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 an output are equally likely) but it is hard to train due to the stochasticity of i.i.d. generation. In this work, 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 realistic 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 vector prior, has been less explored. Yet, it has important applications in drug discovery, for example, where neural networks can be used to sample from the space of stable drug-like molecules.

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, edges or substructures one by one (Liu et al., 2018; Liao et al., 2019; Sun et al., 2020). They are however slow and introduce an order in the points that does not exist in the data. In this paper, 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 permutation

[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.

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 equivariant function.

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 that use this design can generate any number of points, and decouple the number of generated points from the number of trainable parameters in the model. 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 (Krawczuk et al., 2021).

In this work, we first extend the usual definition of equivariance (Kondor & Trivedi, 2018). Given a neural network architecture F_Θ parametrized by Θ and a loss function l, we say that the pair (F, l) is equivariant to the action of a group if the dynamics of Θ trained on gradient descent on l are independent of the group elements used to represent the data. Using this definition, we provide sufficient conditions for equivariance and show that exchangeability is actually not useful in GANs and VAEs, thus explaining why i.i.d. generation is empirically not superior to other approaches.

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* $\boldsymbol{\phi}_i \in \mathbb{R}^a$. 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 build 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 reconstruct the data more accurately 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 the push-forward measure $f_\#(p_Z)$ (i.e., the law of $f(\boldsymbol{z})$) is close to $\mathcal{D}$. 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 $\mathrm{mat\text{-}to\text{-}set}$ 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 set cardinality n has to be sampled. Two strategies have been proposed:

- Most of the literature assumes that n is known during training. The distribution of the number of points in the training data is computed, and at generation time, a variable is sampled from this multinomial distribution. This method assumes that the latent vector $\boldsymbol{z}$ is independent of 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 ground truth cardinality 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.

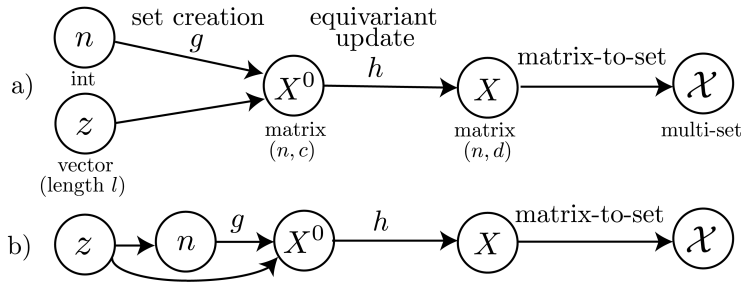

Figure 1: The two graphical models for one-shot set creation. The number of points can either be a) sampled from the dataset distribution or b) learned from the latent vector. In this work, we study ways to parametrize g. For the update h, any equivariant function can be used. Graph generation methods 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 h 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. MLPs ignore the symmetries of the problem and are only trained to generate up to n_{max} points, with no ability to extrapolate to larger sets. Despite these limitations, they 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., all permutations of the rows of $\boldsymbol{X}^0$ are equally likely. 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).

Unfortunately, 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 generate realistic sets or reconstruct the training data (Krawczuk et al., 2021), which we also observe experimentally (Appendix D).

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 $\boldsymbol{X}_{\text{ref}} \in \mathbb{R}^{n_{\text{max}} \times c}$, 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. First-n converges much faster than with i.i.d. generation (Krawczuk et al., 2021), but 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 $\boldsymbol{X}_{\text{ref}}$ introduces a bias because 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 (Maron et al., 2018). 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 AN EQUIVARIANCE PERSPECTIVE ON SET AND GRAPH GENERATION

In this section, we analyse one-shot set and graph generation from the perspective of permutation equivariance. As set and graphs are unordered, the symmetric group $\mathbb{S} = \bigcup_{n\in\mathbb{N}^*} \mathbb{S}_n$ containing all permutations is a symmetry of these tasks. A permutation $\pi \in \mathbb{S}_n$ acts on a $n \times n$ matrix $\boldsymbol{A}$ by permuting its rows and columns (which we write $\pi.\boldsymbol{A} = \pi\, \boldsymbol{A}\, \pi^T$), on a $n \times c$ matrix by permuting its rows ($\pi.\boldsymbol{X} = \pi\boldsymbol{X}$), and leaves a vector $\boldsymbol{z} \in \mathbb{R}^h$ unchanged ($\pi.\boldsymbol{z} = \boldsymbol{z}$). This symmetry constitutes a useless factor of variation in the data that should be factored out in the latent space. It is achieved when the latent representation $\boldsymbol{z}$ is invariant to the action of $\mathbb{S}$ ($\pi.\boldsymbol{z} = \boldsymbol{z}$).

In discriminative models, symmetries are accounted for when a neural network f is equivariant to the action of the permutation group (Kondor, 2008), which writes $\pi.f(\boldsymbol{X}) = f(\pi.\boldsymbol{X})$. This definition can unfortunately not be used for set and graph generation from a latent vector. Indeed, imposing $\pi.f(\boldsymbol{z}) = f(\pi.\boldsymbol{z}) = f(\boldsymbol{z})$ would only allow for trivial solutions where all rows are equal, which is too restrictive. Another definition, that provides more relaxed conditions, is therefore needed for taking symmetries into account. For this purpose, we propose a definition called (F, l)-equivariance.

Our proposition is based on the observation that equivariant models for discriminative tasks are almost always used along with invariant loss functions. For example, the l_2 loss $||\boldsymbol{Y} - \hat{\boldsymbol{Y}}||_2$ is commonly used to learn the future state of a n-body system, but not the l_1 loss $||\boldsymbol{Y} - \hat{\boldsymbol{Y}}||_1$, as it is not rotation invariant. Formally, if $F_\Theta = \{f_\theta : \mathbb{X} \to \mathbb{Y};\ \theta \in \Theta\}$ is an hypothesis class of $\mathbb{G}$-equivariant functions from $\mathbb{X}$ to $\mathbb{Y}$ (for example a neural architecture parametrized by θ), then the loss functions l should satisfy

$$\forall f \in F,\ \forall g \in \mathbb{G},\ \forall(\boldsymbol{X}, \boldsymbol{Y}) \in \mathbb{X} \times \mathbb{Y}, \quad l(g.f(\boldsymbol{X}), g.\boldsymbol{Y}) = l(f(\boldsymbol{X}), \boldsymbol{Y}) \tag{1}$$

When l is chosen accordingly, the gradients with respect to the parameters satisfy $\nabla_\theta\ l(f(g.\boldsymbol{X}), g.\boldsymbol{Y}) = \nabla_\theta\ l(f(\boldsymbol{X}), \boldsymbol{Y})$, i.e., each parameter update is independent of the group elements that are used to represent $\boldsymbol{X}$ and $\boldsymbol{Y}$. It follows that the training dynamics as a whole become independent of the group elements used to represent the data, which makes data augmentation unnecessary. We therefore propose the following definition:

Definition 1 ((F, l)-equivariance)**.** Consider an hypothesis class $F_\Theta \subset \mathbb{Y}^\mathbb{X}$, a group $\mathbb{G}$ that acts on $\mathbb{X}$ and $\mathbb{Y}$ and a loss function l defined on $\mathcal{Y}$. We say that the pair (F_Θ, l) is equivariant to the action of $\mathbb{G}$ if the dynamics of $\theta \in \Theta$ trained with gradient descent on l do not depend on the group elements that are used to represent the training data.

With our definition, the use of an equivariant architecture and an invariant loss is a sufficient condition for (F, l)-equivariance in discriminative settings. As for standard generative architectures for sets and graphs, we can derive the following sufficient conditions (proofs are given in Appendix A):

Lemma 1. *Sufficient conditions for (F, l)-equivariance:*

1. *GANs: if F is a GAN architecture with a permutation invariant discriminator, and l the standard GAN loss, then (F, l) is permutation equivariant. No constraint is imposed on the generator.*
2. *Autoencoders, VAEs, DSPN: if F is an encoder-decoder architecture with a permutation invariant encoder, and the reconstruction loss l satisfies $\forall \pi \in \mathbb{S}, l(\pi.\boldsymbol{X}, \hat{\boldsymbol{X}}) = l(\boldsymbol{X}, \hat{\boldsymbol{X}})$, then (F, l) is permutation equivariant. No constraint is imposed on the decoder function.*
3. *Normalizing flows: if F is an architecture such that the set creation yields an exchangeable distribution, the update is permutation equivariant and invertible, and p_θ denotes the model likelihood, then $(F, -\log p_\theta)$ is permutation equivariant (proved in Köhler et al. (2020)).*

As desired, (F, l)-equivariance is a notion of equivariance that does not impose the equivariance of the probabilistic decoder in generative architectures. Furthermore, for GANs and VAEs, we observe that the set generator does not need to yield exchangeable distributions either. To understand why exchangeability is not needed, recall that in GANs and VAEs a $\mathrm{mat\text{-}to\text{-}set}$ function is *implicitly* applied to the model output: a method that generates matrices that are always permuted in the same way is therefore equivalent to one that generates exchangeable matrices (as the corresponding sets are the same). 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[3]. This observation explains why random i.i.d. generation empirically does not outperform non-exchangeable set creation methods such as MLPs and First-n (Krawczuk et al., 2021).

We also note that the constraints of Lemma 1 are satisfied by most existing architectures, including early ones (Simonovsky & Komodakis, 2018; De Cao & Kipf, 2018; Köhler et al., 2020). For example, the constraint on the loss for VAEs ($\forall \pi \in \mathbb{S}, l(\pi.\boldsymbol{X}, \hat{\boldsymbol{X}}) = l(\boldsymbol{X}, \hat{\boldsymbol{X}})$) is satisfied by the two loss functions commonly used for sets, namely Chamfer loss and the Wasserstein-2 distance, defined as

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

$$d_{\mathcal{W}_2}(\boldsymbol{X}, \boldsymbol{X}') = \inf_{u \in \{\Gamma(\boldsymbol{X}, \boldsymbol{X}')\}} \sum_{1 \leq i \leq n} \sum_{1 \leq j \leq n'} u(\boldsymbol{x}_i, \boldsymbol{x}'_j) \, \|\boldsymbol{x}_i - \boldsymbol{x}'_j\|_2^2, \tag{3}$$

where Γ is the set of couplings (i.e., bistochastic matrices) between $\boldsymbol{X}$ and $\boldsymbol{X}'$. While efficient algorithms exist for computing the Wasserstein distance (Cuturi, 2013; Feydy et al., 2019), the equation $l(\pi.\boldsymbol{X}, \hat{\boldsymbol{X}}) = l(\boldsymbol{X}, \hat{\boldsymbol{X}})$ may be difficult to satisfy in other settings: matching graphs up to permutations, or sets up to the $SE(3)$ group, leads to difficult problems for which no polynomial time algorithm is known (Mémoli, 2007). In these settings, the design of VAE is harder and other architectures may be better suited.

Overall, the notion of (F, l)-equivariance provides justifications to the design of standard architectures for sets and graphs both for generative and discriminative tasks. Contrary to common belief, it also appears that exchangeability not needed to build equivariant GANs and VAEs for sets and graphs. In the following section, we therefore design a new creation mechanism without worrying about the model exchangeability.

4 THE TOP-N CREATION MECHANISM

We have seen in Section 2 that existing set creations methods suffer from important limitations: i.i.d. creation is slow to train due to the random initial points, while MLPs and First-n use a fixed mask to select the correct number of points and cannot extrapolate to larger sets. In order to solve these limitations, we propose a new method called Top-n creation, which is summarized in Figure 2.

Similarly to First-n, Top-n also uses a reference set, but in Top-n this set can have an arbitrary size n_0. Each point in this set is a pair $(\boldsymbol{\phi}, \boldsymbol{r})$: the *angle* $\boldsymbol{\phi} \in \mathbb{R}^a$ is used to decide when to select the point, and $\boldsymbol{r} \in \mathbb{R}^c$ contains the *representation* of the point. Given a latent vector $\boldsymbol{z} \in \mathbb{R}^{l \times 1}$, a

[3]On the contrary, normalizing flows typically compute probability distributions on the space of matrices and not on sets. It is therefore natural that exchangeability (which is a property defined for matrices) appears for normalizing flows and not for the other architectures.

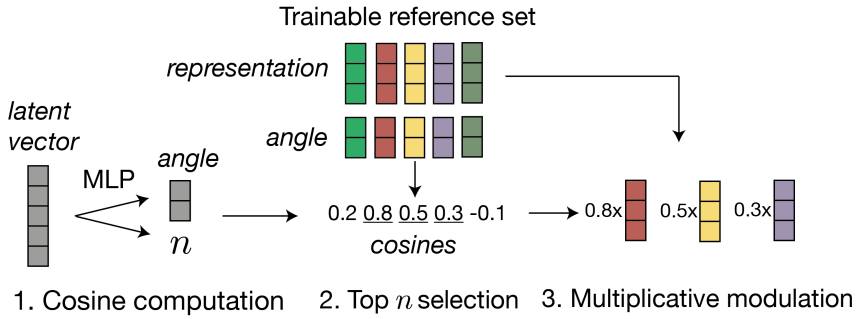

Figure 2: Top-n creation learns to select the most relevant points in a trainable reference set based on the value of the latent vector. To obtain gradients and train the angles and the MLP despite the non-differentiable argsort operation, we modulate the selected representations with the values of the cosines – in practice, we use a FiLM layer (Perez et al., 2018) rather than multiplication.

reference set made of angles $\boldsymbol{\Phi} \in \mathbb{R}^{n_0 \times a}$ and representations $\boldsymbol{R} \in \mathbb{R}^{n_0 \times c}$ and learnable matrices $\boldsymbol{W}_1$ to $\boldsymbol{W}_4$ (respectively of size $1 \times c,\ 1 \times c,\ l \times c,\ l \times c$), Top-n creation computes:

$$\boldsymbol{a} = \text{MLP}_1(\boldsymbol{z}) \quad \in \mathbb{R}^a \quad \text{angle of the latent vector} \tag{4}$$

$$\boldsymbol{c} = \boldsymbol{\Phi}\ \boldsymbol{a}\ /\ \text{vec}((||\boldsymbol{\phi}_i||_2)_{1 \leq i \leq n_0}) \quad \in \mathbb{R}^{n_0} \quad \text{cosine computation} \tag{5}$$

$$\boldsymbol{s} = \text{argsort}_{\downarrow}(\boldsymbol{c})[:n] \quad \in \mathbb{N}^n \quad \text{selection of the top } n \text{ points} \tag{6}$$

$$\tilde{\boldsymbol{c}} = \text{softmax}(\boldsymbol{c}[\boldsymbol{s}]) \quad \in \mathbb{R}^{n \times 1} \quad \text{modulation coefficients} \tag{7}$$

$$\boldsymbol{X}^0 = \boldsymbol{R}[\boldsymbol{s}] \odot \tilde{\boldsymbol{c}}\ \boldsymbol{W}_1 + \tilde{\boldsymbol{c}}\ \boldsymbol{W}_2 \quad \in \mathbb{R}^{n \times c} \quad \text{modulation of } \boldsymbol{R} \text{ with cosine values} \tag{8}$$

$$\boldsymbol{X}^0 = \boldsymbol{X}^0 \odot \boldsymbol{1}_n\ \boldsymbol{z}^T\ \boldsymbol{W}_3 + \boldsymbol{1}_n\ \boldsymbol{z}^T\ \boldsymbol{W}_4 \quad \in \mathbb{R}^{n \times c} \quad \text{condition on the latent vector} \tag{9}$$

The crux of the Top-n creation module consists in selecting the points that will be used to generate a set based on the value of the latent vector (Eqns (4, 5, 6)). Unfortunately, the gradient of the argsort operation is 0 almost everywhere ($\partial \boldsymbol{R}[\boldsymbol{s}]/\partial \boldsymbol{\Phi} = 0$), and a mechanism has to be used in order to train the angles $\boldsymbol{\Phi}$ and the MLP of Eq. (4). In Top-n, we modulate the representation of the selected points $\boldsymbol{R}[\boldsymbol{s}]$ with the cosines $\boldsymbol{c}$ (Eq. 7). This operation provides a path in the computational graph that does not go through the argsort, so that the gradients of $\boldsymbol{\Phi}$ and $\boldsymbol{a}$ are not always 0. For example:

$$\frac{d\boldsymbol{X}^0}{d\boldsymbol{\Phi}} = \frac{\partial \boldsymbol{X}^0}{\partial \boldsymbol{R}[\boldsymbol{s}]} \frac{d\boldsymbol{R}[\boldsymbol{s}]}{d\boldsymbol{\Phi}} + \frac{\partial \boldsymbol{X}^0}{\partial \tilde{\boldsymbol{c}}} \frac{d\tilde{\boldsymbol{c}}}{d\boldsymbol{\Phi}} = \frac{\partial \boldsymbol{X}^0}{\partial \tilde{\boldsymbol{c}}} \frac{d\tilde{\boldsymbol{c}}}{d\boldsymbol{\Phi}},$$

Equations (4) to (8) build upon the Top-K pooling mechanism used by Gao & Ji (2019) for graph coarsening, but differ in several aspects. First, Gao & Ji (2019) compute cosines between the angle $\boldsymbol{a}$ and the representations $\boldsymbol{R}$. This method may tend to select points that are similar. On the contrary, we parametrize the angles and the representations independently, so that two points can have similar angles (in which case they will usually be selected together) but very diverse representations at the same time. Second, Gao & Ji (2019) uses a multiplicative modulation ($\boldsymbol{X}^0 = \boldsymbol{X}_{\text{ref}}[\boldsymbol{s}] \odot \tilde{\boldsymbol{c}}$) in Eq. (8) while we use a more expressive FiLM layer (Perez et al., 2018) that combines both additive and multiplicative modulation.

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. (9).

Our algorithm retains the advantages of First-n creation, but replaces the arbitrary selection of the first n points by a mechanism that can select the most relevant points for a given set. Since it

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	$\mathbf{15.45} \pm 1.41$		*First-n*	8.87 ± 0.18
	Top-n	$\mathbf{14.98} \pm 0.59$		Top-n	$\mathbf{7.53} \pm 0.57$

also decouples the number of points in the reference set from the number of points in the training examples, the size of the reference set becomes a hyperparameter of the model. Empirically, we observe a tradeoff: when using more points in the reference set, each point is updated less often which makes training slower; however, the model tends to avoid overfitting and generalize better.

Since First-n and Top-n use a fixed 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 row-wise 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 size $nd \times 1$ that satisfies:*

$$||f(cat(\boldsymbol{X}, \mathbf{1}_n \boldsymbol{z}^T)) - \boldsymbol{Y}||_{\mathcal{W}_2} \leq \epsilon$$

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 it is based on a hard selection process (the value of the reference points that are not selected is not used) and is therefore non invertible.

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[4]. All training curves are available in Appendix D.

5.1 Set MNIST

We first perform experiments on the SetMNIST benchmark, introduced in Zhang et al. (2019a). The task consists in autoencoding point clouds that are built by thresholding the pixel values in MNIST images, adding noise on the locations and normalizing the coordinates. Our goal is to show 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)[5], which are respectively a sort of diffusion model and a transformer-based autoencoder. Experiment details can be found in Appendix C.

The results for both methods are very similar (Figure 1): the standard exchangeable i.i.d. creation has poor performance and needs more epochs to be trained than First-n and Top-n. Since all sets in this dataset have the same cardinality (360 points), the better performance of Top-n cannot come from its better ability to process varying sizes, but rather shows that it is able to select the most relevant reference points for each set.

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 as-

[4]Source code is available at [linked masked for anonymity]

[5]github.com/LukeBolly/tf-tspn (reimplementation by someone else) and github.com/Cyanogenoid/dspn

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 to reconstruct the training data. On the contrary, MLP and First-n generators overfit the training data very and have poor sampling performance. 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	**0.80**$\pm$.06	0.43$\pm$.08	**4.82**$\pm$.15	0.95$\pm$.18	22$\pm$3	4.42$\pm$.19
Random i.i.d.	1.66$\pm$.05	0.61$\pm$.06	3.39$\pm$.33	0.46$\pm$.09	16$\pm$5	3.29$\pm$.36
First-n	**0.83**$\pm$.02	0.46$\pm$.11	**5.01**$\pm$.17	0.54$\pm$.08	15$\pm$2	**4.68**$\pm$.09
Top-n	0.87$\pm$.07	**0.22**$\pm$.07	**5.13**$\pm$.18	**0.20**$\pm$.03	**8**$\pm$2	**4.71**$\pm$.13

sessed. 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.

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 goal 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 C, as well as an ablation study on the number of points in the reference set. 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 seem to overfit the training data at the expense of generation quality. 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 C) on QM9 molecules 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 non-isomorphic graphs among valid ones) of the generated molecules. We do not report novelty, as QM9 is an enumeration of all possible molecules up to 9 heavy atoms that satisfy a predefined set of constraints (Ruddigkeit et al., 2012; Ramakrishnan et al., 2014): in this setting, generating novel molecules is therefore not an indicator of good performance, but rather a sign that the distribution of the training data has not been properly captured. Note that most recently proposed methods proposed on this task use a non-learned validity

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. Our architecture provides an effective approach to one-shot molecule generation. Apart from random i.i.d. creation, the different set creation methods seem to be equivalent in this setting.

	Method	Generator	Valid (%)	Unique and valid (%)
Baselines	Graph VAE	MLP	$55.7\pm$?	$42.3\pm$?
	Graph VAE + RL	MLP	$94.5\pm$?	$32.4\pm$?
	MolGAN	MLP	$98.0\pm$?	$2.3\pm$?
	GraphTransformerVAE	MLP	$74.6\pm$?	$16.8\pm$?
Implemented	Set2GraphVAE	MLP	60.5 ± 2.2	$\mathbf{55.4} \pm 2.3$
		Random i.i.d.	34.9 ± 15.2	29.9 ± 10.0
		First-n	59.9 ± 2.7	$\mathbf{56.2} \pm 2.7$
		Top-n	59.9 ± 1.4	$\mathbf{56.2} \pm 1.1$

correction code to generate almost 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), an extension of GraphVAE that incorporates reinforcement learning (Kwon et al., 2019), MolGAN (De Cao & Kipf, 2018) and GraphTransformerVAE (Mitton et al., 2021).

We observe that both MolGAN and the Graph VAE that incorporates reinforcement learning suffer from mode collapse: they can generate only a few molecules, but these molecules are often valid. On the contrary, our method is able to generate a higher rate of valid and unique molecules. We observe that the random i.i.d. method is not able to obtain a good train loss (Appendix D), which reflects in the poor generation performance as well. The three other set creation methods seem to perform similarly. Our interpretation is that almost all molecules in QM9 have the same size (9 heavy atoms, since hydrogens are not represented), which makes it less important to properly handle varying graph sizes as is proposed in Top-n.

We finally note that recursive methods such as Li et al. (2018) can often generate higher rates of valid molecules. The reason is that recursive methods can easily check at each step that the added edge will not break valency constraints. They have therefore an unfair advantage over one-shot models which do not incorporate these checks.

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, and is able to generate a wide range of valid molecular graphs. Overall, Top-n is easy to train (as it is deterministic), it extrapolates well and is able to select the most relevant points for each set. 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 C.

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.

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.

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.

Marco Cuturi. Sinkhorn distances: Lightspeed computation of optimal transport. *Advances in neural information processing systems*, 26:2292–2300, 2013.

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.

Jean Feydy, Thibault Séjourné, François-Xavier Vialard, Shun-ichi Amari, Alain Trouve, and Gabriel Peyré. Interpolating between optimal transport and mmd using sinkhorn divergences. In *The 22nd International Conference on Artificial Intelligence and Statistics*, pp. 2681–2690, 2019.

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.

Youngchun Kwon, Jiho Yoo, Youn-Suk Choi, Won-Joon Son, Dongseon Lee, and Seokho Kang. Efficient learning of non-autoregressive graph variational autoencoders for molecular graph generation. *Journal of Cheminformatics*, 11(1):1–10, 2019.

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.

Yibo Li, Liangren Zhang, and Zhenming Liu. Multi-objective de novo drug design with conditional graph generative model. *Journal of cheminformatics*, 10(1):1–24, 2018.

Renjie Liao, Yujia Li, Yang Song, Shenlong Wang, Charlie Nash, William L Hamilton, David Duvenaud, Raquel Urtasun, and Richard S Zemel. Efficient graph generation with graph recurrent attention networks. *arXiv preprint arXiv:1910.00760*, 2019.

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.

Qi Liu, Miltiadis Allamanis, Marc Brockschmidt, and Alexander L Gaunt. Constrained graph variational autoencoders for molecule design. *arXiv preprint arXiv:1805.09076*, 2018.

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.

Haggai Maron, Heli Ben-Hamu, Nadav Shamir, and Yaron Lipman. Invariant and equivariant graph networks. *arXiv preprint arXiv:1812.09902*, 2018.

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.

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.

Raghunathan Ramakrishnan, Pavlo O Dral, Matthias Rupp, and O Anatole von Lilienfeld. Quantum chemistry structures and properties of 134 kilo molecules. *Scientific Data*, 1, 2014.

Lars Ruddigkeit, Ruud Van Deursen, Lorenz C Blum, and Jean-Louis Reymond. Enumeration of 166 billion organic small molecules in the chemical universe database gdb-17. *Journal of chemical information and modeling*, 52(11):2864–2875, 2012.

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.

Yongbin Sun, Yue Wang, Ziwei Liu, Joshua Siegel, and Sanjay Sarma. Pointgrow: Autoregressively learned point cloud generation with self-attention. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, pp. 61–70, 2020.

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.

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.