically, it is a maximization of

$$\begin{aligned} \log p(\mathbf{x}) &\geq \mathbb{E}_\sigma \sum_{t=1}^{D} \log p(x_{\sigma(t)}|\mathbf{x}_{\sigma(<t)}) \\ &= \mathbb{E}_\sigma D \cdot \mathbb{E}_t \log p(x_{\sigma(t)}|\mathbf{x}_{\sigma(<t)}) \\ &= D \cdot \mathbb{E}_t \mathbb{E}_{\sigma(<t)} \mathbb{E}_{\sigma(t)|\sigma(<t)} \log p(x_{\sigma(t)}|\mathbf{x}_{\sigma(<t)}). \end{aligned} \tag{4}$$

The two outer expectations are approximated by sampling an index t uniformly on $\{1, \dots, D\}$ and a random permutation $\sigma \in S_D$ from some predefined order distribution (e.g., uniform) and only keeping the first $t-1$ elements $\sigma(< t)$. When using a uniform distribution over all permutations, we can compute the inner-most expectation exactly, according to

$$\frac{1}{D-t+1} \sum_{k \in \sigma(\geq t)} \log p(x_k|\mathbf{x}_{\sigma(<t)}),$$

i.e., an average over *all* masked elements.

3. Method

3.1. Non-uniform order distribution

In the formulation of ARDMs, the model has to learn to generate via all possible generation orders, which could arguably be a difficult task. On the other hand, some data does not have an obvious canonical order, and being able to generate in an arbitrary order is therefore desirable.

As a middle ground, we propose a formulation where the distribution over the permutations σ is not uniform, but could be task-specific with the aim of facilitating the learning of the generative model. For example, we could formulate the distribution of $p(\sigma)$ itself in an autoregressive manner,

$$p(\sigma) = \prod_{t=1}^{D} p(\sigma(t)|\sigma(< t)). \tag{5}$$

However, as can be seen in Equation (4), training an ARDM includes sampling from $p(\sigma(< t))$. In the uniform case, this can be done efficiently since it simply corresponds to sampling indices without replacement. Although it is in principle possible to sample from $p(\sigma(< t))$ also in the general autoregressive case in Equation (5), this would require sampling trajectories of σ at each training iteration, which inevitably would slow down training. However, as we will show later, we can construct non-uniform distributions over σ that are efficient to sample from.

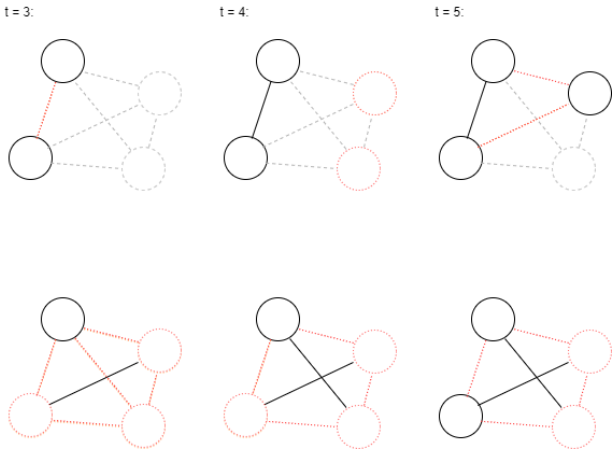

Figure 1. Illustration of the non-uniform (top) and uniform (bottom) generation order for an undirected graph without self-connections. A solid black line indicates that the node/edge has been generated, dashed gray means it is not generated, and dotted red means it has not been generated, but there is a supported generation order where the node/edge is the t:th element to be generated.

3.2. Non-uniform order distribution for graphs

As a concrete example when a non-uniform distribution over the generation orders σ could be beneficial, we turn our focus to generation of graphs. This is a case where the autoregressive formulation with a fixed order does not work without further considerations, as there is no way to number nodes and edges in graphs consistently. Therefore, learning to generate in any, or at least multiple, orders is desirable.

Consider a (directed or undirected) graph with discrete node and edge attributes. As we do not know which nodes that are connected beforehand, we consider generating a complete graph but with one potential edge attribute being "no edge". This effectively allows us to generate the structure of the graph simultaneously with its node and edge attributes. This means that we have to generate up to $n + n^2$ univariate vari-

Table 1. Evaluation metrics on the QM9 dataset, when explicitly modeling hydrogens. The numbers for DiGress (Vignac et al., 2023) are from the original publication, and as we have not included any extra spectral features in our model, we compare with DiGress with and without these features.

MODEL	VALIDITY (%)	UNIQUENESS (%)	ATOM STABLE (%)	MOLECULE STABLE (%)
DIGRESS W/O EXTRA FEATURES	92.3	97.9	97.3	66.8
DIGRESS W/ EXTRA FEATURES	95.4	97.6	98.1	79.8
ARDM-UNIF	86.2	99.0	93.6	30.5
ARDM-NSES	86.2	98.9	95.0	38.4
ARDM-NESN	95.1	98.9	94.6	44.2

ables, where n is the number of nodes in the graph[1]. If we do this uniformly and do not distinguish nodes and edges, we can sample an order $\sigma \sim \mathcal{U}\left(S_{\{1,\dots,n\} \bigcup (\{1,\dots,n\} \times \{1,\dots,n\})}\right)$ (uniformly from all permutations over all nodes and edges). However, this means that the model has to learn to generate, e.g., an edge between two nodes that have not yet been assigned any attributes.

For graphs, we think that it is not necessary to learn to generate in any order, but only a subset of orders. One such subset of orders that we propose is that nodes are generated in a uniformly random order, but before generating a new node, all edges between the latest generated node and all the other generated nodes are generated (similar to (Li et al., 2018)), also in a uniform order. This process, compared with the uniform order, is illustrated in Figure 1. We call this distribution over permutations σ Node-Edges-Node, or NEsN for short.

Algorithm 1 One training step, NEsN

Input: Data sample $\mathbf{x}$, parameters θ, learning rate α
Output: Updated model parameters θ'
Sample $\sigma_V \sim \mathcal{U}(S_n)$
Sample $i \sim \mathcal{U}(1, n)$, number of nodes to keep unmasked
Calculate $m :=$ #edges between node $\sigma_V(i)$ and nodes $\sigma_V(< i)$
Sample $p \sim \mathcal{U}(1, \dots, m)$, number of edges between $\sigma_V(i)$ and $\sigma_V(< i)$ to keep unmasked
if $p \neq m$ **then**
  Sample $\sigma_{E,i}$ uniformly among edges between node i and nodes $\sigma_V(< i)$
  Set deterministically $\sigma_{E,j}$, $\forall j \in \sigma(< i)$
  Compute $\mathcal{L}$ as in Equation (7)
else
  Set deterministically $\sigma_{E,j}$ $\forall j \in \sigma_V(\leq i)$
  Compute $\mathcal{L}$ as in Equation (6)
end if
$\theta' = \theta + \nabla_\theta \mathcal{L}$

Sampling As mentioned before, in the training of ARDMs, we are relying on being able to sample from the marginal distribution of a subset of the permutation order, which if being done autoregressively according to Equation (5) would require t sampling steps to obtain $\sigma(< t)$, and then determine $p(\sigma(t)|\sigma(< t))$.

However, for NEsN, we do not need to do this autoregressively. After sampling t, we uniformly sample a random permutation of the nodes, σ_V, determine the index i of the latest sampled node (based on t), and sample a permutation of the edges that connect node i with the already generated nodes, $\sigma_{E,i}$. We can then mask all nodes in $\sigma_V(> i)$, all edges coming in and out of masked nodes, and a subset of the edges in $\sigma_{E,i}$, depending on t.

An algorithm for sampling the full order σ is presented in Appendix B.

[1]Here, we assume that the number of nodes n has already been sampled, which can be done by, e.g., sampling from the marginal distribution of n over the training data. In principle, we then condition also on n, but omit this for simplicity.

3.3. Loss function when using non-uniform distribution

With this non-uniform order, the inner-most expectation $\mathbb{E}_{\sigma(t)|\sigma(<t)} \log p(x_{\sigma(t)}|\mathbf{x}_{\sigma(<t)})$ in Equation (4) will not result in a sum over all masked elements as in the uniform case. Instead, assume that i nodes have been generated, and with some abuse of notation, let x_i denote node i, and $x_{(i,j)}$ the edge between node i and j. Then, there are two possibilities:

- All edges between generated nodes are already generated, and the next step is to generate a node. The expectation then becomes

$$\frac{1}{\#\sigma_V(> i)} \sum_{k \in \sigma_V(>i)} \log p(x_k|\mathbf{x}_{\sigma(<t)}). \tag{6}$$

- Not all, but $p < \#\sigma_{E,i}$ edges between the i:th node and the other already generated nodes have been generated. The next step is then to generate a new edge, and the

expectation becomes

$$\frac{1}{\#\sigma_{E,i}(> p)} \sum_{(k,l)\in\sigma_{E,i}(>p)} \log p(x_{(k,l)}|\mathbf{x}_{\sigma(<t)}). \quad (7)$$

In other words, the expectation is either over all unmasked nodes, or over all unmasked edges that connect the i:th node and the already generated nodes. An algorithm for a single training step as we have implemented it can be found in Algorithm 1, and some more comments on this implementation can be found in Appendix A.

4. Connection to denoising diffusion

The derivation of the proposed method relies on an autoregressive formulation, akin to Uria et al. (2014) and Hoogeboom et al. (2022a). Hoogeboom et al. (2022a) has shown a connection between the ARDM formulation and denoising diffusion model with absorbing state noise (Austin et al., 2021) by establishing that ARDMs are equivalent to absorbing state noise in continuous time. The destructive process (which usually is an integral part of a diffusion model, but does not have a prominent role in ARDMs) reduces to finding times τ_i when each variable x_i decays into the absorbed/masked state. These time steps will then translate into an order of the variables, and ARDMs model the reverse absorbing process. As noted in their discussion, the (destructive) order will become uniform, and hence, the reverse (generative) order will also be uniform.

To connect our proposed ARDM with non-uniform generation order to denoising diffusion we view this from a slightly different perspective. Note that a diffusion model can be thought of as defining a joint distribution over trajectories $q(\mathbf{x}^{(0:T)})$, where $\mathbf{x}^{(0)}$ denotes an actual data sample and $\mathbf{x}^{(t)}$ increasingly "noisy" samples. Typically, this is defined as a Markovian stochastic processes starting at the data distribution: $q(\mathbf{x}^{(0:T)}) = q(\mathbf{x}^{(0)}) \prod_{t=1}^{T} q(\mathbf{x}^{(t)} \mid \mathbf{x}^{(t-1)})$. However, it is also possible to *define* the distribution over trajectories using a non-Markovian factorization, as

$$q(\mathbf{x}^{(0:T)}) = q(\mathbf{x}^{(0)})q(\mathbf{x}^{(T)}) \prod_{t=1}^{T-1} q(\mathbf{x}^{(t-1)} \mid \mathbf{x}^{(t)}, \mathbf{x}^{(0)}),$$

i.e., as a reverse-time diffusion bridge between a final noisy sample $\mathbf{x}^{(T)}$ and a data sample $\mathbf{x}^{(0)}$. Next, as in standard diffusion models we define a generative Markovian process, yielding a joint distribtuion over trajectories

$$p(\mathbf{x}^{(0:T)}) = p(\mathbf{x}^{(T)}) \prod_{t=0}^{T-1} p(\mathbf{x}^{(t-1)} \mid \mathbf{x}^{(t)}, \mathbf{x}^{(0)}),$$

and train the model by minimizing the Kullback–Leibler divergence $D_{\mathrm{KL}}(q(\mathbf{x}^{(0:T)}) \| p(\mathbf{x}^{(0:T)}))$.

In our auto-regressive setting, we have $T = D$, $\mathbf{x}^{(T)}$ a completely masked (null) state, and $q(\mathbf{x}^{(t-1)} \mid \mathbf{x}^{(t)}, \mathbf{x}^{(0)})$ a distribution which unmasks one element of $\mathbf{x}^{(t-1)}$. Thus, the distribution over orders in which elements are unmasked can be matched between the augmented data distribution q and the generative distribution p, effectively opening up for non-uniform generation orders. Doing so, it can be shown that the KL-based objective is equivalent to the objective proposed above from an autoregressive perspective.

5. Experiments

We evaluate our proposed method on the QM9 dataset (Wu et al., 2018). This dataset contains molecules with up to nine heavy atoms. In our experiment, we explicitly model hydrogens, resulting in $n \leq 29$ atoms for all molecules in the dataset. We use the same metric as Hoogeboom et al. (2022b) and Vignac et al. (2023): Validity (fraction of molecules that are determined valid by RDKit[2]), Uniqueness (fraction of the valid molecules that are unique), Atom stable (fraction of atoms with the correct valency) and Molecule stable (fraction of molecules where all atoms are stable).

We use an autoregressive model where nodes and edges are generated in a random order (ARDM-Unif), which we compare with our proposed order which generates all edges between generated nodes before generating a new node (ARDM-NEsN), and one which first generates all nodes, and then all edges (ARDM-NsEs).

We follow DiGress (Vignac et al., 2023) and use a graph transformer[3] (Dwivedi & Bresson, 2021) which outputs class logits for the node or edge to be generated. The results from our experiments, and a comparison with DiGress are presented in Table 1. In DiGress, they augment their input data with extra structural and spectral features, e.g., cycle counts and eigenvalues of the Laplacian. In our case, however, a noisy graph consists of masked variables (instead of variables which are of potentially the "wrong" class as in DiGress), and we can therefore not compute these extra features. We therefore compare our model (which does not use extra features) with DiGress with and without extra features.

6. Discussion

The results indicate that using a non-uniform generation order improves the generative capabilities compared to the uniform distribution. Note in particular that we improve the validity score, both compared to DiGress without extra features, and to the alternative ARDM implementations. Com-

[2] http://www.rdkit.org

[3] We have used the implementation from the DiGress repository, https://github.com/cvignac/DiGress/

pared to Digress with extra features, we have comparable performance in these metrics. Since the uniqueness score is computed only for valid molecules, this means that ARDM-NEsN generates more unique (and valid) molecules in total than the alternatives. Compared to DiGress, the Molecule stable metric (fraction of molecules where all atoms have the correct valency) is still inferior. One thing to notice, however, is that ARDM requires exactly $n + n(n-1)/2$ sampling steps (as the graphs are undirected and without self-connections). As we sample n from the training data, $\mathbb{E}\left[\#\text{steps}\right] = \mathbb{E}\left[n + n(n-1)/2\right] \approx 176$, which can be compared with DiGress which uses a fixed number of 500 sampling steps (default value in their implementation).

Although in this work we have focused on graphs, one could think of other applications where a non-uniform generation order could be implemented. One example we have in mind is images, where instead of using a fixed order (like a raster-scan) or a completely random order, we could limit the order so that the next pixel to be generated has at least one neighboring pixel that has been already been generated.

7. Conclusion

In this work, we have extended Order-Agnostic Autoregressive Diffusion Models by examining the distribution of generation orders. By not using a uniform distribution over all possible generation orders, we show initial results on how this improves the generative capabilities in generating molecular graphs.

Acknowledgements

This research is financially supported by the Swedish Research Council via the project *Handling Uncertainty in Machine Learning Systems* (contract number: 2020-04122), the Wallenberg AI, Autonomous Systems and Software Program (WASP) funded by the Knut and Alice Wallenberg Foundation, and the Excellence Center at Linköping–Lund in Information Technology (ELLIIT).

References

Austin, J., Johnson, D. D., Ho, J., Tarlow, D., and van den Berg, R. Structured Denoising Diffusion Models in Discrete State-Spaces. In *Advances in Neural Information Processing Systems*, November 2021.

Chen, X., Han, X., Hu, J., Ruiz, F., and Liu, L. Order Matters: Probabilistic Modeling of Node Sequence for Graph Generation. In *Proceedings of the 38th International Conference on Machine Learning*, pp. 1630–1639. PMLR, July 2021.

Dwivedi, V. P. and Bresson, X. A Generalization of Transformer Networks to Graphs, January 2021.

Ho, J., Jain, A., and Abbeel, P. Denoising Diffusion Probabilistic Models. In *Advances in Neural Information Processing Systems*, volume 33, pp. 6840–6851. Curran Associates, Inc., 2020.

Hoogeboom, E., Nielsen, D., Jaini, P., Forré, P., and Welling, M. Argmax Flows and Multinomial Diffusion: Learning Categorical Distributions, October 2021.

Hoogeboom, E., Gritsenko, A. A., Bastings, J., Poole, B., van den Berg, R., and Salimans, T. Autoregressive Diffusion Models. In *International Conference on Learning Representations*, February 2022a.

Hoogeboom, E., Satorras, V. G., Vignac, C., and Welling, M. Equivariant Diffusion for Molecule Generation in 3D. In *Proceedings of the 39th International Conference on Machine Learning*, pp. 8867–8887. PMLR, June 2022b.

Li, Y., Vinyals, O., Dyer, C., Pascanu, R., and Battaglia, P. Learning Deep Generative Models of Graphs, March 2018.

Shih, A., Sadigh, D., and Ermon, S. Training and Inference on Any-Order Autoregressive Models the Right Way. In *Advances in Neural Information Processing Systems*, October 2022.

Sohl-Dickstein, J., Weiss, E. A., Maheswaranathan, N., and Ganguli, S. Deep Unsupervised Learning using Nonequilibrium Thermodynamics, November 2015.

Song, Y. and Ermon, S. Generative Modeling by Estimating Gradients of the Data Distribution. In *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc., 2019.

Uria, B., Murray, I., and Larochelle, H. A Deep and Tractable Density Estimator. In *Proceedings of the 31st International Conference on Machine Learning*, pp. 467–475. PMLR, January 2014.

Vignac, C., Krawczuk, I., Siraudin, A., Wang, B., Cevher, V., and Frossard, P. DiGress: Discrete Denoising diffusion for graph generation. In *The Eleventh International Conference on Learning Representations*, February 2023.

Wu, Z., Ramsundar, B., Feinberg, E. N., Gomes, J., Geniesse, C., Pappu, A. S., Leswing, K., and Pande, V. MoleculeNet: A benchmark for molecular machine learning. *Chemical Science*, 9(2):513–530, January 2018. ISSN 2041-6539. doi: 10.1039/C7SC02664A.

Algorithm 2 Sampling of σ in NEsN

Input: number of nodes, n
Output: Order σ
Sample node ordering σ_V uniformly
for $i = 2$ **to** n **do**
  Uniformly sample edge ordering $\sigma_{E,i}$ among edges between node i and nodes $\sigma_V(< i)$
end for
Concatenate σ_V and all $\sigma_{E,i}$ as
$\sigma = (\sigma_V(1), \sigma_V(2), \sigma_{E,2}, \sigma_V(3), \sigma_{E,3}, \ldots, \sigma_{E,n})$

Algorithm 3 Generation with NEsN

Input: Model parameters θ, empircal distribution over number of nodes in a graph $\hat{p}(n)$
Output: A graph $\mathbf{x}$
Sample $n \sim \hat{p}(n)$
Sample σ using Algorithm 2
Initialize $\mathbf{x}$
for $t = 1$ **to** $n^2 + n$ **do**
  $x_{\sigma(t)} \sim p_\theta(x_{\sigma(t)} | \mathbf{x}_{\sigma(<t)})$
end for

A. Training algorithm

The training algorithm as we implemented it is presented in Algorithm 1. In this algorithm, when we say "set deterministically", we use the fact that the edges in these "sub-orders" will all be unmasked, and the exact order they were "unmasked" will therefore not matter for the loss. Therefore, there is no need to sample these, and to save computation, they can be set according to some predefined order.

B. Sampling of an order in NEsN

An algorithm for sampling an order σ in the NEsN formulation can be found in Algorithm 2.

C. Generation algorithm

An algorithm for generating a graph using NEsN is presented inAlgorithm 3.