TOP-N: EQUIVARIANT SET AND GRAPH GENERATION WITHOUT EXCHANGEABILITY

Anonymous authors
Paper under double-blind review

ABSTRACT

We consider one-shot probabilistic decoders that map a vector-shaped prior to a distribution over sets or graphs. These functions can be integrated into variational autoencoders (VAE), generative adversarial networks (GAN) or normalizing flows, and have important applications in drug discovery. Set and graph generation is most commonly performed by generating points (and sometimes edge weights) i.i.d. from a normal distribution, and processing them along with the prior vector using Transformer layers or graph neural networks. This architecture is designed to generate exchangeable distributions (all permutations of a set are equally likely) but it is hard to train due to the stochasticity of i.i.d. generation. We propose a new definition of equivariance and show that exchangeability is in fact unnecessary in VAEs and GANs. We then introduce *Top-n*, a deterministic, non-exchangeable set creation mechanism which learns to select the most relevant points from a trainable reference set. Top-n can replace i.i.d. generation in any VAE or GAN – it is easier to train and better captures complex dependencies in the data. Top-n outperforms i.i.d. generation by 15% at SetMNIST reconstruction, generates sets that are 64% closer to the true distribution on a synthetic molecule-like dataset, and is able to generate more diverse molecules when trained on the classical QM9 dataset. With improved foundations in one-shot generation, our algorithm contributes to the design of more effective molecule generation methods.

1 INTRODUCTION

In recent years, many architectures have been proposed for learning vector representations of sets and graphs. Sets and graphs are unordered and therefore have the same symmetry group, which offers a fruitful theoretical framework for the development of new models (Bronstein et al., 2021). The opposite problem, i.e., learning to generate sets and graphs from a low-dimensional prior, has been less explored. Yet, it has important applications in drug discovery, for example, where neural networks can be used to learn the space of stable drug-like molecules and generate new candidate drugs.

As set and graph generation are very similar problems, we generally treat them together in this paper[1]. In particular, one can navigate between set and graph representations using Set2Graph functions (Serviansky et al., 2020) or graph neural networks.

There are two main classes of probabilistic decoders for sets: recursive and one-shot. Recursive generators are conceptually simpler, since they add nodes or edges one by one. They are however slow and introduce an order in the points that does not exist in the data. As a result, they need to learn to generate points in any possible order. For this reason, we instead focus on one-shot generation, which allows for more principled designs. These probabilistic decoders can be incorporated into any architecture (generative adversarial networks, variational autoencoders, normalizing flows or diffusion models) and trained to generate sets from a latent vector.

By definition, one-shot generative models for sets feature a layer that maps a vector to an initial set. We group all layers until this one into a *creation module*, and the subsequent layers into an *update module*. The design of the update is well understood: as it maps a set to another set, any

[1] In the following, we only use the terminology of sets in order to avoid overly abstract notations. We discuss the distinctive properties of graph generation (e.g., graph matching problems) when needed.

permutation equivariant neural network suits this task. In contrast, the creation step poses two major challenges, which are i) the need to generate sets of various sizes, and ii) the generation of different points from a single prior vector, which cannot be done with a permutation-invariant function that treats all points symmetrically.

Set creation is typically performed in two steps: first, points are drawn i.i.d. from a low-dimensional normal distribution, and then the latent vector is appended to each point. Architectures based on i.i.d. creation are more flexible and generalise better than set creation alternatives based on multi-layer perceptrons (MLP). Furthermore, when combined with a permutation equivariant update module, they produce exchangeable distributions, a property which is commonly held as the equivalent of permutation equivariance for generative models. However, the stochasticity of i.i.d. creation makes learning difficult, as the network has to learn to reconstruct a set for any combination of values drawn from the base distribution.

In this work, we first extend the usual definition of equivariance (Kondor & Trivedi, 2018): while equivariance is typically defined for a *function*, we say that a *learning algorithm* is equivariant to the action of a group if the training dynamics are independent of the group elements used to represent the data. We use this definition to show that exchangeability is actually not useful in GANs and VAEs, thus explaining why i.i.d. generation is empirically not superior to multi-layer perceptrons.

We then propose a new set creation method, called *Top-n creation*, which relies on a trainable reference set where each of the point has a *representation* $\boldsymbol{r}_i \in \mathbb{R}^c$ and an *angle* $\boldsymbol{\theta}_i \in \mathbb{R}^a$. To generate a set with n elements, we select the n points whose angles have the largest cosine with the latent vector. In order to make this process differentiable, we improve upon the Top-K pooling mechanism (Gao & Ji, 2019) initially proposed for graph coarsening. Top-n can eventually be integrated in any VAE or GAN to form a complete generative model for sets or graphs. As it is deterministic, it is easier to train than i.i.d. generators; in contrary to MLPs, it can extrapolate to larger sets than those seen during training.

We benchmark our creation method on both set and graph generation tasks: it reveals that Top-n is able to better exploit the compositional structure of some problems (on a set version of MNIST), to extrapolate to larger sets and fit more closely the true distribution (on a dataset of synthetic molecule-like structures in 3D), and to generate realistic graphs (on the QM9 chemical dataset). Overall, our work provides new theoretical foundations for one-shot set and graph generation and a more efficient method that contributes to the development of effective learning-based methods for drug discovery.

2 THE ONE-SHOT SET GENERATION PROBLEM

We consider the problem of learning a probabilistic decoder f that maps latent vectors $\boldsymbol{z} \in \mathbb{R}^l$ to multi-sets[2] $\mathcal{X} = \{\boldsymbol{x}_1, ..., \boldsymbol{x}_n\}$ where each $\boldsymbol{x}_i \in \mathbb{R}^d$ for some varying number of points n. Given sample sets from an unknown distribution $\mathcal{D}$, f should be such that, if $\boldsymbol{z}$ is drawn from a prior distribution $p_Z(\boldsymbol{z})$, then $f(\boldsymbol{z})$ has the same law as $\mathcal{D}$. In practice, representing sets is not convenient on standard hardware, and sets are internally represented by matrices $\boldsymbol{X} \in \mathbb{X} = \bigcup_{n \in \mathbb{N}} \mathbb{R}^{n \times d}$ where each row represents a point $\boldsymbol{x}_i \in \mathbb{R}^d$. Algorithms that return a set implicitly assume the use of a function mat-to-set that maps a matrix $\boldsymbol{X}$ to the corresponding set $\mathcal{X}$.

Existing architectures for one-shot set or graph generation use one of the two generation mechanisms described in Figure 1. First of all, a number of points for the set has to be sampled. Two strategies have been proposed:

- Most of the literature assumes that the number of points in each set is known during training. The distribution of the number of points in the training data is computed, and at generation time, a variable n is drawn from this distribution. This method assumes that the latent vector $\boldsymbol{z}$ is independent of the number of points n, so that the generative mechanism writes $p(\mathcal{X}|n, \boldsymbol{z})\ p(n)\ p(\boldsymbol{z})$.
- Kosiorek et al. (2020) instead proposed to learn the value of n from the latent vector using a MLP. This layer is trained via an auxiliary loss, but the predicted value is used only at generation time (the true value of n is used during training). The generative model is in this case $p(\mathcal{X}|\boldsymbol{z}, n)\ p(n|\boldsymbol{z})\ p(\boldsymbol{z})$, i.e., $\boldsymbol{z}$ and n are not independent anymore.

[2]For the sake of simplicity, we will refer to *sets* instead of *multi-sets* in the rest of the paper.

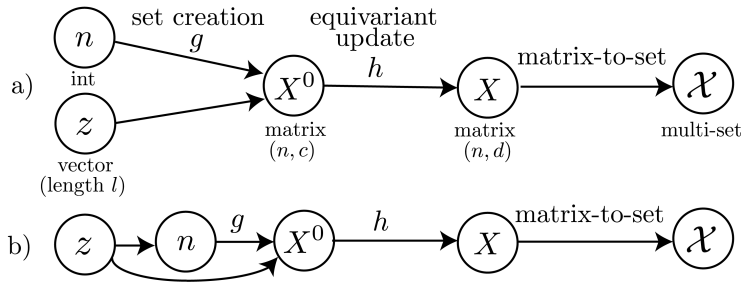

Figure 1: The two graphical models for one-shot set creation. The number of points can either be a) sampled from the dataset distribution or b) learned from the latent vector. In this work, we study ways to parametrize g. For the update h, any equivariant function can be used. Graph generation methods follow a similar structure, except that edge weights are generated in addition to the node features matrices $\boldsymbol{X}^0$ and $\boldsymbol{X}$.

Once n is sampled, one-shot generation models can formally be decomposed into several components: a first function g (that we call *creation function*) maps the latent vector to an initial set $\boldsymbol{X}^0 \in \mathbb{R}^{n \times c}$. This function is usually simple, and is therefore not able to model complex dependencies within each set. For this reason, $\boldsymbol{X}^0$ may then be refined by a second function that we call *update*, so that the whole model can be written $f = \text{mat-to-set} \circ h \circ g$. We now review the parametrizations that have been proposed for the creation and update modules.

2.1 Methods for set creation

MLP based generation Early works (Achlioptas et al. (2018); Zhang et al. (2019b) for sets, Guarino et al. (2017); De Cao & Kipf (2018); Simonovsky & Komodakis (2018) for graphs) proposed to learn a MLP from $\mathbb{R}^l$ to $\mathbb{R}^{n_{\text{max}} c}$, where n_{max} is the largest set size in the training data. The output vector is then reshaped as a $n_{\text{max}} \times c$ matrix, and masked to keep only the first n rows.

As they ignore the symmetries of the problem, MLPs are known to perform very badly at set representation learning (Zaheer et al., 2017). Furthermore, they are only trained to generate up to n_{max} points and cannot extrapolate to larger sets. Despite these limitations, MLPs often perform on par with more complex methods, and still constitute the standard way to generate small graphs (Madhawa et al., 2020; Mitton et al., 2021). We explain in Section 3 this surprising phenomenon.

Random i.i.d generation Along with MLP based generation, the most popular method for set and graph creation is to draw n points i.i.d. from a low dimensional normal distribution, and to concatenate the latent vector to each sample (Köhler et al., 2020; Yang et al., 2019b; Kosiorek et al., 2020; Stelzner et al., 2020; Satorras et al., 2021; Zhang et al., 2021; Liu et al., 2021).

The main advantage of random i.i.d. creation is that it does not constrain the number of points that can be generated. Furthermore, it is exchangeable, i.e., for any permutation π it satisfies $\mathbb{P}(\pi \boldsymbol{X}) = \mathbb{P}(\boldsymbol{X})$. This property is widely considered as the equivalent of equivariance for generative models (Yang et al., 2019a; Biloš & Günnemann, 2021; Kim et al., 2021; Köhler et al., 2020; Li et al., 2020).

Random i.i.d. creation however has several drawbacks: one of them, called the *collision problem* by Krawczuk et al. (2021), happens when several points are drawn close to each other. Unless the network has a large Lipschitz constant, these points will be mapped to final representations that close to each other as well. This may be an issue in some datasets – for example, two atoms are typically never closer than one atomic distance.

Even more importantly, i.i.d. generation makes training very difficult. Generative models based on i.i.d. creation indeed have two stochastic components: the prior vector that is sampled through the reparametrization trick (Kingma & Welling, 2013) and the values for the initial set drawn from the normal distribution – both components make it harder to reconstruct the training data. It is revealing to observe that most methods use a very low-dimensional normal distribution (sometimes in 1d), which is not enough to provide interesting features.

First-n creation While MLP and random generation are the most common methods for set creation, Zhang et al. (2019a) and Krawczuk et al. (2021) use another design: they always start from the same learnable set $\boldsymbol{X}_{\text{ref}} \in \mathbb{R}^{n_{\max} \times c}$, and concatenate the latent vector to each point of this set. Similarly to MLP based creation, they then mask this matrix to keep only the first n rows: we therefore call this method First-n creation.

Despite its simplicity, First-n creation solves most limitations of i.i.d creation: convergence is much faster and this method does not suffer from the collision problem. However, the network is only trained to generate up to $n_{\max}$ points and has no ability to extrapolate to larger sets. Furthermore, selecting the first n rows of the reference set $\boldsymbol{X}_{\text{ref}}$ is an arbitrary mechanism that biases the training procedure (as the first rows are selected more often than the last ones).

Creation methods for graph generation For graph generation, edge weights (or edge features) also need to be learned. To the best of our knowledge, First-n creation has not been used for this purpose yet, and the weights are generated either by a MLP or by sampling i.i.d. normal entries. Alternatively, Bresson & Laurent (2019) and Krawczuk et al. (2021) proposed to generate a set at creation time, and use a Set2Graph update function in order to learn the graph adjacency matrix.

2.2 Methods for set update

Since all creation methods except MLPs can only generate very simple sets and adjacency matrices, additional layers are usually used to refine these objects – we gather these layers into the *update* module. Because these layers map a set or a graph to another set/graph, the update module falls into the standard framework of permutation equivariant representation learning and all existing equivariant layers can be used: Deep sets (Zaheer et al., 2017), self-attention (Vaswani et al., 2017; Lee et al., 2019), Set2Graph (Serviansky et al., 2020), graph neural networks (Battaglia et al., 2018) or higher-order neural networks (Morris et al., 2019). Recently, Transformer layers have constituted the most popular method for set and graph update (Bresson & Laurent, 2019; Kosiorek et al., 2020; Stelzner et al., 2020; Krawczuk et al., 2021) – we also use such layers in our experiments.

3 A permutation equivariance view on set generation

We have seen that exchangeability is often considered as a key property that set creation methods should satisfy. Yet, this property does not empirically result in superior performance (Mitton et al., 2021). To understand this phenomenon, we analyse the set generation problem using a new definition of equivariance.

As set elements are unordered, the symmetric group $\mathbb{S} = \bigcup_{n \in \mathbb{N}^*} \mathbb{S}_n$ containing all permutations is a symmetry of the problem. This symmetry constitutes a useless factor of variation in the data that needs to be factored out in the latent space. This is achieved when the latent representation z is invariant to the action of $\mathbb{S}$.

In discriminative models, symmetries are accounted for when a neural network f is either equivariant ($f(\pi \boldsymbol{X}) = \pi f(\boldsymbol{X})$) or invariant ($f(\pi \boldsymbol{X}) = f(\boldsymbol{X})$) to the action of the permutation group (Kondor, 2008; Cohen et al., 2019). This definition can unfortunately not be used for generative modeling: since the input is a latent *vector* and not a *set*, it cannot be permuted. Instead, we propose a definition of equivariance for a learning algorithm, which is better suited to generative models:

Definition 1. We say that a learning algorithm is equivariant to the action of a group if the training dynamics do not depend on the group elements that are used to represent the training data.

By writing a stochastic gradient descent step for a model f with parameters θ, it is sufficient to prove that the loss function l satisfies:

$$l(\boldsymbol{X}; f_\theta) = l(\pi \boldsymbol{X}; f_\theta) \quad \forall n > 0, \forall \boldsymbol{X} \in \mathbb{R}^{n \times d},\ \forall \pi \in \mathbb{S}_n$$

This definition generalizes the usual one: if f is a permutation equivariant function and the loss function satisfies $l(\pi f(\boldsymbol{X}), \pi \boldsymbol{Y}) = l(f(\boldsymbol{X}), \boldsymbol{Y})$ for any permutation π and training pair $(\boldsymbol{X}, \boldsymbol{Y})$, then the resulting learning algorithm is equivariant. We illustrate this property with a discriminative task in Appendix A. For generative models, we derive the following conditions:

Lemma 1. *Sufficient conditions for equivariance:*

1. *GANs: if the discriminator is permutation invariant, any algorithm trained with the standard GAN loss is permutation equivariant. No constraint is imposed on the generator.*
2. *Autoencoders, DSPN: if the encoder is permutation invariant and the reconstruction loss l satisfies $\forall \pi \in \mathbb{S}, l(\pi X, \hat{X}) = l(X, \hat{X})$, then the learning algorithm is permutation equivariant. No constraint is imposed on the decoder function.*
3. *Normalizing flows: if the creation yields an exchangeable distribution and the update is permutation equivariant, then the algorithm is permutation equivariant (proved in Köhler et al. (2020)).*

While these results are straightforward to derive (proofs are given in Appendix B), they permit to understand why exchangeability is actually not always needed. In Lemma 1, we first see that this property does not appear in the sufficient conditions for GANs and VAEs. This is due to the mat-to-set function which is implicitly applied to the model output. In other words, *the fact that the output of the model is not a matrix but a set is an assumption, not something that needs to be proved*: matrices are simply a convenient way to represent multi-sets on hardware. Then, on the contrary, normalizing flows compute probability distributions on the space of matrices and not on sets. This is why exchangeability (which is a property defined for matrices) appears a sufficient condition. Note that it would be possible, but maybe less convenient, to define probability distributions directly over sets, in which case exchangeability would not be needed.

To illustrate this phenomenon, we can consider the procedure RANDOMIZE that takes as an input the prediction $\boldsymbol{X}$ of any model f (for example an MLP), draws a permutation π uniformly in $\mathbb{S}_n$, and returns $\pi\boldsymbol{X}$. Then RANDOMIZE $\circ f$ always returns an exchangeable distribution. Yet, RANDOMIZE does not improve generation quality as it simply changes the *representation* a set. This observation removes the main argument in favor of random i.i.d. generation in VAEs and GANs, and explains why the other methods empirically perform as well.

The results in Lemma 1 have also some consequences on design choices for different architectures. First, the equivariance constraint on the loss function for Set VAEs reads $\forall \pi \in \mathbb{S}, l(\pi\boldsymbol{X}, \hat{\boldsymbol{X}}) = l(\boldsymbol{X}, \hat{\boldsymbol{X}})$. The most natural way to satisfy this property is to solve a *matching* problem, i.e., to minimize a given criterion over the permutations of one set. This property gives a formal justification to the two loss functions that are commonly used in this case, namely the Chamfer loss and the Wasserstein-2 distance, respectively defined as

$$d_{\text{Chamfer}}(\boldsymbol{X}, \boldsymbol{X}') = \Big(\sum_{1 \leq i \leq n} \min_{1 \leq j \leq n'} ||\boldsymbol{x}_i - \boldsymbol{x}'_j||_2^2 + \sum_{1 \leq j \leq n'} \min_{1 \leq i \leq n} ||\boldsymbol{x}_i - \boldsymbol{x}'_j||_2^2 \Big) \ / \ 2$$

$$d_{\mathcal{W}_2}(\boldsymbol{X}, \boldsymbol{X}') = \inf_{u \in \{\text{couplings}(\boldsymbol{X}, \boldsymbol{X}')\}} \sum_{1 \leq i \leq n} \sum_{1 \leq j \leq n'} u(\boldsymbol{x}_i, \boldsymbol{x}'_j) \, \|\boldsymbol{x}_i - \boldsymbol{x}'_j\|_2^2$$

Then, for Graph VAEs, the constraint on the adjacency matrix reads $l(\pi\boldsymbol{A}\pi^T, \boldsymbol{A}') = l(\boldsymbol{A}, \boldsymbol{A}')$, and a graph matching problem has further to be solved in order to build a permutation invariant loss. This problem is unfortunately at least as hard as the graph isomorphism problem, which significantly restricts the applicability of VAEs to graph generation.

Similarly, if the symmetry group of the problem is $\mathbb{S} \times \text{SE}(3)$ (where $\text{SE}(3)$ is the group of rigid transformations of $\mathbb{R}^3$), the loss function needs to match sets up to translations, rotations and reflections. This problem, called *point set registration*, can be formalized as the computation of the *Gromov-Hausdorff distance* between sets (Mémoli, 2007). It is equivalent to the quadratic assignment problem, which is NP-hard. Although many approximations have been proposed (Bellekens et al., 2015), the difficulty of registration significantly hampers the design of SE(3) equivariant autoencoders.

4 THE TOP-N CREATION MECHANISM

Building on existing set creation methods and their limitations (inability to extrapolate for MLPs and First-n, slow training for random i.i.d creation), we propose a new method called Top-n creation. This method can be seen as an extension of First-n creation, where the number of points in the

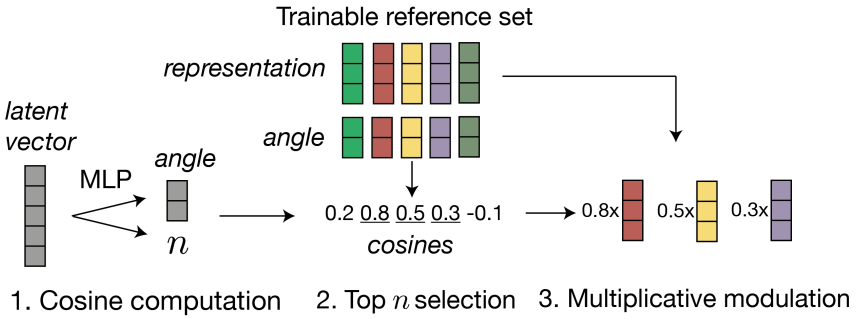

Figure 2: Top-n selects points in a trainable reference set based on the value of the latent vector. To make the process differentiable, a modulation mechanism has to be used – in practice, we use a Film layer (Perez et al., 2018) rather than multiplication. Top-n creation is deterministic and can select the best points from the reference set based on the latent vector.

reference set if decoupled from the maximal set size. The selection of the First-n points is replaced by a differentiable mechanism that learns to select the best n points.

The architecture is summarized in Figure 2. Similarly to First-n selection, we propose to use a reference set, but it can have an arbitrary size. Each point in this set has two components: one (the *angle*) is used to decide when to select the point, and the other contains the *representation* of the point. Given a latent vector $\boldsymbol{z} \in \mathbb{R}^l$, a reference set made of representations $\boldsymbol{R} \in \mathbb{R}^{n_0 \times c}$ and angles $\boldsymbol{\Theta} \in \mathbb{R}^{n_0 \times a}$ as well as learnable matrices $\boldsymbol{W}_i$, Top-n creation consists of the following computations:

$$\boldsymbol{a} = \text{MLP}(\boldsymbol{z}) \quad \in \mathbb{R}^a \quad \text{compute an angle for the set} \tag{1}$$

$$\boldsymbol{c} = \boldsymbol{\Theta}\ \boldsymbol{a}\ /\ \text{vec}((||\boldsymbol{\theta}_i||_2)_{1 \leq i \leq n}) \quad \in \mathbb{R}^{n_0} \quad \text{cosine computation} \tag{2}$$

$$\boldsymbol{s} = \text{sort}(\boldsymbol{c})[:n] \quad \in \mathbb{N}^n \quad \text{selection of the top } n \text{ points} \tag{3}$$

$$\tilde{\boldsymbol{c}} = \text{softmax}(\boldsymbol{c}[\boldsymbol{s}]) \quad \in \mathbb{R}^n \quad \text{modulation coefficients} \tag{4}$$

$$\boldsymbol{X} = \boldsymbol{R}[\boldsymbol{s}] \odot \tilde{\boldsymbol{c}}\ \boldsymbol{W}_1 + \tilde{\boldsymbol{c}}\ \boldsymbol{W}_2 \quad \in \mathbb{R}^{n \times c} \quad \text{modulate } \boldsymbol{R} \text{ with cosine values} \tag{5}$$

$$\boldsymbol{X} = \boldsymbol{X} \odot \boldsymbol{z}\ \boldsymbol{W}_3 + \boldsymbol{z}\ \boldsymbol{W}_4 \quad \in \mathbb{R}^{n \times c} \quad \text{condition on the latent vector} \tag{6}$$

The crux of the Top-n creation module is to select the points that will be used to generate a set based on the value of the latent vector (Eqns (1, 2, 3)). Unfortunately, this process is not differentiable, and a mechanism has to be used in order to train the angle representations. For this purpose, we build a enhanced version of the Top-K pooling mechanism (Gao & Ji, 2019). Top-K pooling uses the value of the cosines $\boldsymbol{c}$ to transform the representation of the selected points, which allows gradients to flow through these cosines. While Gao & Ji (2019) uses a multiplicative modulation ($\boldsymbol{X} = \boldsymbol{X}_{\text{ref}}[\boldsymbol{s}] \odot \tilde{\boldsymbol{c}}$), we replace this step by a FiLM layer (Perez et al., 2018) that combines both additive and multiplicative modulation in Eq. (5).

Finally, while previous works usually append the latent vector to each point, we propose to exploit the equivalence between summation and concatenation when a linear layer is applied. This property can be written

$$\text{cat}(\boldsymbol{X}^{(0)},\ \boldsymbol{1}_n \boldsymbol{z}^T)\ \boldsymbol{W} = \boldsymbol{X}^{(0)} \boldsymbol{W}_1 + \boldsymbol{1}_n (\boldsymbol{z}^T \boldsymbol{W}_2),$$

for $\boldsymbol{W} = \text{cat}(\boldsymbol{W}_1, \boldsymbol{W}_2)$. Contrary to the concatenation (left-hand side), the sum (right-hand side) does not compute several times $\boldsymbol{z}^T \boldsymbol{W}_2$, which reduces the complexity of this layer from $O(n(c+l)c)$ to $O(nc^2 + cl + nl)$. Again, we combine the sum and multiplicative modulation in a FiLM layer to build a more expressive model in Eq. (6).

Overall, Top-n creation retains the advantages of First-n creation (no stochasticity nor collision problem), but removes the arbitrary selection of the first n points. This is especially useful for tasks that have a compositional structure, as the latent vector can help select the most relevant points. Top-n also decouples the number of points in the reference set from the number of points in the training examples: contrary to First-n, it trains all n_0 points even when the training sets do not contain as many points. It results in a better ability to extrapolate to larger sets than those seen during training.

Table 1: Mean Chamfer loss and 95% confidence interval over 6 runs. Methods in italic are those used in the original papers for TSPN (Kosiorek et al., 2020) and DSPN (Zaheer et al., 2017). Top-n consistently improves over the other set creation methods.

Method	Set creation	Chamfer (e-5)	Method	Set creation	Chamfer (e-5)
TSPN	*Random i.i.d.*	16.42 ± 0.53	DSPN	Random i.i.d.	9.52 +- 0.41
	First-n	15.45 ± 1.41		*First-n*	8.87 ± 0.18
	Top-n	$\mathbf{14.98 \pm 0.59}$		Top-n	$\mathbf{7.53 \pm 0.57}$

Since First-n and Top-n always use the same set of reference points, one may wonder if they restrict the model expressivity. We however show that it is not the case: used with a two-layer MLP, the First-n and Top-n modules are universal approximators over sets.

Proposition 1 (Expressivity)**.** *For any set size n, maximal norm M and precision parameter ϵ, there is a 2-layer MLP f and reference points $\{\boldsymbol{x}_1, ..., \boldsymbol{x}_n\}$ such that, for any set $\{\boldsymbol{y}_1, ..., \boldsymbol{y}_n\}$ of points in $\mathbb{R}^d$ with $\forall i,\ ||\boldsymbol{y}_i|| \leq M$ there is a latent vector $\boldsymbol{z}$ of length $n \times d$ that satisfies:*

$$||f(\boldsymbol{X}, \boldsymbol{z}) - \boldsymbol{Y}||_{\mathcal{W}_2} \leq \epsilon$$

Top-n creation can be used in any GAN or autoencoder-based architecture as a replacement for other set creation methods. It is however not suited to normalizing flows, because these require a constant space dimensionality across layers, which is incompatible with our universal approximation result: a MLP with a fixed width may be unable to approximate every set, which could result in a predicted likelihood of 0 for some training sets.

5 Experiments

We compare Top-n to other set and graph creation methods on several tasks: autoencoding point clouds on a set version of MNIST, generating realistic 3D structures on a synthetic molecule-like dataset, and generating varied valid molecules on the QM9 chemical dataset.

5.1 Set MNIST

We first perform experiments on the SetMNIST benchmark, introduced in Zhang et al. (2019a). This datasets consists of point clouds that are built by thresholding the pixel values in MNIST images, adding noise on the locations and normalizing the coordinates. The task consists in autoencoding these point clouds. We aim at showing that Top-n can favorably replace other set creation methods without having to tune the rest of the architecture extensively. For this purpose, we use existing implementations of DSPN (Zhang et al., 2019a) TSPN (Kosiorek et al., 2020)[3], which are respectively a sort of diffusion model and a transformer-based autoencoder. Experiment details can be found in Appendix D.

The results for both methods are very similar (Figure 1): i.i.d. creation has poor performance and needs more epochs to be trained than First-n and Top-n. We attribute the better performance of Top-n to the compositional nature of SetMNIST: if the latent vector contains information about the digit to represent, Top-n generators can select the most relevant points based on the most common locations for each digit, which First-n generators cannot do.

5.2 Synthetic dataset

Since SetMNIST does not allow to measure generation quality easily, we further benchmark the different set generators on a synthetic dataset for which the quality of the generated sets can be assessed. This dataset is a simplified model of molecules in 3D, and retains some of its characteristics: i) atoms are never too close to each other ii) there is a bond between two atoms if and only if they are closer than a given distance (that depends on the atom types) iii) if formal charges are forbidden, each atom has a predefined valency.

[3] github.com/Cyanogenoid/dspn (original code) and github.com/LukeBolly/tf-tspn (reimplementation)

Table 2: Mean and 95% confidence interval over 5 runs on synthetic molecule-like data. The random i.i.d generator extrapolates well, but fails at reconstructing the training data. On the contrary, MLP and First-n generators fit the training data very well but fail at generalizing. Top-n generator fits the training data well and generates better sets, especially in the extrapolation setting.

Method	Train	Generation		Extrapolation		
	Wasserstein distance	Valency loss	Diversity score	Valency loss	Incorrect valency (%)	Diversity score
MLP	$\mathbf{0.80}_{\pm.06}$	$0.43_{\pm.08}$	$4.82_{\pm.15}$	$0.95_{\pm.18}$	$22_{\pm3}$	$4.42_{\pm.19}$
Random i.i.d.	$1.66_{\pm.05}$	$0.61_{\pm.06}$	$3.39_{\pm.33}$	$0.46_{\pm.09}$	$16_{\pm5}$	$3.29_{\pm.36}$
First-n	$0.83_{\pm.02}$	$0.46_{\pm.11}$	$5.01_{\pm.17}$	$0.54_{\pm.08}$	$15_{\pm2}$	$4.68_{\pm0.09}$
Top-n	$0.87_{\pm.07}$	$\mathbf{0.22}_{\pm.07}$	$\mathbf{5.13}_{\pm.18}$	$\mathbf{0.20}_{\pm.03}$	$\mathbf{8}_{\pm2}$	$\mathbf{4.71}_{\pm.13}$

Each set is created via rejection sampling: points are drawn iteratively from a uniform distribution within a bounding box in $\mathbb{R}^3$. The first point is always accepted, and the next ones are accepted only if they satisfy a predefined set of constraints: i) they are not closer to any other point than a given threshold *min-distance*. ii) they are connected to the rest of the set, i.e., have at least one neighbor than a distance *neighbor-distance* iii) they do not have too many neighbors. Our dataset is made of 2000 sets that have between 2 and 35 points, with 9 points per set on average. It is a simplification of real molecules in several aspects: there are only single bonds, angles between bonds are not constrained and the atom types are only defined by the valency, which reflects the fact that atoms with the same valency tend to play a similar role and be more interchangeable.

The task is to reconstruct the atom positions and generate new realistic sets. During training, we measure the Wasserstein ($\mathcal{W}_2$) distance between the input and reconstructed sets. At generation time, we compute the $\mathcal{W}_2$ distance between the distribution of valencies in the dataset and in the generated set, as well as a diversity score to ensure that a method does not always generate the same set. For that purpose we use the average $\mathcal{W}_2$ distance between 1000 pairs of generated sets. We also measure the same metrics in an extrapolation setting where sets have on average 10 more points. Since the valency distribution may not be exactly the same on bigger sets, we also report the proportion of generated atoms whose valency is not present in the dataset (0, or more than 4). In practice, we find that the two measures are very correlated.

We train a VAE with different creation methods on this dataset. Details about the model and the loss function can be found in Appendix D. The results are shown in Table 2. We observe that the random i.i.d. generator generalizes well but reconstructs the training sets very poorly, which reflects the fact that the model is hard to train due to the stochastic i.i.d. generation. The generated sets also lack diversity, which may reveal that the network tends to ignore the features that are drawn and rely on the layers biases. On the contrary, the MLP and First-n generators fit the training data very well but extrapolate badly. Finally, Top-n is able to generate points that have the right valency and performance does not degrades when extrapolating to larger sets.

5.3 Molecular graph generation

Finally, we evaluate Top-n on a graph generation task. We train a graph VAE (detailed in Appendix D) on QM9 molecules (Wu et al., 2018) and check its ability to generate a wide range of valid molecules. As a generation metric, we simply report the validity (i.e., the proportion of molecules that satisfy a set of basic chemical rules) and uniqueness (the proportion of valid non-isomorphic graphs) of the generated molecules. We do not report novelty, as QM9 is an enumeration of all possible molecules up to 9 heavy atoms. Note that most recent methods proposed on this task use a non-learned validity correction code to generate 100% of valid molecules, which obfuscates the real performance of the learned model. We therefore only compare to works that do not correct validity: GraphVAE (Simonovsky & Komodakis, 2018), MolGAN (De Cao & Kipf, 2018) and GraphTransformerVAE (Mitton et al., 2021).

We observe that MolGAN and random i.i.d. generators both suffer from mode collapse: they generate only a few molecules, but these molecules are often valid. Top-n generation very significantly improves upon these methods, and also generates more valid molecules than First-n. However, MLPs still seem to perform best at this task. It is nevertheless striking that the performance of all

Table 3: Performance of different generators trained at generating molecules on QM9. Results for GraphVAE, MolGAN and GraphTransformerVAE are from the original authors – confidence intervals were not reported. Random generation and MolGAN suffer from mode collapse, which explains both the high validity and the very poor uniqueness. The better performance of GraphVAE may be due to the expensive graph matching that they use or the fact that they post-process graphs to ensure connectivity. We attribute the limited performance of all methods to the assumption made at generation time that edges are independent, which is not a realistic model for molecules.

	Method	Generator	Valid (%)	Unique and valid (%)
Baselines	Graph VAE	MLP	$55.7\pm$?	$\mathbf{42.3\pm?}$
	MolGAN	MLP	$98.0\pm$?	$2.3\pm$?
	GraphTransformerVAE	MLP	$74.6\pm?$	$16.785\pm?$
Implemented	Set2GraphVAE	MLP	39.3 ± 1.9	38.3 ± 1.9
		i.i.d.	74.3 ± 21.6	19.6 ± 12.2
		First-n	35.1 ± 3.0	34.6 ± 2.8
		Top-n	38.0 ± 5.4	37.4 ± 5.1

methods is quite limited. To understand why, we need to consider the probabilistic model that is typically used in one-shot methods. At training time, the network learns probability distribution for the atoms, the edges and the edge types. At generation time, Bernouilli (for the edge presence) or multinomial variables (for the atom and edge types) are sampled from these distributions. Implicitly, this model makes an independence assumption between all nodes and edges. Such an hypothesis may be reasonable for edges in a social network, but it is clearly not suited to molecules where atoms have a fixed valency. Designing one-shot generative models without independence assumption is therefore probably be key to effective one-shot molecule generation.

6 Conclusion

In this work, we strengthened the theoretical foundations of one-shot set and graph generation by proposing a definition of equivariant learning algorithms. We showed in particular that, contrary to common belief, exchangeability is not a required property in GANs and VAE. We then proposed Top-n, a non-exchangeable model for one-shot set and graph creation. Our method consistently achieves better performance on set generation. On molecule generation, all methods seem to be limited by the independence assumption implicitly made at generation time, but Top-n is nevertheless competitive. Overall, Top-n is easy to train, it extrapolates well and is able to handle the compositional structure of some datasets by selecting the most relevant points. Our method can be incorporated in any GAN or VAE architecture and replace favorably other set creation methods.

We finally note that our experiments on graph generation and molecule generation in 3D feature only small graphs with 9 nodes on average. This set size is in line with most literature on the topic (Meldgaard et al., 2021; Satorras et al., 2021; Mitton et al., 2021), and we observe on the synthetic dataset that good reconstruction is significantly harder to achieve on large sets. This observation leads to a simple question that has not yet been investigated: can set and graph generation methods avoid the curse of dimensionality? For MLP-based generators, the answer is probably no: as they learn vectors of size $n_{\text{max}}d$ that are then reshaped to matrices, standard universal approximation results will most likely yield a sample complexity that is exponential in n_{max}. Whether and under what circumstances other set generators can overcome this problem is an important question that conditions the success of one-shot models for larger sets and graphs.

Ethics statement Although our method can be used to generate any type of sets or graphs, it was primarily built and benchmarked with molecule generation applications in mind. While learning based methods will probably not replace expert knowledge in the short term, they can already be used to generate *hit* molecules, i.e., small and simple molecules that have promising activity on a predefined target. These molecules can then be refined by computational chemists in order to optimise activity and develop new effective drugs. We are therefore convinced that research on set and graph generation can have a direct positive impact on society.

Reproducibility statement All code is included in the supplementary materials. Further details can be found in Appendix D. More comments, code for the other tasks and pre-trained models will be added upon acceptance of this work.

REFERENCES

Panos Achlioptas, Olga Diamanti, Ioannis Mitliagkas, and Leonidas Guibas. Learning representations and generative models for 3d point clouds. In *International conference on machine learning*, pp. 40–49. PMLR, 2018.

Peter Battaglia, Jessica Blake Chandler Hamrick, Victor Bapst, Alvaro Sanchez, Vinicius Zambaldi, Mateusz Malinowski, Andrea Tacchetti, David Raposo, Adam Santoro, Ryan Faulkner, Caglar Gulcehre, Francis Song, Andy Ballard, Justin Gilmer, George E. Dahl, Ashish Vaswani, Kelsey Allen, Charles Nash, Victoria Jayne Langston, Chris Dyer, Nicolas Heess, Daan Wierstra, Pushmeet Kohli, Matt Botvinick, Oriol Vinyals, Yujia Li, and Razvan Pascanu. Relational inductive biases, deep learning, and graph networks. *arXiv*, 2018. URL `https://arxiv.org/pdf/1806.01261.pdf`.

Ben Bellekens, Vincent Spruyt, Rafael Berkvens, Rudi Penne, and Maarten Weyn. A benchmark survey of rigid 3d point cloud registration algorithms. *Int. J. Adv. Intell. Syst*, 8:118–127, 2015.

Marin Biloš and Stephan Günnemann. Equivariant normalizing flows for point processes and sets, 2021. URL `https://openreview.net/forum?id=LIR3aVGIlln`.

Xavier Bresson and Thomas Laurent. A two-step graph convolutional decoder for molecule generation. *arXiv preprint arXiv:1906.03412*, 2019.

Michael M Bronstein, Joan Bruna, Taco Cohen, and Petar Veličković. Geometric deep learning: Grids, groups, graphs, geodesics, and gauges. *arXiv preprint arXiv:2104.13478*, 2021.

Taco S Cohen, Mario Geiger, and Maurice Weiler. A general theory of equivariant cnns on homogeneous spaces. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett (eds.), *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc., 2019. URL `https://proceedings.neurips.cc/paper/2019/file/b9cfe8b6042cf759dc4c0cccb27a6737-Paper.pdf`.

Gabriele Corso, Luca Cavalleri, Dominique Beaini, Pietro Liò, and Petar Veličković. Principal neighbourhood aggregation for graph nets. In *Advances in Neural Information Processing Systems*, 2020.

George Cybenko. Approximation by superpositions of a sigmoidal function. *Mathematics of control, signals and systems*, 2(4):303–314, 1989.

Nicola De Cao and Thomas Kipf. Molgan: An implicit generative model for small molecular graphs. *ICML Workshop on Theoretical Foundations and Applications of Deep Generative Models*, 2018.

Fabian B. Fuchs, Daniel E. Worrall, Volker Fischer, and Max Welling. Se(3)-transformers: 3d roto-translation equivariant attention networks. In *Advances in Neural Information Processing Systems 34 (NeurIPS)*, 2020.

Hongyang Gao and Shuiwang Ji. Graph u-nets. In *international conference on machine learning*, pp. 2083–2092. PMLR, 2019.

Michael Guarino, Alexander Shah, and Pablo Rivas. Dipol-gan: Generating molecular graphs adversarially with relational differentiable pooling. 2017.

Jinwoo Kim, Jaehoon Yoo, Juho Lee, and Seunghoon Hong. Setvae: Learning hierarchical composition for generative modeling of set-structured data. *arXiv preprint arXiv:2103.15619*, 2021.

Diederik P Kingma and Max Welling. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*, 2013.

Jonas Köhler, Leon Klein, and Frank Noé. Equivariant flows: exact likelihood generative learning for symmetric densities. In *International Conference on Machine Learning*, pp. 5361–5370. PMLR, 2020.

Imre Risi Kondor. *Group theoretical methods in machine learning*. Columbia University, 2008.

Risi Kondor and Shubhendu Trivedi. On the generalization of equivariance and convolution in neural networks to the action of compact groups. In *International Conference on Machine Learning*, pp. 2747–2755. PMLR, 2018.

Adam R Kosiorek, Hyunjik Kim, and Danilo J Rezende. Conditional set generation with transformers. *Workshop on Object-Oriented Learning at ICML 2020*, 2020.

Igor Krawczuk, Pedro Abranches, Andreas Loukas, and Volkan Cevher. Gg-gan: A geometric graph generative adversarial network, 2021. URL `https://openreview.net/forum?id=qiAxL3Xqx1o`.

Juho Lee, Yoonho Lee, Jungtaek Kim, Adam Kosiorek, Seungjin Choi, and Yee Whye Teh. Set transformer: A framework for attention-based permutation-invariant neural networks. In *International Conference on Machine Learning*, pp. 3744–3753. PMLR, 2019.

Yang Li, Haidong Yi, Christopher Bender, Siyuan Shan, and Junier B Oliva. Exchangeable neural ode for set modeling. In H. Larochelle, M. Ranzato, R. Hadsell, M. F. Balcan, and H. Lin (eds.), *Advances in Neural Information Processing Systems*, volume 33, pp. 6936–6946. Curran Associates, Inc., 2020. URL `https://proceedings.neurips.cc/paper/2020/file/4db73860ecb5533b5a6c710341d5bbec-Paper.pdf`.

Meng Liu, Keqiang Yan, Bora Oztekin, and Shuiwang Ji. GraphEBM: Molecular graph generation with energy-based models. In *Energy Based Models Workshop - ICLR 2021*, 2021. URL `https://openreview.net/forum?id=Gc51PtL_zYw`.

Kaushalya Madhawa, Katsuhiko Ishiguro, Kosuke Nakago, and Motoki Abe. Graph{nvp}: an invertible flow-based model for generating molecular graphs, 2020. URL `https://openreview.net/forum?id=ryxQ6T4YwB`.

Søren Ager Meldgaard, Jonas Köhler, Henrik Lund Mortensen, Mads-Peter V Christiansen, Frank Noé, and Bjørk Hammer. Generating stable molecules using imitation and reinforcement learning. *arXiv preprint arXiv:2107.05007*, 2021.

Facundo Mémoli. On the use of gromov-hausdorff distances for shape comparison. 2007.

Joshua Mitton, Hans M Senn, Klaas Wynne, and Roderick Murray-Smith. A graph vae and graph transformer approach to generating molecular graphs. *arXiv preprint arXiv:2104.04345*, 2021.

Christopher Morris, Martin Ritzert, Matthias Fey, William L Hamilton, Jan Eric Lenssen, Gaurav Rattan, and Martin Grohe. Weisfeiler and leman go neural: Higher-order graph neural networks. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pp. 4602–4609, 2019.

Ethan Perez, Florian Strub, Harm De Vries, Vincent Dumoulin, and Aaron Courville. Film: Visual reasoning with a general conditioning layer. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 32, 2018.

Victor Garcia Satorras, Emiel Hoogeboom, Fabian B Fuchs, Ingmar Posner, and Max Welling. E (n) equivariant normalizing flows for molecule generation in 3d. *arXiv preprint arXiv:2105.09016*, 2021.

Hadar Serviansky, Nimrod Segol, Jonathan Shlomi, Kyle Cranmer, Eilam Gross, Haggai Maron, and Yaron Lipman. Set2graph: Learning graphs from sets. *Advances in Neural Information Processing Systems*, 33, 2020.

Martin Simonovsky and Nikos Komodakis. Graphvae: Towards generation of small graphs using variational autoencoders. In *International conference on artificial neural networks*, pp. 412–422. Springer, 2018.

Karl Stelzner, Kristian Kersting, and Adam R Kosiorek. Generative adversarial set transformers. In *Workshop on Object-Oriented Learning at ICML*, volume 2020, 2020.

Nathaniel Thomas, Tess Smidt, Steven Kearnes, Lusann Yang, Li Li, Kai Kohlhoff, and Patrick Riley. Tensor field networks: Rotation-and translation-equivariant neural networks for 3d point clouds. *arXiv preprint arXiv:1802.08219*, 2018.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in neural information processing systems*, pp. 5998–6008, 2017.

Zhenqin Wu, Bharath Ramsundar, Evan N Feinberg, Joseph Gomes, Caleb Geniesse, Aneesh S Pappu, Karl Leswing, and Vijay Pande. Moleculenet: a benchmark for molecular machine learning. *Chemical science*, 9(2):513–530, 2018.

Carl Yang, Peiye Zhuang, Wenhan Shi, Alan Luu, and Pan Li. Conditional structure generation through graph variational generative adversarial nets. In *NeurIPS*, pp. 1338–1349, 2019a.

Guandao Yang, Xun Huang, Zekun Hao, Ming-Yu Liu, Serge Belongie, and Bharath Hariharan. Pointflow: 3d point cloud generation with continuous normalizing flows. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 4541–4550, 2019b.

Manzil Zaheer, Satwik Kottur, Siamak Ravanbakhsh, Barnabas Poczos, Russ R Salakhutdinov, and Alexander J Smola. Deep sets. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett (eds.), *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017. URL https://proceedings.neurips.cc/paper/2017/file/f22e4747da1aa27e363d86d40ff442fe-Paper.pdf.

David W Zhang, Gertjan J. Burghouts, and Cees G. M. Snoek. Set prediction without imposing structure as conditional density estimation. In *International Conference on Learning Representations*, 2021. URL https://openreview.net/forum?id=04ArenGOz3.

Yan Zhang, Jonathon Hare, and Adam Prugel-Bennett. Deep set prediction networks. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett (eds.), *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc., 2019a. URL https://proceedings.neurips.cc/paper/2019/file/6e79ed05baec2754e25b4eac73a332d2-Paper.pdf.

Yan Zhang, Jonathon Hare, and Adam Prügel-Bennett. Fspool: Learning set representations with featurewise sort pooling. *arXiv preprint arXiv:1906.02795*, 2019b.

A THE ROLE OF THE LOSS FUNCTION IN EQUIVARIANCE

While previous work has focused on the definition of equivariance for functions, we provide an illustrative example to show that the loss function plays an important role as well. Consider the problem of predicting the future state of a n-body system. The symmetry group for this task is $\mathbb{G} = \mathbb{S}_n \times SE(3)$: the problem is invariant if all points are rotated, translated or reflected, and they can be permuted as well. We write $g.\boldsymbol{X}$ the action of a group element g on a set $\boldsymbol{X}$. We consider a SE(3) equivariant function for set f. It can for example be a SE(3) transformer (Fuchs et al., 2020) or a Tensor field network (Thomas et al., 2018).

Then, by writing one gradient update step, we easily obtain that if the loss function l satisfies $l(f(g.\boldsymbol{X}^t),\ g.\boldsymbol{X}^{t+1}) = l(f(\boldsymbol{X}^t),\ \boldsymbol{X}^{t+1})$ then the learning algorithm is equivariant. Informally, the loss should not change if both the input and the prediction and the target are transformed by the same group element. For example, $||f(\boldsymbol{X}^t) - \boldsymbol{X}^{t+1}||_2$ is a valid loss, but $||f(\boldsymbol{X}^t) - \boldsymbol{X}^{t+1}||_1$ is not (it is not rotation invariant) and $||\text{diag}((\lambda_i)_{i\leq n})(f(\boldsymbol{X}^t) - \boldsymbol{X}^{t+1})||_2$ neither if the λ_i are not all equal (it is not permutation invariant).

B PROOF OF LEMMA 1

Generative adversarial networks Given a set generator f, a discriminator function d, and $\boldsymbol{X}_1, ..., \boldsymbol{X}_n$ a training dataset, the standard loss function for GANs is formulated as

$$l(f, d, \boldsymbol{X}_1, ..., \boldsymbol{X}_m) = \frac{1}{m}\sum_{i=1}^{m} \log(d(\boldsymbol{X}_i))] + \mathbb{E}_{\mathbb{Z}}[\log(1 - d(f(\boldsymbol{z}))).$$

In order to obtain $l(f, d, \boldsymbol{X}_1, ..., \boldsymbol{X}_m) = l(f, d, \pi_1\boldsymbol{X}_1, ..., \pi_m\boldsymbol{X}_m)$ for every choice of π_i, it is therefore sufficient to choose a permutation invariant discriminator.

Autoencoder based models We assume that the autoencoder is made of a permutation invariant encoder *enc* and an arbitrary decoder *dec*. For any set size n, set $\boldsymbol{X} \in \mathbb{R}^{n\times d}$ and permutation $\pi \in \mathbb{S}_n$ we have

$$\begin{aligned} l(\pi\boldsymbol{X}, \hat{\boldsymbol{X}}) = l(\boldsymbol{X}, \hat{\boldsymbol{X}}) &\implies l(\pi\boldsymbol{X}, \textit{dec}(\textit{enc}(\boldsymbol{X})) = l(\boldsymbol{X}, \hat{\boldsymbol{X}}) \\ &\implies l(\pi\boldsymbol{X}, \textit{dec}(\textit{enc}(\pi\boldsymbol{X})) = l(\boldsymbol{X}, \hat{\boldsymbol{X}}) \quad (\textit{enc} \text{ is invariant}) \\ &\implies l(\pi\boldsymbol{X}, \pi\hat{\boldsymbol{X}}) = l(\boldsymbol{X}, \pi\boldsymbol{X}) \end{aligned}$$

C PROOF OF PROPOSITION 2

Proof. As First-n creation is a special case of Top-n generation (where the selected points are always the same and the cosines for these points are equal to 1), it is sufficient to prove the property for First-n creation.

Given a set $\boldsymbol{Y} = \{\boldsymbol{y}_1, ..., \boldsymbol{y}_n\}$ of points in $\mathbb{R}^{d\times n}$, we propose to sort the points $\boldsymbol{y}_i$ by alphanumeric sort (sort the values using the first feature, then sort points that have the same first features along the second feature, etc.). We denote the resulting matrix by $\boldsymbol{Y}'$. We choose as a latent vector $\boldsymbol{z} = \textit{flatten}(\boldsymbol{Y}') \in \mathbb{R}^{nd}$. It is the vector that contains the representation of $\boldsymbol{y}'_1$ in the first d features, $\boldsymbol{y}'_2$ is the next d features... By construction, $\boldsymbol{z}$ is a permutation invariant representation of the set $\boldsymbol{Y}$ (this reflects the fact that there are canonical ordering for sets, which is not the case for general graphs).

We choose as a reference set the canonical basis in $\mathbb{R}^n$ ($\boldsymbol{R} = \boldsymbol{I}_n$). After the latent vector is appended to the representation of each point, we have $\boldsymbol{r}_i = (\boldsymbol{e}_i, \boldsymbol{z})$. We are now looking for a function f that allows to approximate the set $\boldsymbol{Y}$, i.e., that satisfies $\forall i \leq n, f(\boldsymbol{e}_i, \boldsymbol{z}) = \boldsymbol{z}[i\ d : (i+1)d]$. We can choose $f(\boldsymbol{e}_i, \boldsymbol{z}) = \boldsymbol{e}_i^T\boldsymbol{W}_1\boldsymbol{W}_2\boldsymbol{z}$, where

$$\boldsymbol{W}_1 = \begin{bmatrix} 1 & 1 & 1 & 0 & 0 & 0 & & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 & ... & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & & 1 & 1 & 1 \end{bmatrix} \in \mathbb{R}^{n\times nd} \quad \text{and} \quad \boldsymbol{W}_2 = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \\ & ... & \\ 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \in \mathbb{R}^{nd\times d}$$

.

This function is continuous, and its output is $\boldsymbol{Y}' = \textit{reshape}_{n\times d}(\boldsymbol{z})$. $\boldsymbol{Y}'$ is equal to $\boldsymbol{Y}$ up to a permutation of the rows, so that $||\boldsymbol{Y} - \boldsymbol{Y}'||_{\mathcal{W}_2} = 0$. If the entries of $\boldsymbol{z}$ are bounded, we can use standard approximation results for continuous functions over a compact space (Cybenko, 1989) to conclude that it be uniformly approximated by a 2-layer MLP. □

D DETAILS ABOUT THE EXPERIMENAL SETTING

D.1 SET MNIST

Since we use existing code for this task, we refer to the respective papers (Zhang et al., 2019a) and (Kosiorek et al., 2020) for details on the model used and the loss function. Note that the code used

for TSPN is not the original code (which is not available) but a reimplementation, which explains why the results are different from the ones reported in the paper.

For the Top-n generation, we arbitrarily set the number of points in the reference set to twice the largest generated set (this value was not tuned). We observed that reducing the learning rate improves results for all methods – it is the only hyper-parameter that was tuned.

D.2 SYNTHETIC SET GENERATION

Model The set encoder is made of: a 2-layer MLP, 3 transformer layers followed by a PNA global pooling layer (that computes the sum, mean, max and standard deviation over each channel) and a 2-layer MLP. The decoder is made of a set generator followed by a linear layer, 3 transformer layers and a 2-layer MLP. We use residual connections when possible and batch normalization between each layer.

We experimented with the two ways to sample the number of points presented in Section 2, but we found the results to be quite similar. We therefore opted sampling the number of points from the data distribution, which is the simplest method.

Loss function We use a standard variational autoencoder loss with a Wasserstein reconstruction term and two additional regularizers. Given an input set $\boldsymbol{X}$ and its reconstruction $\hat{\boldsymbol{X}}$, the loss can be written:

$$L(\boldsymbol{X}, \hat{\boldsymbol{X}}) = d_{W_2}(\boldsymbol{X}, \hat{\boldsymbol{X}}) + \lambda_1 \, KL(p(\boldsymbol{z} \mid \boldsymbol{W}), \mathcal{N}(0, \boldsymbol{I}_l)) + \lambda_2 reg_2\,(\hat{\boldsymbol{X}}) + \lambda_3 \, reg_3(\hat{\boldsymbol{X}})$$

where

$$reg_2(\boldsymbol{X}) = \sum_{1 \leq i < j \leq n} (d_0 - ||\boldsymbol{x}_i - \boldsymbol{x}_j||_2)_+ \quad \text{with} \quad d_0 = 1$$

prevents atoms from being generated too close to each other and $reg_3(\boldsymbol{X})$ penalizes atoms that have either no neighbor, or a too large valency (we refer to the code for details on the way to compute it).

Training details In order to train the model efficiently, mini-batches have to be used. This may not be easy when dealing with sets and graphs, since they do not have all the same shape. To circumvent this issue, we reorganise the training data in order to ensure that all sets inside a mini-batch have the same size. At generation time, this method cannot be applied, so we simply generate sets one by one.

The optimizer is Adam with its default parameters. We use a learning rate of $2e^{-4}$ and a scheduler that halves it when reconstruction performance does not improve significantly after 750 epochs. Experimentally, we found the learning rate decay to be important to achieve good reconstruction.

D.3 GRAPH GENERATION

Model Our encoder is a graph neural network is made of 3 message-passing layers followed by a PNA global pooling layer (Corso et al., 2020) and a final MLP. For the decoder, we use a set creation method followed by transformer layers. The resulting representation is then processed independently by i) a MLP which generates node features ii) a Set2Graph layer (Serviansky et al., 2020) followed by two MLPs to generate edge probabilities and edge features.

Graph matching and loss function As explained in Section 3, graph matching in loss function is required to get a permutation equivariant learning algorithm. Instead of using a proper graph matching method, we propose to use the atom types to perform an imperfect but much cheaper alignment between the target and the predicted molecules.

For both molecules, we compute a score for each atom i defined as:

$$s(i) = 10^5 \textit{atom-type}(i) + 10^4 \textit{num-edges}(i) + \sum_{j \in \mathcal{N}(i)} \textit{edge-type}(i,j) * \textit{atom-type(j)}$$

This score cannot differentiate between all atoms in each molecule, but it reduces drastically the number of permutations that can represent the same input. It is motivated by the fact that empirically, we observe that our method quickly learns to reconstruct the molecular formula very well. We then sort the atoms according to there score, and use this permutation to reorder the adjacency matrix and the edge types.

Our model learns to predict a probabilistic model for the atom types, edge presence and edge types. For this purpose, we use standard cross entropy loss between the predicted probabilities and the ground truth. However, these metrics can be hard to optimize because of the imperfect graph matching algorithm. We therefore regularize these metrics with several other measures at the graph level, that do not depend on matching:

- The mean squared error between the real atomic formula and the predicted one.
- The mean squared error between the average number of edges per atom in the input and predicted molecule.
- The mean squared error between the distribution of edge types in the real and predicted molecule.

Finally, we add a matching dependent term, which is the mean squared error between the valencies of the input and the target molecule.

Training details The model is trained over 600 epochs with a batch size of 512 and a learning rate of $2e^{-3}$. It is halved after 100 epochs when the loss does not improve anymore. The optimizer is Adam with default parameters.