Score-based 3D molecule generation with neural fields

Matthieu Kirchmeyer*, **Pedro O. Pinheiro***, **Saeed Saremi**
Prescient Design, Genentech

Abstract

We introduce a new functional representation for 3D molecules based on their continuous atomic density fields. Using this representation, we propose a new model based on neural empirical Bayes [1] for unconditional 3D molecule generation in the continuous space using neural fields. Our model, FuncMol, encodes molecular fields into latent codes using a conditional neural field, samples noisy codes from a Gaussian-smoothed distribution with Langevin MCMC, denoises these samples in a single step and finally decodes them into molecular fields. FuncMol performs all-atom generation of 3D molecules without assumptions on the molecular structure and scales well with the size of molecules, unlike most existing approaches. Our method achieves competitive results on drug-like molecules and easily scales to macro-cyclic peptides, with at least one order of magnitude faster sampling. The code is available at https://github.com/prescient-design/funcmol.

1 Introduction

Generative modeling of 3D molecular structures, if deployed successfully, has the potential to help on many problems in material and life sciences. Recently, state-of-the-art generative models from image and text were adapted to 3D molecule generation, achieving some degree of success [2, 3]. However, unlike other domains where the data modality is defined by the representation itself (e.g., a digital image *is* a tensor of pixels), there are multiple ways to represent a molecule. Therefore, an important question when modeling 3D molecules is: *what constitutes a good representation for molecules?*

Recent methods for 3D molecule generation usually represent molecules as point clouds of atoms [4] or discrete grids of atomic densities [5], which we will refer to as voxel grids. Point clouds are processed by graph neural networks (GNNs), usually based on equivariant architectures [6, 7]. GNNs are known to be less expressive than other architectures due to the message passing formalism [8, 9, 10] and often scale quadratically with the number of atoms. On the other hand, voxel grids are compatible with more expressive models (e.g., convnets and transformers) but computation and memory scales cubically with the volume occupied by the molecules. These limitations in expressivity and scalability hinder the scope of application of these models.

In this work, we propose a new representation for molecules that overcomes those limitations. Inspired by the 3D computer vision community [11], we represent *molecules as fields encoding atomic occupancy*, i.e., continuous functions that map 3D coordinates to atomic densities. While vision data is obtained via discrete measurements, molecular fields are continuous by nature. We handle these fields as such, by parameterizing the molecular occupancy field with a neural network[1], shared among all molecules, and modulation codes, specific to each molecule. The former models common molecular structures (e.g., bonds, angles, valencies, symmetries) while the later encodes variations that make each molecule unique. Given a modulation code, we decode the molecular field

* Equal contribution. Correspondence to kirchmeyer.matthieu@gene.com, oliveira_pinheiro.pedro@gene.com, saremi.saeed@gene.com

[1] Fields that are parametrized by neural networks are referred to as neural fields, implicit neural representations (INR) or coordinate-based neural networks.

38th Conference on Neural Information Processing Systems (NeurIPS 2024).

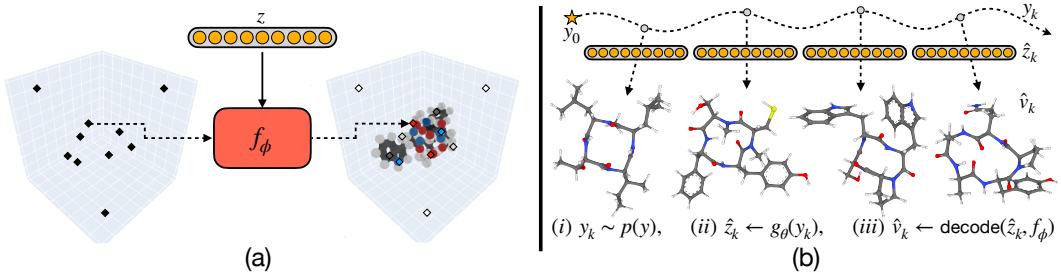

Figure 1: (a) a conditional neural field encodes a molecular field v into a low dimensional latent code z. (b) using a learned score function g_θ, FuncMol performs sampling in latent space via Langevin MCMC. These codes are decoded back into molecules.

by predicting the occupancy of each atom at given 3D coordinates (see Figure 1(a)). This decodes the molecules into explicit representations (such as discrete grids at arbitrary resolution or a `.sdf` format file), useful for downstream tasks.

We perform generative modeling in the continuous function space simply by sampling new modulation codes. Our proposed approach, *FuncMol*, leverages a modulation code denoiser to sample molecules following the (score-based) neural empirical Bayes (NEB) approach [1]. NEB enjoys many properties such as fast-mixing, simplicity for training and fast sampling speed. Sampling is composed of three steps: (i) *(walk)* sample noisy modulation codes with a Langevin Markov chain Monte Carlo (MCMC), (ii) *(jump)* estimate the "clean" modulation codes, and (iii) *(decode)* convert the estimated codes into a molecule. Figure 1(b) illustrates a "walk-jump sampling" chain, with samples generated by our model trained on a macrocyclic peptides dataset [12].

The neural molecular field representation has many advantages over prior representations: (i) it represents complex high-dimensional data in a relatively low-dimensional compact space, (ii) it is scalable (w.r.t. the number of points, size of molecules and resolution) and has low memory footprint, (iii) it does not make any assumptions on molecular structure or geometry, (iv) it can represent molecular structures at arbitrary resolutions and for a free-form discretization, (v) it is compatible with expressive machine learning architectures, and (vi) it is domain-agnostic and can be used for a variety of molecular design problems that can be expressed over fields, e.g., atomic densities, surfaces, pharmacophores, molecular orbitals, electron densities etc.

In summary, our contributions are as follows. We introduce a new way to represent molecular structures with neural fields. These representations are low-dimensional, compact, scalable and do not make any assumptions on the molecular structure. We then propose FuncMol, a score-based model for 3D molecule generation that leverages these representations. We show that FuncMol performs competitively against representative baselines on the drug-like molecules dataset GEOM-drugs [13], based on a wide set of standard and new metrics that we introduce to better measure the generation quality. These results were achieved with one order magnitude faster sampling time. Finally, we illustrate FuncMol's ability to scale to larger 3D molecules by training it on CREMP [12], a recent macro-cyclic peptide dataset, to which our baselines are currently unable to scale.

2 Related work

Neural fields. Neural fields, also referred to as implicit neural representations (INRs), are coordinate-based neural networks that map coordinates (e.g., pixels on an image or coordinates in 3D Euclidean space) to features (e.g., RGB values or atomic occupancies). The idea of representing data points implicitly as neural networks dates back to the work of [14]. Recently, these representations have been successfully applied to model continuous signals, e.g., 2D images [15, 16, 17], 3D shapes [18, 19, 20, 21], 3D scenes [22, 23], videos [24, 25], physics [26, 27], due to their appealing properties. Recently, two concurrent seminal work lead to a fast progress of neural fields by overcoming the spectral bias of coordinate-based neural networks [28]. Sitzmann *et al.* [29] propose SIREN, a neural network that uses periodic activation functions, while Tancik *et al.* [30] considers a positional encoding based on Fourier features. Built on top of those architectures, multiplicative filter networks (MFNs) [31] represent fields as a simple linear combination over an exponential number of

basis functions (e.g. Fourier or Gabor basis). Due to their simplicity and strong performance, we use MFNs to model the atomic occupancy fields.

Generative models of fields. Generative models for neural fields were first applied in 3D computer vision problems. Mescheder *et al.* [19] learn the distribution of shape occupancy fields with VAEs [32], while [18, 33] achieves similar objectives using GANs [34]. Diffusion models [35] have also been applied to learn the distribution of neural fields [36, 37, 38, 39]. Some work [37, 40] parameterize the neural field with the vector of all the corresponding weights. However, when the signal is complex and the neural fields have large number of parameters (e.g., in the order of millions), it is preferable to parameterize the field with a latent code with much lower dimension [36, 41, 42, 43]. Dupont *et al.* [36] fit the whole dataset with a shared coordinate-based network and learn a latent modulation code for each field with gradient-based meta learning [44]. Similarly to them, we parameterize neural fields with latent modulation codes. However, instead of applying meta learning, we learn the latent codes through stochastic optimization, either following the "auto-encoding" [19] or the "auto-decoding" [20] framework.

3D molecule generation. Most 3D molecule generation approaches represent atoms as points (with coordinates and atom types) and molecules as a set of points. For example, [45, 46, 47] propose autoregressive approaches to sample atoms, while [48, 49] use normalizing flows [50]. Hoogeboom *et al.* [4] propose EDM, a diffusion model [35] applied to point cloud of atoms with E(3) equivariance [6]. Many follow-up works extend EDM [51, 52, 53]. For example, [54, 55, 56] improve its performance by leveraging extra information during training (such as molecular graph and formal charges). This contribution is orthogonal to ours and can potentially be incorporated into our generative model. Other approaches [57, 58] map atomic densities on discrete 3D regular grids and leverage computer vision techniques for generation. Recently, VoxMol [5] (and its latent version [59]), a score-based generative model based on neural empirical Bayes [1], shows that voxel-based representations can achieve state-of-the-art results on 3D drug-like molecule generation. However, these methods scale cubically with the volume occupied by molecules, which limits its scope of application. Neural fields are the continuous generalization of discrete 3D grid representations: they achieve good performance on 3D molecule generation and are more efficient in terms of memory and computation.

Conditional molecule generation. Voxels and point-clouds have also been used for conditional 3D molecule generation, usually by building upon an unconditional model. The authors in [57] condition generation on 3D pharmacophores features, [60, 61, 62, 63] generate ligands conditioned on protein pockets, [64] generate molecules conditioned on fragments and [65, 66] generate 3D conformations conditioned on molecular graphs. We are aware of only one other work that uses field-based representation for molecules [67]. There are several differences between our works: they use different data representation, neural network architecture and noise model. While they consider the problem of generating molecule conformations given a molecular graph, we handle the more general problem of unconditional 3D molecule generation (without access to a molecular graph). Our model can easily be adapted to conformer generation by conditioning the generative model to the molecular graph. Moreover, our approach can also be conditioned to tasks where we do not have access to molecular graphs, such as structure-based drug design or electron density generation.

3 Neural atomic occupancy fields

We now describe how we represent molecules as continuous occupancy fields, how we approximate them with neural fields and how we decode the neural fields to retrieve molecular conformations. We finish the section by providing some useful properties of our neural field representations.

3.1 Molecules as continuous occupancy fields

We represent atoms as continuous Gaussian-like shapes in 3D space, centered around their atomic coordinates. Molecules are defined as fields mapping every point in the 3D space to the atomic densities of each atom type, $v : \mathbb{R}^3 \to \mathbb{R}^n$, where n is the number of atom types in the dataset $\mathcal{D}$. We follow previous work [68, 69, 70], and compute the occupancy field v_a for each atom type a by integrating the occupancy generated by all atoms of this type, i.e.:

$$\forall x \in \mathbb{R}^3, v_a(x) = 1 - \prod_{i=1}^{n_a} \Big(1 - \exp\Big(-\Big(\frac{\|x - x_{a_i}\|}{.93r}\Big)^2\Big)\Big), \tag{1}$$

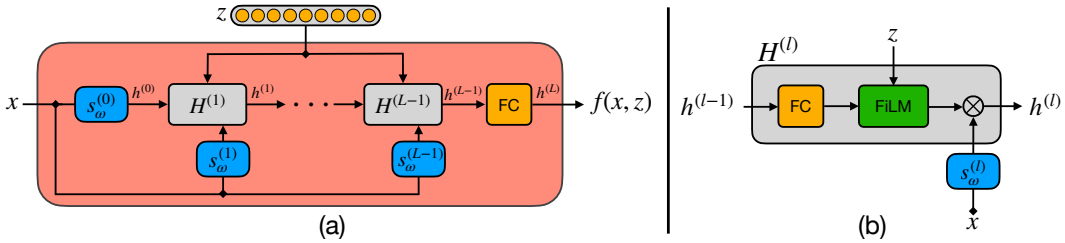

Figure 2: Conditional neural field f_ϕ using the multiplicative filter network architecture. (a) A latent code z and some coordinates x are given as input to the model that outputs the occupancy field at that location for the corresponding molecule, $f_\phi(x, z)$. (b) The code and coordinates are processed via FiLM layers and Hadamard products. We denote the overall operation at layer l as $H^{(l)}$.

where a_i is the i^{th} atom of type a, for a total of n_a atoms. We set the atoms' radius to be $r = .5$Å for all atom types. Molecular fields are smooth functions taking values between 0 (far away from all atoms) and 1 (at the center of atoms).

3.2 Molecular neural fields

Each molecule in the dataset is mapped to a modulation code $z \in \mathbb{R}^d$ and we parameterize the molecular occupancy v with a conditional neural field $f_\phi : \mathbb{R}^3 \times \mathbb{R}^d \to \mathbb{R}^n$. Our objective is to learn the parameters ϕ and the modulation code z such that for any molecular field v and coordinate $x \in \mathbb{R}^3$, $f_\phi(x, z) = v(x)$. We approximate the molecular fields with a linear combination of an exponential large number of parameterized basis functions K, such that amplitudes are modulated by the individual codes z:

$$f_\phi(x, z) = \sum_{k=1}^{K} A_k(z) s_k(x) + \text{bias},$$

where A_k and s_k are the amplitudes and basis functions, respectively. We achieve this parametrization by modeling the neural field with multiplicative filter networks (MFN) [31], a type of coordinate-based network that provides an elegant way to perform this linear combination under some assumptions on the basis functions. We introduce the parameters associated with these functions in Equation (4).

Our conditional MFN is a network with L multiplicative blocks, as illustrated on Figure 2(a). We implement conditioning of the MFN's parameters with FiLM layers [71]. Each multiplicative block is composed of a fully-connected layer, a FiLM modulation layer and a elementwise product with a basis, as illustrated on Figure 2(b). The neural field can be expressed by the following recursive expression:

$$\begin{aligned}
h^{(0)}(x) &= s_{\omega^{(0)}}(x), \\
h^{(l)}(x) &= \left(\gamma^{(l-1)} \odot \left(W_f^{(l-1)} h^{(l-1)}(x)\right) + \left(b^{(l-1)} + \beta^{(l-1)}\right)\right) \odot s_{\omega^{(l)}}(x), \quad l \in (1, L-1), \\
f_\phi(x, z) &\triangleq h^{(L)}(x) = W_f^{(L-1)} h^{(L-1)}(x) + b^{(L-1)},
\end{aligned}$$

where $s_{\omega^{(l)}}$ is a spatial basis function parameterized by $\omega^{(l)}$, $\odot$ denotes the Hadamard product and

$$\beta^{(l-1)} = W_\beta^{(l-1)} z, \text{ and } \gamma^{(l-1)} = W_\gamma^{(l-1)} z,$$

are the bias and scale modulation terms. Note that FiLM layers are not conditioned on the spatial basis, which guarantees that only amplitudes A_k in Equation (2) depend on the code. We propose two different approaches to learn the parameters of the neural field $\phi = \{W_f^{(l)}, b^{(l)}, \omega^{(l)}, W_\beta^{(l)}, W_\gamma^{(l)}\}$ and the modulation codes z (one per each molecule in $\mathcal{D}$).

Auto-decoding. In this setting, introduced by [20], we initialize each code randomly and directly learn them (together with the parameters of the neural field) with backpropagation. This is achieved by solving the following optimization problem:

$$\underset{\phi, \{z_v\}_{v \in \mathcal{D}}}{\arg\min} \sum_{v \in \mathcal{D}} \int \|f_\phi(x, z_v) - v(x)\|_2^2 \, \mathrm{d}x, \tag{2}$$

where the integral is approximated by sampling finite sets of points $\mathcal{X} \subset \mathbb{R}^3$. While auto-decoding was usually applied in settings with relatively few samples, we were able to scale the training to large datasets of one million samples (see Appendix B). See Algorithm 1 on Appendix B for more details.

Auto-encoding. This approach, introduced by [19] and illustrated in Appendix B Figure 4, generates the modulation code via an encoder ζ_ψ, parameterized by ψ and decodes the neural field back. ζ_ψ is a (trainable) 3D convolutional network encoder that takes (low-resolution) voxel grids $\mathcal{G}$ as inputs. This approach is flexible and compatible with other encoder architectures and molecule representations (e.g., GNN/point clouds). The parameters of the encoder and the neural field are learned with the following objective:

$$\arg\min_{\phi,\psi} \sum_{v\in\mathcal{D}} \int \|f_\phi(x, \zeta_\psi(\mathcal{G}) - v(x)\|_2^2 \, \mathrm{d}x. \tag{3}$$

Once training is done, we generate the code with the trained encoder. See Algorithm 2 on Appendix B for more details. Instead of learning the codes individually, this approach learns an encoder, which allows to leverage data augmentation more efficiently. As a result it helps learn a more structured latent space. These benefits are reflected empirically in our experiments.

3.3 From codes to atomic coordinates

By leveraging the modulation codes z and the neural field f_ϕ, we have access to the (learned) continuous occupancy field, $f_\phi(\cdot, z)$. However, in many useful applications in chemistry and biology, we are more interested in the 3D conformation of molecules. Next, we describe how we can extract the molecular conformation from a learned (or generated, as we will see next) modulation code.

We start by identifying all atoms in the field, their approximate locations and their type. To this end, we render a discretized voxel grid from the molecular field using an uniform discretization of space and the neural field $f_\phi(\cdot, z)$. We then apply a peak finding algorithm to infer the number of atoms in the molecule on each channel of the grid (each representing a separate atom type) and their (discretized) coordinates. Finally, we introduce a new continuous refinement to find the local maximum of the neural field. For each identified atoms a, we refine its coordinates around the neighborhood of the coordinates found with the peak detector x_a^0:

$$x_a = \underset{x\in\mathbb{R}^3:\|x-x_a^0\|\leq r}{\arg\max} [f_\phi(x, z)]_a,$$

where $[f_\phi(x, z)]_a$ denotes the field restricted to the channel corresponding to the atom type. This continuous refinement finds atomic coordinates that lie beyond the initial coarse uniform discretization. In practice, we batch the refinement process across molecules and use L-BFGS. We demonstrate in Appendix E.2 its efficiency compared to prior non-continuous refinement approaches from [72, 5].

3.4 Molecular neural fields properties

The proposed conditional neural field enjoys many properties that make it a natural choice for handling large 3D molecules represented as continuous fields.

Flexibility w.r.t. basis. Conditioning MFNs gives the flexibility to choose any type of spatial basis that satisfies a multiplicative-sum property (see the definition in [31]). In our preliminary experiments, we observed that setting the spatial basis to Gabor filters performed better than Fourier filters as they account for the sparse nature of occupancy fields. For each layer l, we consider the following Gabor parameterization, also used in [27]:

$$s_{\omega^{(l)}}(x) = \exp\Big(-\frac{\nu^{(l)}}{2}\|x - \mu^{(l)}\|_2^2\Big)(\cos(\Omega^{(l)}x), \sin(\Omega^{(l)}x)), \tag{4}$$

where $\mu^{(l)}$ is the mean of the Gabor filter, $\nu^{(l)}$ is the scale, $\Omega^{(l)}$ is the frequency and $(\cdot,\cdot)$ refers to the concatenation operator. Equation (4) combines both real and imaginary parts of the complex Gabor filter. This allows to remove phase parameters and reduce the overall parameter count of MFNs [31]. Other choices of basis are also possible and are left for future work.

Parameter efficiency. Our overall conditional MFN formulation is parameter efficient and shares parameters across molecules and channels (i.e. atom types). As [27], we excluded the basis functions parameters ω from FiLM to further decrease the parameter count.

Memory efficiency. Our conditional neural field can be trained on any free-form discretization of the input field. Occupancy values are computed on the fly. This allowed to train FuncMol with large batch size even on large 3D molecules. We found that training the neural field by up-sampling points

close to the atoms' center improved training time as further detailed in Appendix B. Alternative approaches like VoxMol [5] cannot be trained efficiently in this setting: for reference, on the macrocyclic peptide generation task of Section 5.4, on 4 A100 GPUs VoxMol's training cost per epoch was 10 hours while our neural field's training cost was less than 12 minutes.

Reconstruction quality and robustness to noise. Finally our neural field reconstructs accurately the input data as demonstrated in Appendix E.1. Moreover, operating on these latent codes makes our model extremely robust to noise in code space. We demonstrate this property in Appendix E.3 by reporting the sampling metrics when perturbing the codes z by a Gaussian noise.

Sampling efficiency. We use the latent codes for generative modeling as explained in Section 4. Most sampling operations are done on a small dimensional latent space, while decoding into a full molecular field is done only after sampling. As we show in Section 5, our approach (which involves sampling latent code followed decoding them into molecules) achieves at least one order magnitude faster molecule sampling time than previous methods.

4 Score-based generative modeling

We use our latent modulation representations for a downstream generative modeling task. Section 4.1 describes the neural empirical Bayes (NEB) formalism used in our method and Section 4.2 explains how we perform sampling.

4.1 Neural empirical Bayes

Let $p(z)$ be the distribution of codes and $p(v)$ be the (unknown) distribution of molecular fields, defined more formally as the pushforward of $p(z)$ via the mapping $z \mapsto f_\phi(\cdot, z)$. NEB estimates the score function of a smoothed density of the codes $p(y)$, $g_\theta(y) \approx \nabla \log p(y)$. Indeed sampling from a smoothed density $p(y)$ benefits from faster mixing than on the original density $p(z)$ [1, 73, 74]. This smoothed distribution is defined by transforming the random variable Z with an additive isotropic Gaussian noise with a known noise level σ, $Y = Z + N$, where $N \sim \mathcal{N}(0, \sigma^2 I_d)$. The noise level σ plays a key role, trading-off simplicity of the denoising objective and the sampling quality.

NEB is based on an empirical Bayes view of (denoising) score-based models that relates the estimator of clean data (denoiser) and the score function of the smoothed density at a fixed noise level [75, 76, 1]. The denoiser is taken to be the least-square estimator of Z given $Y = y$ which is the Bayes estimator, i.e. $\hat{z}(y) = \mathbb{E}[Z|Y = y]$. Under Gaussian noise, denoiser and smoothed score function are related by

$$\hat{z}(y) = y + \sigma^2 \nabla \log p(y). \tag{5}$$

The denoiser is parameterized by a neural network and learned by minimizing the following objective:

$$\mathcal{L}(\theta) = \mathbb{E}_{z\sim p(z), \varepsilon \sim \mathcal{N}(0,\sigma^2 I_d)} \left\| z - \hat{z}_\theta(z + \varepsilon) \right\|_2^2. \tag{6}$$

The score function is recovered from a learned denoiser via Equation (5) and is used for sampling smoothed codes (see Section 4.2). In practice, we optimize the empirical loss based on the latent codes inferred from a set of molecular fields $\mathcal{D}$. See pseudo-code in Appendix B, Algorithm 3.

4.2 Walk-jump sampling

We use the score function g_θ to sample codes using the *walk-jump sampling* (WJS) scheme [1, 77, 73, 78]. This approach samples molecules from $p(z)$ using the learned score function of noisy data instead of clean data. It consists of two main steps: walking and jumping as detailed in Appendix B, Algorithm 4. Figure 1(b) illustrates these two main steps in a WJS chain: walking consists in generating noisy codes while jumping consists in generating full 3D molecules.

(initialization) To improve mixing, as [77], we initialize the chains by adding uniform noise to Gaussian noise (with the same σ used when training the denoiser). In practice we define the uniform noise over the range of code values, i.e., $y_0 = \mu + \varepsilon$, $\varepsilon \sim \mathcal{N}(0, \sigma^2 I_d)$, where $\mu \sim \mathcal{U}_d(\min_{z\in\mathcal{D}_z, i\in\{1\cdots d\}} z_i, \max_{z\in\mathcal{D}_z, i\in\{1\cdots d\}} z_i)$ and $\mathcal{D}_z$ is the training dataset of codes.

(walk step) Noisy codes are sampled from $p(y)$ with Langevin MCMC algorithms that discretize the underdamped Langevin diffusion [79] starting from y_0 and $u_0 = 0$:

$$\mathrm{d}u_t = -\gamma u_t \mathrm{d}t \; + \; g_\theta(y_t)\mathrm{d}t \; + \; \sqrt{2\gamma}\, \mathrm{d}B_t, \qquad \mathrm{d}y_t = u_t \mathrm{d}t \; , \tag{7}$$

where B_t is the standard Brownian motion in $\mathbb{R}^d$ and γ is the friction (the "mass" is set to 1). We discretize this SDE using the ABOBA scheme from Sachs *et al.* [80], given a discretization step δ and a fixed number of walk steps K. We analyze the impact of K in Appendix E.4.

(jump step) At a given time step K, clean samples are estimated by denoising the smooth code, i.e., $z_K = \hat{z}_\theta(y_K)$. The codes are used to obtain the atomic coordinates as detailed in Section 3.3.

5 Experiments

We now evaluate our model for unconditional generation. We start with a description of our experimental setup (Section 5.1), then present our results on two popular small molecule datasets (Sections 5.2 and 5.3) and a recent macro-cyclic peptide dataset (Section 5.4).

5.1 Experimental setup

Datasets. We evaluate FuncMol on three datasets: *QM9* [81], *GEOM-drugs* [82] and *CREMP* [12]. QM9 contains an enumeration of all possible molecules up to 9 heavy atoms (29 including hydrogens) satisfying some constraints [83]. GEOM-drugs contains multiple conformations for 430K drug-sized molecules (computed with semi-empirical density functional theory), with an average of 44 heavy atoms per molecule. CREMP is a recent dataset that contains multiple conformations of macrocyclic peptides 4-6 residue long, with an average of 74 heavy atoms per molecule. We model hydrogen explicitly and consider 5 chemical elements for QM9 (C, H, O, N, F), 6 for CREMP (C, H, O, N, F, S) and 8 for GEOM-drugs (C, H, O, N, F, S, Cl and Br), ignoring the P, I and B elements that occur extremely rarely. We use a split of 100K/20K/13K molecules for QM9, 1.1M/146K/146K on GEOM-drugs and 409K/10K/9K on CREMP for train, validation and test, respectively. We use the same pre-processing and splits in [54] for QM9 and GEOM-drugs and in [84] for CREMP.

Implementation details. Our main model, *FuncMol*, follows the auto-encoding approach described in Section 3.2. The codes z are computed with an encoder that takes as input a low-resolution voxelized representation of the molecular field with grid dimension of $16\times16\times16$. The encoder is a 3D CNN containing 4 residual blocks, where each block contains 3 convolutional layers followed by BatchNorm, ReLU and pooling (max pooling on the first three blocks and average pooling on the last one) layers. We consider modulation codes with dimension 1024 on QM9 and 2048 on GEOM-drugs and CREMP. We use the same neural field network for all datasets: a conditional MFN with Gabor filters and 6 FiLM-modulated layers, where each fully-connected layer has 2048 hidden units. We augment the training set by applying random rotations on the three Euler angles. The weights of the latent code encoder and neural field decoder are trained jointly.

We also show results for the auto-decoding based model, *FuncMol*$_{\text{dec}}$. In this setting, we initialize the codes randomly and optimize them together with the neural field weights. This approach is less fit for performing large amounts of augmentation as it solves a costly per-sample optimization problem; thus we did not apply data augmentation. As a consequence, we observed that this model is more prone to memorization than the auto-encoding approach (e.g., on GEOM-drugs, around 33% of the generated molecules are copies from the training set).

We normalize the codes to have zero mean and unit variance. We choose a noise level in normalized space of $\sigma = 1.2$ for GEOM-drugs and CREMP, $\sigma = 2.0$ for QM9. Our code denoiser is a modified version of the denoiser used in [36]: a fully-connected network with 18 residual blocks (each with two linear layers with 6144 hidden units) and skip connections. We remove the bias of all layers and use ReLU activations as in [85]. To limit memorization in FuncMol$_{\text{dec}}$, we add dropout (ratio 0.3) between the fully-connected layers in each residual block. For QM9, we consider a smaller network (6 residual blocks and 4096 hidden units). We initialize the MCMC chains with noise and use the following sampling hyperparameters $\gamma = 1.0$ and $\delta = \sigma/2$ as in [5, 78]. For evaluation purposes, we generate one sample per chain. We consider 1000 steps for QM9 and GEOM-drugs and 10000 steps for CREMP. See Appendix B for more details on the implementation.

Baselines. We compare FuncMol and FuncMol$_{\text{dec}}$ to three state-of-the-art approaches. *EDM* [4] and *GeoLDM* [53] are diffusion models operating on point clouds (the latter is a latent-space extension of the former). *VoxMol* [5] is a voxel-based generative model that uses neural empirical Bayes, similar to our generative approach. All of the methods generate molecules as a set of atom types and their coordinates. EDM and GeoLDM apply diffusion directly to point clouds, while VoxMol and FuncMol

Table 1: QM9 results w.r.t. test set for 10000 samples per model. ↑/↓ indicate that higher/lower numbers are better. The row *data* are randomly sampled molecules from the validation set. We report 1-sigma error bars over 3 sampling runs.

	stable mol %↑	stable atom%↑	valid %↑	unique %↑	valency W_1↓	atom TV↓	bond TV↓	bond len W_1↓	bond ang W_1↓	time s/mol.↓
data	98.7	99.8	98.9	99.9	.001	.003	.000	.000	.120	-
EDM	97.9	99.8	99.0	98.5	.011	.021	.002	.001	.440	0.54
GeoLDM	97.5	99.9	100.	98.0	.005	.017	.003	.007	.435	0.65
VoxMol	89.3	99.2	98.7	92.1	.023	.029	.009	.003	1.96	0.83
$\text{FuncMol}_{\text{dec}}$	88.6	99.2	100.	81.1	.022	.066	.032	.006	1.21	0.05
FuncMol	89.2 (±.4)	99.0 (±.07)	100. (±0)	92.8 (±0.3)	.021 (±0.001)	.012 (±0.001)	.006 (±0.003)	.005 (±0.009)	1.56 (±0.06)	0.05

rely on an additional (cheap) post-processing step to extract atomic coordinates from voxel grids or modulation codes, respectively. We follow previous work [58, 54, 5, 62, 86], and use standard cheminformatics software (OpenBabel [87]) to determine the molecule's atomic bonds given their atomic coordinates. The same post-processing is applied to all models for fairness of comparison.

Metrics We consider several metrics used in previous work [5] to benchmark unconditional molecule generation for the standard QM9 and GEOM-drugs datasets (for the CREMP metrics, see Section 5.4): *stable mol* and *stable atom*, the percentage of stable molecules and atoms (as defined in [4]); *validity*, the percentage of generated molecules that passes RDKit [88]'s sanitization filter; *uniqueness*, the proportion of valid molecules that have different canonical SMILES; *valency* W_1, the Wasserstein distance between the distribution of valencies in the generated and test set; *atoms TV* and *bonds TV*, the total variation between the distribution of atom types and bond types; *bond length* W_1 and *bond angle* W_1, the Wasserstein distance between the distribution of bond and lengths. We also report the *average sampling time per molecule*. In the case of our method, this time includes the MCMC "walk" steps, the denoising "jump", the rendering, peak detection and bond inference.

To further investigate the quality of molecular conformations and other molecular properties on GEOM-drugs, we consider some additional metrics. These include: *single fragment*, the percentage of molecules that contains only a single fragment; *median strain energy* [89], the difference between the internal energy of the generated molecule's pose and a relaxed pose of the molecule using RDKit's Universal Force Field [90], computed over all molecules; *ring size TV*, the total variation between the empirical distribution of ring sizes (i.e. number of heavy atoms in rings) in generated and test sets; *number of atoms/mol TV*, the total variation between the empirical distribution of number of atoms per molecule in generated and test sets (in the case of molecules with multiple fragments, we consider only the largest fragment); *QED, SA and logp*, measure the drug-likeness score [91], the synthesizability score [92] and the lipophilic efficiency, respectively (computed with RDKit).

Ablations. In Appendix E we report a series of ablation studies for the neural field and the generative model. Appendix E.1 measures the reconstruction quality of the training molecules. Appendix E.2 illustrates the improvements due to continuous atomic coordinate refinement. Appendix E.3 shows that our field-based decoder is robust to noise, making it an ideal choice for generative modeling. Appendix E.4 ablates the impact of the number of walk steps in the WJS scheme of Section 4.2. Finally, Appendix E.5 ablates the impact of the chosen resolution when sampling codes and decoding them back to molecules. In practice, we observe that 0.25Å provides a good trade-off between the sampling time and the quality of the generated molecules.

5.2 Results on QM9

As pointed by previous authors [93, 4], this dataset is not fully suited for unconditional generative models: a model that captures the training distribution will have to generate samples from training set, due to the enumeration. However, many previous work report results on this dataset. Therefore, we also show results for completeness.

Table 2: GEOM-drugs results, standard metrics w.r.t. test set for 10000 samples per model. ↑/↓ indicate that higher/lower numbers are better. The row *data* are randomly sampled molecules from the validation set. We report 1-sigma error bars over 3 sampling runs.

	stable mol %↑	stable atom%↑	valid %↑	unique %↑	valency $W_{1\downarrow}$	atom TV↓	bond TV↓	bond len $W_{1\downarrow}$	bond ang $W_{1\downarrow}$	time s/mol.↓
data	99.9	99.9	99.8	100.0	.001	.001	.025	.000	0.05	-
EDM	40.3	97.8	87.8	99.9	.285	.212	.048	.002	6.42	9.35
GeoLDM	57.9	98.7	100.	100.	.197	.099	.024	.009	2.96	8.96
VoxMol	75.0	98.1	93.4	99.6	.254	.033	.024	.002	0.64	7.55
$\text{FuncMol}_{\text{dec}}$	69.7 (±.6)	95.3 (±.1)	100. (±.0)	77.5 (±.6)	.268 (±.001)	.035 (±.001)	.028 (±.001)	.003 (±.000)	2.13 (±.01)	0.29
FuncMol	69.7 (±.2)	98.8 (±.0)	100. (±.0)	95.3 (±.1)	.245 (±.001)	.109 (±.001)	.052 (±.000)	.003 (±.000)	2.49 (±.06)	0.29

Table 1 report the metrics described in Section 5.1. We see that FuncMol slightly improves VoxMol and both models perform worse compared to the equivariant point-cloud based baselines. We note that sampling time of FuncMol is an order of magnitude better than baselines.

5.3 Results on GEOM-drugs

Table 2 reports the same set of metrics as in the previous dataset. FuncMol performs favorably over point cloud diffusion models and is close to VoxMol's performance. In particular, FuncMol and VoxMol generate molecules that are significantly more stable and better capture the distribution of bond angles. Table 3 shows results on additional metrics (described in Section 5.1). We also include the following plots of Appendix F: Figure 9 shows the cumulative distribution function of strain energies for generated molecules and Figures 10 and 11 show the histograms of the other metrics.

The results are clear: *FuncMol samples better drug-like molecules than point-cloud diffusion models*. In fact, about half the molecules of point cloud methods have multiple fragments, they have an order of magnitude higher median strain energy, the distribution of ring sizes is off and the QED, SA and logp scores are lower. The results of FuncMol are close to VoxMol in most but not all metrics. However, our approach is much more scalable and efficient: *FuncMol generates molecules an order of magnitude faster than previous methods* (see the last column of Table 2). Appendix H shows some molecules generated by FuncMol on GEOM-drugs.

Table 3: GEOM-drugs results, additional metrics w.r.t. test set for 10000 samples per model. ↑/↓ indicate that higher/lower numbers are better. The row *data* are randomly sampled molecules from the validation set. We report 1-sigma error bars over 3 sampling runs.

	single frag %↑	median energy↓	ring sz TV↓	atms/mol TV↓	QED ↑	SA ↑	logp ↑
data	100.	54.5	.011	.000	.658	.832	2.95
EDM	42.2	951.3	.976	.604	.472	.514	1.11
GeoLDM	51.6	461.5	.644	.469	.497	.593	1.05
VoxMol	82.6	69.2	.264	.636	.659	.762	2.73
$\text{FuncMol}_{\text{dec}}$	80.2 (±.6)	96.4 (±1.1)	.324 (±.008	.970 (±.008)	.677 (±.015)	.788 (±.038)	2.87 (±.00)
FuncMol	70.5 (±.2)	109.7 (±1.1)	.427 (±.006)	1.05 (±.00)	.713 (±.001)	.811 (±.005)	3.09 (±.02)

5.4 Results on CREMP

To showcase the scalability of FuncMol, we train it on a dataset of larger molecules. We choose the macrocylic peptides of CREMP, that contains on average 74 atoms, making it challenging to train models using point-clouds. These molecules also pose serious limitations to voxel-based approaches as they require modeling a volume of 24^3 cubic Angstroms. We tried to train VoxMol on this dataset

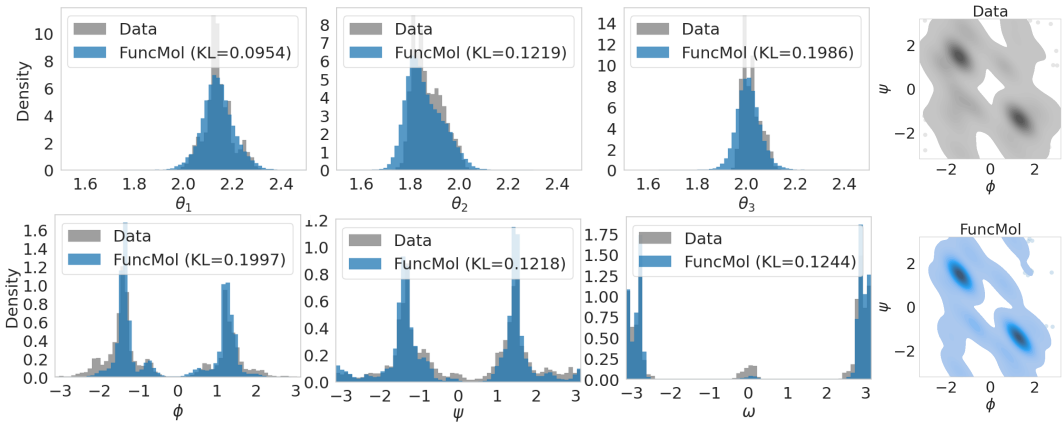

Figure 3: Qualitative evaluation on CREMP following [84]. Left: Comparison of the bond angles (θ_1, θ_2, θ_3) in each amino acid residue and dihedral distributions (ϕ, ψ, ω) for each residue from the reference test set (gray) and the generated samples (blue). KL divergence is calculated as KL(test $||$ sampled). Right: Ramachandran plots [94] (colored by density where darker tones represent high density regions).

using the official implementation, but did not succeed in training it: it takes around 10 hours per epoch on 4 A100 GPUs, while FuncMol takes, on the same hardware, less than 12 minutes per neural field epoch and 15s per denoiser epoch. We use the same code dimension and neural field architecture as in GEOM-drugs—the computational training cost of FuncMol remains unchanged, despite the increased complexity of the molecules.

Figure 3 shows that FuncMol captures well the underlying distribution of macrocyclic conformations. We show the distribution of bond angles (θ_1, θ_2, θ_3) and dihedrals (ϕ, ψ, ω) of both molecules from test set and generated molecules. We also show the KL-divergence between test and generated samples. Approximately 65% of the generated molecules were valid peptides (that is, we could extract a sequence of amino acids from the SMILES strings). The Ramachandran plots [94] show that FuncMol recovers the main modes of the distribution. We note that the bond angles and dihedrals distributions are learned without having any explicit priors on the structure of these peptides. Appendix H shows some generated macrocylic peptides.

Finally, our model takes around 1.5s to generate a molecule. For reference, should VoxMol be trained successfully, it would take over a minute to sample a single molecule (assuming similar sampling parameters as in other datasets). This is a substantial speedup that showcases the potential of FuncMol to scale to even larger molecules.

6 Discussion

We introduce a new continuous representation of 3D molecules based on their atomic occupancy field and a score-based generative model operating on this representation. Each molecule is assigned a (learned) code that modulates a shared conditional neural field network. We demonstrate that we can build an all-atom generative model of 3D molecules, FuncMol, with state-of-the-art sampling time and competitive performance on challenging drug-like datasets. We believe that this model introduces a new paradigm for all-atom 3D modeling of molecules that has many useful properties, namely scalability, expressivity, and flexibility, as it can model various molecular design problems (involving structure, electron densities, etc.) with minor architecture changes.

Future research directions include exploring different neural field architectures, adapt the model for conditional generation (e.g., structure conditioning) or model the molecular bonds alongside the atomic coordinates[2]. Moreover, the scalability of our approach can be a potential alternative for all-atoms representations of large biomolecules.

[2]Recent work [54, 95] show that this improves generation quality. See Appendix G to see how our method compares with a representative baseline using bond information.

Acknowledgements We would like to thank the Prescient Design team for helpful discussions and Genentech's HPC team for providing a reliable environment to train and analyze models.

References

[1] Saeed Saremi and Aapo Hyvärinen. Neural empirical bayes. *JMLR*, 2019. (cit. on pp. 1, 2, 3, and 6)

[2] Camille Bilodeau, Wengong Jin, Tommi Jaakkola, Regina Barzilay, and Klavs F Jensen. Generative models for molecular discovery: Recent advances and challenges. *Computational Molecular Science*, 2022. (cit. on p. 1)

[3] Benoit Baillif, Jason Cole, Patrick McCabe, and Andreas Bender. Deep generative models for 3d molecular structure. *Current Opinion in Structural Biology*, 2023. (cit. on p. 1)

[4] Emiel Hoogeboom, Vıctor Garcia Satorras, Clément Vignac, and Max Welling. Equivariant diffusion for molecule generation in 3d. In *ICML*, 2022. (cit. on pp. 1, 3, 7, and 8)

[5] Pedro O. Pinheiro, Joshua Rackers, Joseph Kleinhenz, Michael Maser, Omar Mahmood, Andrew Martin Watkins, Stephen Ra, Vishnu Sresht, and Saeed Saremi. 3d molecule generation by denoising voxel grids. In *NeurIPS*, 2023. (cit. on pp. 1, 3, 5, 6, 7, 8, 18, and 21)

[6] Vıctor Garcia Satorras, Emiel Hoogeboom, and Max Welling. E (n) equivariant graph neural networks. In *ICML*, 2021. (cit. on pp. 1 and 3)

[7] Mario Geiger and Tess Smidt. e3nn: Euclidean neural networks. *arXiv:2207.09453*, 2022. (cit. on p. 1)

[8] Keyulu Xu, Weihua Hu, Jure Leskovec, and Stefanie Jegelka. How powerful are graph neural networks? *ICLR*, 2019. (cit. on p. 1)

[9] 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 *AAAI*, 2019. (cit. on p. 1)

[10] Sergey N Pozdnyakov and Michele Ceriotti. Incompleteness of graph convolutional neural networks for points clouds in three dimensions. *arXiv:2201.07136*, 2022. (cit. on p. 1)

[11] Yiheng Xie, Towaki Takikawa, Shunsuke Saito, Or Litany, Shiqin Yan, Numair Khan, Federico Tombari, James Tompkin, Vincent sitzmann, and Srinath Sridhar. Neural fields in visual computing and beyond. *Computer Graphics Forum*, 2022. (cit. on p. 1)

[12] Colin A Grambow, Hayley Weir, Christian N Cunningham, Tommaso Biancalani, and Kangway V Chuang. CREMP: Conformer-rotamer ensembles of macrocyclic peptides for machine learning. *arXiv:2305.08057*, 2023. (cit. on pp. 2, 7, and 18)

[13] Simon Axelrod and Rafael Gomez-Bombarelli. Geom, energy-annotated molecular conformations for property prediction and molecular generation. *Scientific Data*, 2022. (cit. on pp. 2 and 18)

[14] Kenneth O Stanley. Compositional pattern producing networks: A novel abstraction of development. *Genetic programming and evolvable machines*, 2007. (cit. on p. 2)

[15] Eric R Chan, Marco Monteiro, Petr Kellnhofer, Jiajun Wu, and Gordon Wetzstein. pi-gan: Periodic implicit generative adversarial networks for 3d-aware image synthesis. In *CVPR*, 2021. (cit. on p. 2)

[16] Kai Wang, Zhaopan Xu, Yukun Zhou, Zelin Zang, Trevor Darrell, Zhuang Liu, and Yang You. Neural network diffusion. *arXiv:2402.13144*, 2024. (cit. on p. 2)

[17] Yilun Du, Katie Collins, Josh Tenenbaum, and Vincent Sitzmann. Learning signal-agnostic manifolds of neural fields. *NeurIPS*, 2021. (cit. on p. 2)

[18] Zhiqin Chen and Hao Zhang. Learning implicit fields for generative shape modeling. In *CVPR*, 2019. (cit. on pp. 2 and 3)

[19] Lars Mescheder, Michael Oechsle, Michael Niemeyer, Sebastian Nowozin, and Andreas Geiger. Occupancy networks: Learning 3d reconstruction in function space. In *CVPR*, 2019. (cit. on pp. 2, 3, and 5)

[20] Jeong Joon Park, Peter Florence, Julian Straub, Richard Newcombe, and Steven Lovegrove. DeepSDF: Learning continuous signed distance functions for shape representation. In *CVPR*, 2019. (cit. on pp. 2, 3, and 4)

[21] Mateusz Michalkiewicz, Jhony K Pontes, Dominic Jack, Mahsa Baktashmotlagh, and Anders Eriksson. Implicit surface representations as layers in neural networks. In *ICCV*, 2019. (cit. on p. 2)

[22] Vincent Sitzmann, Michael Zollhöfer, and Gordon Wetzstein. Scene representation networks: Continuous 3d-structure-aware neural scene representations. *NeurIPS*, 2019. (cit. on p. 2)

[23] Ben Mildenhall, Pratul P. Srinivasan, Matthew Tancik, Jonathan T. Barron, Ravi Ramamoorthi, and Ren Ng. NeRF: Representing scenes as neural radiance fields for view synthesis. In *ECCV*, 2020. (cit. on p. 2)

[24] Zhengqi Li, Simon Niklaus, Noah Snavely, and Oliver Wang. Neural scene flow fields for space-time view synthesis of dynamic scenes. In *CVPR*, 2021. (cit. on p. 2)

[25] Zeyuan Chen, Yinbo Chen, Jingwen Liu, Xingqian Xu, Vidit Goel, Zhangyang Wang, Humphrey Shi, and Xiaolong Wang. Videoinr: Learning video implicit neural representation for continuous space-time super-resolution. In *ICCV*, 2022. (cit. on p. 2)

[26] Maziar Raissi, Paris Perdikaris, and George E Karniadakis. Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations. *Journal of Computational Physics*, 2019. (cit. on p. 2)

[27] Yuan Yin, Matthieu Kirchmeyer, Jean-Yves Franceschi, Alain Rakotomamonjy, and patrick gallinari. Continuous PDE dynamics forecasting with implicit neural representations. In *ICLR*, 2023. (cit. on pp. 2, 5, and 18)

[28] Nasim Rahaman, Aristide Baratin, Devansh Arpit, Felix Draxler, Min Lin, Fred Hamprecht, Yoshua Bengio, and Aaron Courville. On the spectral bias of neural networks. In *ICML*, 2019. (cit. on p. 2)

[29] Vincent Sitzmann, Julien N. P. Martel, Alexander W. Bergman, David B. Lindell, and Gordon Wetzstein. Implicit neural representations with periodic activation functions. In *NeurIPS*, 2020. (cit. on p. 2)

[30] Matthew Tancik, Pratul P. Srinivasan, Ben Mildenhall, Sara Fridovich-Keil, Nithin Raghavan, Utkarsh Singhal, Ravi Ramamoorthi, Jonathan T. Barron, and Ren Ng. Fourier features let networks learn high frequency functions in low dimensional domains. In *NeurIPS*, 2020. (cit. on p. 2)

[31] Rizal Fathony, Anit Kumar Sahu, Devin Willmott, and J. Zico Kolter. Multiplicative filter networks. In *ICLR*, 2021. (cit. on pp. 2, 4, 5, and 18)

[32] Diederik P Kingma and Max Welling. Auto-encoding variational bayes. *ICML*, 2014. (cit. on p. 3)

[33] Emilien Dupont, Yee Whye Teh, and Arnaud Doucet. Generative models as distributions of functions. *AISTATS*, 2022. (cit. on pp. 3 and 17)

[34] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. *NeurIPS*, 2014. (cit. on p. 3)

[35] Jascha Sohl-Dickstein, Eric Weiss, Niru Maheswaranathan, and Surya Ganguli. Deep unsupervised learning using nonequilibrium thermodynamics. In *ICML*, 2015. (cit. on p. 3)

[36] Emilien Dupont, Hyunjik Kim, S. M. Ali Eslami, Danilo Jimenez Rezende, and Dan Rosenbaum. From data to functa: Your data point is a function and you can treat it like one. In *ICML*, 2022. (cit. on pp. 3, 7, and 18)

[37] Ziya Erkoç, Fangchang Ma, Qi Shan, Matthias Nießner, and Angela Dai. Hyperdiffusion: Generating implicit neural fields with weight-space diffusion. In *ICCV*, 2023. (cit. on p. 3)

[38] Gene Chou, Yuval Bahat, and Felix Heide. Diffusion-sdf: Conditional generative modeling of signed distance functions. In *ICCV*, 2023. (cit. on p. 3)

[39] Biao Zhang, Jiapeng Tang, Matthias Niessner, and Peter Wonka. 3dshape2vecset: A 3d shape representation for neural fields and generative diffusion models. *ACM Transactions on Graphics (TOG)*, 2023. (cit. on p. 3)

[40] Aviv Navon, Aviv Shamsian, Idan Achituve, Ethan Fetaya, Gal Chechik, and Haggai Maron. Equivariant architectures for learning in deep weight spaces. In *ICML*, 2023. (cit. on p. 3)

[41] Matthias Bauer, Emilien Dupont, Andy Brock, Dan Rosenbaum, Jonathan Richard Schwarz, and Hyunjik Kim. Spatial functa: Scaling functa to imagenet classification and generation. *arXiv:2302.03130*, 2023. (cit. on p. 3)

[42] Luca De Luigi, Adriano Cardace, Riccardo Spezialetti, Pierluigi Zama Ramirez, Samuele Salti, and Luigi di Stefano. Deep learning on implicit neural representations of shapes. In *ICLR*, 2023. (cit. on p. 3)

[43] Allan Zhou, Kaien Yang, Yiding Jiang, Kaylee Burns, Winnie Xu, Samuel Sokota, J Zico Kolter, and Chelsea Finn. Neural functional transformers. In *NeurIPS*, 2023. (cit. on p. 3)

[44] Chelsea Finn, Pieter Abbeel, and Sergey Levine. Model-agnostic meta-learning for fast adaptation of deep networks. In *ICML*, 2017. (cit. on p. 3)

[45] Niklas WA Gebauer, Michael Gastegger, and Kristof T Schütt. Generating equilibrium molecules with deep neural networks. *arXiv:1810.11347*, 2018. (cit. on p. 3)

[46] Niklas Gebauer, Michael Gastegger, and Kristof Schütt. Symmetry-adapted generation of 3d point sets for the targeted discovery of molecules. *NeurIPS*, 2019. (cit. on p. 3)

[47] Youzhi Luo and Shuiwang Ji. An autoregressive flow model for 3d molecular geometry generation from scratch. In *ICLR*, 2022. (cit. on p. 3)

[48] Jonas Köhler, Leon Klein, and Frank Noé. Equivariant flows: exact likelihood generative learning for symmetric densities. In *ICML*, 2020. (cit. on p. 3)

[49] Victor Garcia Satorras, Emiel Hoogeboom, Fabian Fuchs, Ingmar Posner, and Max Welling. E(n) equivariant normalizing flows. *NeurIPS*, 2021. (cit. on p. 3)

[50] Danilo Rezende and Shakir Mohamed. Variational inference with normalizing flows. In *ICML*, 2015. (cit. on p. 3)

[51] Lei Huang, Hengtong Zhang, Tingyang Xu, and Ka-Chun Wong. Mdm: Molecular diffusion model for 3d molecule generation. In *AAAI*, 2023. (cit. on p. 3)

[52] Lemeng Wu, Chengyue Gong, Xingchao Liu, Mao Ye, and Qiang Liu. Diffusion-based molecule generation with informative prior bridges. *NeurIPS*, 2022. (cit. on p. 3)

[53] Minkai Xu, Alexander S Powers, Ron O Dror, Stefano Ermon, and Jure Leskovec. Geometric latent diffusion models for 3d molecule generation. In *ICML*, 2023. (cit. on pp. 3 and 7)

[54] Clement Vignac, Nagham Osman, Laura Toni, and Pascal Frossard. Midi: Mixed graph and 3d denoising diffusion for molecule generation. In *ECML*, 2023. (cit. on pp. 3, 7, 8, 10, 18, and 24)

[55] Chenqing Hua, Sitao Luan, Minkai Xu, Zhitao Ying, Jie Fu, Stefano Ermon, and Doina Precup. Mudiff: Unified diffusion for complete molecule generation. In *Learning on Graphs Conference*, 2024. (cit. on p. 3)

[56] Xingang Peng, Jiaqi Guan, Qiang Liu, and Jianzhu Ma. Moldiff: Addressing the atom-bond inconsistency problem in 3d molecule diffusion generation. In *ICML*, 2023. (cit. on pp. 3 and 24)

[57] Miha Skalic, José Jiménez, Davide Sabbadin, and Gianni De Fabritiis. Shape-based generative modeling for de novo drug design. *Journal of chemical information and modeling*, 2019. (cit. on p. 3)

[58] Matthew Ragoza, Tomohide Masuda, and David Ryan Koes. Learning a continuous representation of 3d molecular structures with deep generative models. *NeurIPS, MLSB Workshop*, 2020. (cit. on pp. 3 and 8)

[59] Ewa Nowara, Pedro Pinheiro, Sai Mahajan, Omar Abul'atta, Andrew Watkins, Saeed Saremi, and Michael Maser. Nebula: Neural empirical bayes under latent representations for efficient and controllable design of molecular libraries. *ICML. Worshop on AI4Sciences*, 2024. (cit. on p. 3)

[60] Matthew Ragoza, Tomohide Masuda, and David Ryan Koes. Generating 3D molecules conditional on receptor binding sites with deep generative models. *Chemical science*, 2022. (cit. on p. 3)

[61] Xingang Peng, Shitong Luo, Jiaqi Guan, Qi Xie, Jian Peng, and Jianzhu Ma. Pocket2mol: Efficient molecular sampling based on 3D protein pockets. In *ICML*, 2022. (cit. on p. 3)

[62] Jiaqi Guan, Wesley Wei Qian, Xingang Peng, Yufeng Su, Jian Peng, and Jianzhu Ma. 3D equivariant diffusion for target-aware molecule generation and affinity prediction. *ICLR*, 2023. (cit. on pp. 3 and 8)

[63] Pedro O Pinheiro, Arian Jamasb, Omar Mahmood, Vishnu Sresht, and Saeed Saremi. Structure-based drug design by denoising voxel grids. *ICML*, 2024. (cit. on p. 3)

[64] Ilia Igashov, Hannes Stärk, Clément Vignac, Arne Schneuing, Victor Garcia Satorras, Pascal Frossard, Max Welling, Michael Bronstein, and Bruno Correia. Equivariant 3d-conditional diffusion model for molecular linker design. *Nature Machine Intelligence*, 2024. (cit. on p. 3)

[65] Minkai Xu, Lantao Yu, Yang Song, Chence Shi, Stefano Ermon, and Jian Tang. Geodiff: A geometric diffusion model for molecular conformation generation. In *ICLR*, 2022. (cit. on p. 3)

[66] Bowen Jing, Gabriele Corso, Jeffrey Chang, Regina Barzilay, and Tommi Jaakkola. Torsional diffusion for molecular conformer generation. *NeurIPS*, 2022. (cit. on p. 3)

[67] Yuyang Wang, Ahmed A. Elhag, Navdeep Jaitly, Joshua M. Susskind, and Miguel Angel Bautista. Generating molecular conformer fields, 2023. (cit. on p. 3)

[68] Lin Li, Chuan Li, and Emil Alexov. On the modeling of polar component of solvation energy using smooth gaussian-based dielectric function. *Journal of Theoretical and Computational Chemistry*, 2014. (cit. on p. 3)

[69] Michael J Willatt, Félix Musil, and Michele Ceriotti. Atom-density representations for machine learning. *The Journal of chemical physics*, 2019. (cit. on p. 3)

[70] Gabriele Orlando, Daniele Raimondi, Ramon Duran-Romaña, Yves Moreau, Joost Schymkowitz, and Frederic Rousseau. Pyuul provides an interface between biological structures and deep learning algorithms. *Nature communications*, 2022. (cit. on p. 3)

[71] Ethan Perez, Florian Strub, Harm de Vries, Vincent Dumoulin, and Aaron Courville. FiLM: Visual reasoning with a general conditioning layer. *AAAI*, 2018. (cit. on p. 4)

[72] Matthew Ragoza, Tomohide Masuda, and David Ryan Koes. Learning a continuous representation of 3d molecular structures with deep generative models. In *Neurips, Structural Biology workshop*, 2020. (cit. on p. 5)

[73] Saeed Saremi, Rupesh Kumar Srivastava, and Francis Bach. Universal smoothed score functions for generative modeling. *arXiv:2303.11669*, 2023. (cit. on p. 6)

[74] Saeed Saremi, Ji Won Park, and Francis Bach. Chain of log-concave Markov chains. *ICLR*, 2024. (cit. on p. 6)

[75] Herbert E Robbins. An empirical bayes approach to statistics. In *Breakthroughs in Statistics: Foundations and basic theory*. 1992. (cit. on p. 6)

[76] Koichi Miyasawa. An empirical Bayes estimator of the mean of a normal population. *Bulletin of the International Statistical Institute*, 1961. (cit. on p. 6)

[77] Saeed Saremi and Rupesh Kumar Srivastava. Multimeasurement generative models. *ICLR*, 2022. (cit. on p. 6)

[78] Nathan C Frey, Dan Berenberg, Joseph Kleinhenz, Isidro Hotzel, Julien Lafrance-Vanasse, Ryan Lewis Kelly, Yan Wu, Arvind Rajpal, Stephen Ra, Richard Bonneau, Kyunghyun Cho, Andreas Loukas, Vladimir Gligorijevic, and Saeed Saremi. Protein discovery with discrete walk-jump sampling. In *ICLR*, 2024. (cit. on pp. 6 and 7)

[79] Xiang Cheng, Niladri S. Chatterji, Peter L. Bartlett, and Michael I. Jordan. Underdamped Langevin MCMC: A non-asymptotic analysis. In *COLT*, 2018. (cit. on p. 6)

[80] Matthias Sachs, Benedict Leimkuhler, and Vincent Danos. Langevin dynamics with variable coefficients and nonconservative forces: from stationary states to numerical methods. *Entropy*, 2017. (cit. on p. 7)

[81] 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*, 2018. (cit. on pp. 7 and 18)

[82] Simon Axelrod and Rafael Gómez-Bombarelli. Geom, energy-annotated molecular conformations for property prediction and molecular generation. *Scientific Data*, 9(1):185, 2022. (cit. on p. 7)

[83] Raghunathan Ramakrishnan, Pavlo O Dral, Matthias Rupp, and O Anatole Von Lilienfeld. Quantum chemistry structures and properties of 134 kilo molecules. *Scientific data*, 2014. (cit. on p. 7)

[84] Colin A. Grambow, Hayley Weir, Nathaniel L. Diamant, Alex M. Tseng, Tommaso Biancalani, Gabriele Scalia, and Kangway V. Chuang. Ringer: Rapid conformer generation for macrocycles with sequence-conditioned internal coordinate diffusion. *arXiv*, 2023. (cit. on pp. 7 and 10)

[85] Sreyas Mohan, Zahra Kadkhodaie, Eero P. Simoncelli, and Carlos Fernandez-Granda. Robust and interpretable blind image denoising via bias-free convolutional neural networks. In *ICLR*, 2020. (cit. on p. 7)

[86] Arne Schneuing, Yuanqi Du, Charles Harris, Arian Jamasb, Ilia Igashov, Weitao Du, Tom Blundell, Pietro Lió, Carla Gomes, Max Welling, et al. Structure-based drug design with equivariant diffusion models. *preprint arXiv:2210.13695*, 2022. (cit. on p. 8)

[87] Noel M O'Boyle, Michael Banck, Craig A James, Chris Morley, Tim Vandermeersch, and Geoffrey R Hutchison. Open babel: An open chemical toolbox. *Journal of cheminformatics*, 2011. (cit. on p. 8)

[88] Greg Landrum. Rdkit: Open-source cheminformatics software, 2016. (cit. on pp. 8 and 18)

[89] Charles Harris, Kieran Didi, Arian R Jamasb, Chaitanya K Joshi, Simon V Mathis, Pietro Lio, and Tom Blundell. Benchmarking generated poses: How rational is structure-based drug design with generative models? *arXiv preprint arXiv:2308.07413*, 2023. (cit. on p. 8)

[90] Anthony K Rappé, Carla J Casewit, KS Colwell, William A Goddard III, and W Mason Skiff. Uff, a full periodic table force field for molecular mechanics and molecular dynamics simulations. *Journal of the American chemical society*, 1992. (cit. on p. 8)

[91] G Richard Bickerton, Gaia V Paolini, Jérémy Besnard, Sorel Muresan, and Andrew L Hopkins. Quantifying the chemical beauty of drugs. *Nature chemistry*, 2012. (cit. on p. 8)

[92] Peter Ertl and Ansgar Schuffenhauer. Estimation of synthetic accessibility score of drug-like molecules based on molecular complexity and fragment contributions. *Journal of cheminformatics*, 2009. (cit. on p. 8)

[93] Clement Vignac and Pascal Frossard. Top-n: Equivariant set and graph generation without exchangeability. *ICLR*, 2022. (cit. on p. 8)

[94] GN t Ramachandran and V Sasisekharan. Conformation of polypeptides and proteins. *Advances in protein chemistry*, 23:283–437, 1968. (cit. on p. 10)

[95] Xingang Peng, Jiaqi Guan, Qiang Liu, and Jianzhu Ma. MolDiff: Addressing the atom-bond inconsistency problem in 3D molecule diffusion generation. In Andreas Krause, Emma Brunskill, Kyunghyun Cho, Barbara Engelhardt, Sivan Sabato, and Jonathan Scarlett, editors, *Proceedings of the 40th International Conference on Machine Learning*, volume 202 of *Proceedings of Machine Learning Research*, pages 27611–27629. PMLR, 23–29 Jul 2023. (cit. on p. 10)

[96] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. In *MICCAI*, 2015. (cit. on p. 17)

[97] Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. In *ICLR*, 2019. (cit. on p. 18)

[98] David Sehnal, Sebastian Bittrich, Mandar Deshpande, Radka Svobodová, Karel Berka, Václav Bazgier, Sameer Velankar, Stephen K Burley, Jaroslav Koča, and Alexander S Rose. Mol* Viewer: modern web app for 3D visualization and analysis of large biomolecular structures. *Nucleic Acids Research*, 2021. (cit. on p. 18)

Appendices

This supplementary material is organized as follows:

1. Appendix A includes a broader impact statement.
2. Appendix B includes extra implementation details.
3. Appendix C shows some additional analysis of latent space including some downstream task evaluation.
4. Appendix D presents results of a diffusion baseline on QM9.
5. Appendix E provides some ablation studies for the model.
6. Appendix F shows additional quantitative results.
7. Appendix H shows additional qualitative results.
8. Appendix G provides some comparison to a bond-diffusion baseline.
9. Appendix I includes the NeurIPS checklist.

A Broader Impact Statement

This work introduces some technical advancements in unconditional 3D molecule generation, an important component of molecular design and pharmaceutical research. A key advantage of our model is that it scales to larger molecules unlike existing models and has at least one order magnitude faster sampling time. Although extensive validation through wet-lab experiments and clinical trials is necessary, successful developments in this area have the potential to enhance human health, impacting a wide number of fields such as drug discovery, biology, materials science to cite a few. As with any technology, ensuring safe, ethical, and accountable deployment of these models is necessary to guarantee a positive impact on society.

B Implementation details

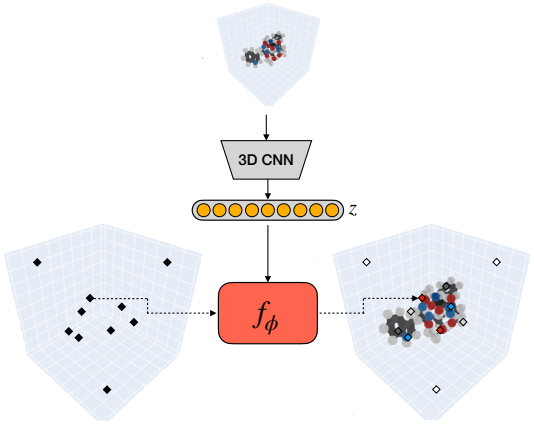

Figure 4: Auto-encoding approach for neural field representation. A voxelized representation of molecule is encoded int the latent space z with a 3D CNN. This representation is then decoded with a conditional MFN for any point x in space.

Here, we provide some more implementation details that complement Section 5.

Conditional neural field. The codes z are computed with an encoder that takes as input a low-resolution voxelized representation of the molecular field with grid dimension of $N \times 16^3$. We use $N = 5$ for QM9, $N = 8$ for GEOM-drugs and $N = 6$ for CREMP. We use resolution of .5Å to generate the low-resolution grid on QM9, 1Å on GEOM-drugs and 1.667Å for CREMP. Before

voxelizing the molecules, we first center the atoms around the tightest bounding box encapsulating the molecule, apply a random rotation to the atoms (each Euler angle rotated randomly between [0,2π)) and normalize their coordinates to the range of $[-1, 1]$. The encoder is a 3D CNN containing 4 residual blocks (number of hidden units 256, 512, 1024, 2048 for each block), where each block contains 3 convolutional layers followed by BatchNorm, ReLU and pooling layers (we use max pooling on the first three blocks and average pooling on the last one). The encoder has 145M on QM9 and 229M on GEOM-drugs and CREMP. In the case of FuncMol$_{\text{dec}}$, we do not use any encoder and directly optimize the codes z, one for each molecule in the dataset.

The neural field and codes are optimized over a free-form discretization grid $\mathcal{X}$, that changes at each iteration. For each training step, we sample a random training molecule and randomly pick $N = 4000$ points, half of the points are taken out of an uniform discretization grid of resolution .25Å, and the remaining points are sampled equally across cubes of size 3×3×3 and resolution .25Å, centered on each atom in the molecule. We found that this choice helped speed up training. For each point, we compute the atomic occupancy value for each atom using Equation (1).

The parameters of the conditional neural field are optimized with Adam. For FuncMol, we use a learning rate of 10^{-4} for the encoder and 5×10^{-4} for the decoder using a node of 2 A100 GPUs with a batch size of 96 per GPU. For FuncMol$_{\text{dec}}$, we efficiently scaled auto-decoding to large datasets by optimizing the codes with SparseAdam, using a learning rate 10^{-3}. The decoder optimizer is Adam with a learning rate of 10^{-3}. We train the models for 900 epochs on QM9, 300 epochs on GEOM-drugs and 1000 epochs on CREMP. Algorithm 1 and Algorithm 2 provide pseudocodes for learning the conditional neural field decoder and the latent codes (FuncMol$_{\text{dec}}$) or the encoder (FuncMol).

Algorithm 1: Auto-decoding conditional neural field training pseudo-code—Equation (2)

Input : $\mathcal{D}$ dataset of molecular fields, $\{z_v \leftarrow 0\}_{v\in\mathcal{D}}$ codes, $\phi \leftarrow \phi_0$ conditional MFN parameters; N number of points to sample

while *not converged* **do**

  for *batch* $\mathcal{B} \subset \mathcal{D}$ **do**

    Sample a discretization grid $\mathcal{X}$ and compute occupancy $v(x), \forall v \in \mathcal{B}, \forall x \in \mathcal{X}$

    $\ell_{\text{dec}}(\phi, \{z_v\}_{v\in\mathcal{B}}, \mathcal{X}) = \sum_{v\in\mathcal{B},x\in\mathcal{X}} \|f_\phi(x, z_v) - v(x)\|_2^2$

    $\{z_v\}_{v\in\mathcal{B}} \leftarrow \{z_v\}_{v\in\mathcal{B}} - \eta_z \nabla_z \ell_{\text{dec}}(\phi, \{z_v\}_{v\in\mathcal{B}}, \mathcal{X})$; `/* Update codes */`

    $\phi \leftarrow \phi - \eta_\phi \nabla_\phi \ell_{\text{dec}}(\phi, \{z_v\}_{v\in\mathcal{B}}, \mathcal{X})$; `/* Update decoder weights */`

Algorithm 2: Auto-encoding conditional neural field training pseudo-code—Equation (3)

Input : $\mathcal{D}$ dataset of molecular fields, $\psi \leftarrow \psi_0$ voxel encoder parameters; $\phi \leftarrow \phi_0$ conditional MFN parameters; N number of points to sample, uniform "low-resolution" voxel grid $\mathcal{G}$

while *not converged* **do**

  for *batch* $\mathcal{B} \subset \mathcal{D}$ **do**

    Sample a discretization grid $\mathcal{X}$ and compute occupancy $v(x), \forall v \in \mathcal{B}, \forall x \in \mathcal{X}$ and low-resolution voxel grid $\mathcal{G}_v, \forall v \in \mathcal{B}$.

    $\ell_{\text{dec}}(\phi, \psi, \mathcal{X}, \mathcal{B}) = \sum_{v\in\mathcal{B},x\in\mathcal{X}} \|f_\phi(x, \zeta_\psi(\mathcal{G}_v)) - v(x)\|_2^2$

    $\phi \leftarrow \phi - \eta_\phi \nabla_\phi \ell_{\text{dec}}(\phi, \psi, \mathcal{X}, \mathcal{B})$; `/* Update decoder weights */`

    $\psi \leftarrow \psi - \eta_\psi \nabla_\psi \ell_{\text{dec}}(\phi, \psi, \mathcal{X}, \mathcal{B})$; `/* Update encoder weights */`

Modulation code denoiser $\hat{z}_\theta$. Once the modulation codes and the conditional neural field are learned, we pre-process the codes to have zero mean and unit variance, then learn a denoiser in normalized space using $\sigma = 1.2$ on GEOM-drugs and CREMP and $\sigma = 2$ for QM9, following Algorithm 3.

Our denoiser has a projection linear layer (that embed the 1024 / 2048 code into a 6144 space) followed by several residual blocks, where each block contains (in this order): group normalization layer, ReLU non-linearity, fully-connected layer, normalization layer, ReLU non-linearity, drop-out with rate 0.3 for FuncMol$_{\text{dec}}$ or none for FuncMol and another fully-connected layer. We then add one final layer to go back to the original 1024 / 2048 code space. We use similar "skip-connections" as in the MLP denoiser of [33], adapted from 2D U-Net architectures [96]. For GEOM-drugs and CREMP, we consider a model with 1.9B parameters (12 residual blocks, 6144 hidden units). For QM9, we train a model of size 445M parameters (6 residual blocks, 4096 hidden units). The models

are trained with batch size 2048 on a single A100 GPU for 2500 epochs with AdamW [97] (learning rate 10^{-3}, weight decay 10^{-2}) and exponential moving average (EMA) with a decay of.9999. As [36], we use the following learning rate schedule: we warm-up the learning rate linearly from 0 to 3e-4 for the first 4000 iterations, then decay it proportionally to the square root of the iteration count. The pseudo-code is given in Algorithm 3.

Algorithm 3: Denoiser training pseudo-code - Equation (6)

Input : $\mathcal{D}_z = \{z_v\}_{v \in \mathcal{D}}$ normalized codes, denoiser $\hat{z}_\theta$
while not converged **do**
  for batch $\mathcal{B} \subset \mathcal{D}_z$ **do**
    $y \leftarrow z + \varepsilon, \quad \varepsilon \sim \mathcal{N}(0, \sigma^2 I_d)$
    $\ell_{\text{denoiser}}(\theta, \mathcal{B}) = \sum_{z \in \mathcal{B}} \|z - \hat{z}_\theta(y)\|_2^2$
    $\theta \leftarrow \theta - \nabla_\theta \ell_{\text{denoiser}}(\theta, \mathcal{B})$,
    $\theta_{\text{EMA}} \leftarrow \text{EMA}_{0.9999}(\theta_{\text{EMA}}, \theta)$

Sampling. The walk-jump sampling approach is very flexible and allows us to configure sampling in different ways. For example, we can choose the number of walk steps between jumps, the maximum number of walk steps per chain or the number of chains run in parallel. Different sampling hyperparameters can change the statistics of samples, e.g., samples that are close to each other on a sample chain will likely be similar molecules. Therefore, we decided to fix some sampling hyperparameters for benchmarking purposes. In all our quantitative experiments, we generate samples in the following way: (i) we initialize all the chains y_0 in parallel, (ii) we "walk" K steps with Langevin MCMC to sample smoothed codes y_K, and (iii) we "jump" with the denoiser (in a single step) to get the clean codes $\hat{z}_K$. In practice, we sampled 10000 molecules using 1000 MCMC steps for both QM9 and GEOM-drugs, and 10000 steps for CREMP on a single A100 GPU.

Algorithm 4: Sampling pseudo-code - the For loop corresponds to walk steps in Equation (7)

Input δ (step size), γ (friction), K (steps), denoiser $\hat{z}_\theta$ trained at noise level σ.
$y_0 \sim \mathcal{U}_d(\min_{z \in \mathcal{D}_z, i \in \{1 \cdots d\}} z_i, \max_{z \in \mathcal{D}_z, i \in \{1 \cdots d\}} z_i) + \mathcal{N}(0, \sigma^2 I_d)$
$u_0 \leftarrow 0$
for $k = 0, \ldots, K-1$ **do**
  $y_{k+1/2} \leftarrow y_k + \frac{\delta}{2} u_k$
  $g \leftarrow g_\theta(y_{k+1/2}) \triangleq (\hat{z}_\theta(y_{k+1/2}) - y_{k+1/2})/\sigma^2$; `/* score Equation (5) */`
  $u_{k+1/2} \leftarrow u_k + \frac{\delta}{2} g$
  $u_{k+1} \leftarrow \exp(-\gamma\delta) u_{k+1/2} + \frac{\delta}{2} g + \sqrt{\big(1 - \exp(-2\gamma\delta)\big)}\varepsilon, \quad \varepsilon \sim \mathcal{N}(0, I_d)$
  $y_{k+1} \leftarrow y_{k+1/2} + \frac{\delta}{2} u_{k+1}$
Output $\hat{z}_K \leftarrow \hat{z}_\theta(y_K)$; `/* jump step (denoising) */`

From codes to molecules. After generating modulation codes, we need to extract the atom types and coordinates from them. This is a constrained optimization problem, and we provide a simple algorithm to find its solution: (i) render a voxel grid representation of the molecule at resolution of .25Å (tensors of dimensions $5 \times 32 \times 32 \times 32$, $8 \times 64 \times 64 \times 64$ and $6 \times 96 \times 96 \times 96$ on QM9, GEOM-drugs and CREMP, respectively), (ii) find the peaks of the voxel grids—they correspond to a discretized version of the atomic coordinates—with a simple $3 \times 3 \times 3$ kernel, and (iii) find the local optima of the atomic coordinates with the approach described in Section 3.3. Our continuous refinement approach leverages L-BFGS with learning rate 1.0 and is batched across 100 molecules of same size.

Assets used in this work. Our code is available at `https://github.com/prescient-design/funcmol`. Our neural field code is based on the open source implementation of MFN from [31] and the conditional version from [27]. Our code for walk-jump sampling is based on the open source implementation of VoxMol from [5]. Our metrics are computed using code from [54] and RDKit [88]. Our datasets *GEOM-drugs* [13], *CREMP* [12] and QM9 [81] are downloaded from the corresponding webpages. We use the protein visualization tool of [98]. All these assets are available publicly and to our knowledge have a CC-BY 4.0 license.

C Analysis of the latent space

We perform three experiments to qualitatively explore the learned manifold and show empirically that it is well structured.

First, we pick several pairs of molecules and show the interpolation trajectory in latent modulation space. We project the interpolated codes back to the learned manifold of molecules via a noise/denoise operation. Figure 5 illustrates six trajectories, where we observe that molecules close in latent space share similar structure.

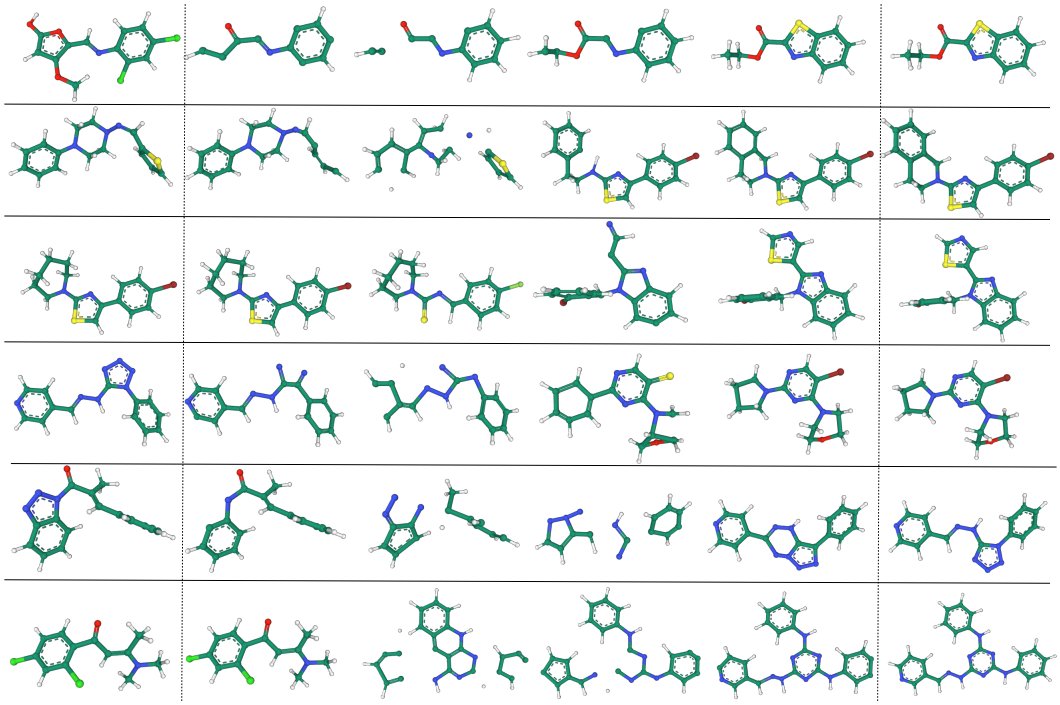

Figure 5: Interpolation in the latent modulation space for different pairs of molecules from GEOM-drugs. Each interpolated codes is protected back to the learned manifold of molecules via a noise/denoise operation. FuncMol produces semantically meaningful patterns in the interpolated space and we observe that molecules close in latent space share similar structure.

Second, we show t-SNE plots to demonstrate that the modulation space z encodes molecular properties of QM9. For four different properties, we use t-SNE to embed 400 molecules divided equally between those with the highest and those with the lowest property values. Figure 6 shows that molecules with similar property values cluster together.

Finally, we evaluate the latent codes on downstream tasks. We train a linear regression model on frozen latent codes (a.k.a. linear probing) to see how the learned modulations correlate with different properties. Figure 7 shows the scatter plots and Spearman correlation for four different properties. We observe that the codes are highly predictive of the considered properties, despite being trained in an unsupervised fashion.

D Diffusion baseline

We consider one additional model, $\text{FuncMol}_{\text{dec, diff}}$ for the auto-decoding setting. This model is similar to $\text{FuncMol}_{\text{dec}}$ but we sample codes with a diffusion model instead of walk-jump sampling. We use the same neural field and modulation codes as in $\text{FuncMol}_{\text{dec}}$ and we train a multi-level denoiser (with 1000 levels of noise) instead of a single-level one. The modulation codes are sampled like in standard diffusion models: we start from a Gaussian noise and iteratively apply the denoiser until we arrive on clean codes. We tried to train the diffusion variant of the model on GEOM-drugs,

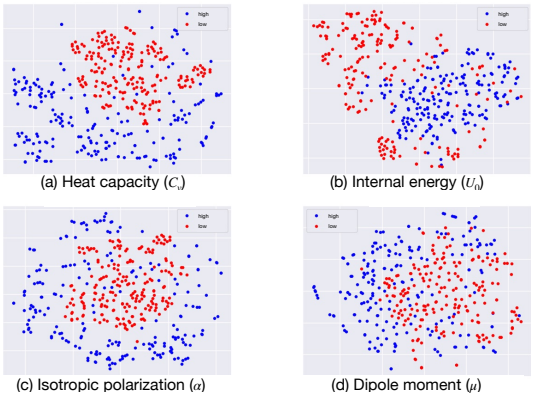

Figure 6: t-SNE plots of latent modulations codes of QM9 molecules for different molecular properties. For each plot, we pick 200 molecules from validation set with high value of a property (blue) and 200 with low value (red). We show results for four properties: (a) heat capacity (C_v), (b) internal energy (U_0), (c) isotropic polarization (α) and (d) dipole moment (μ).

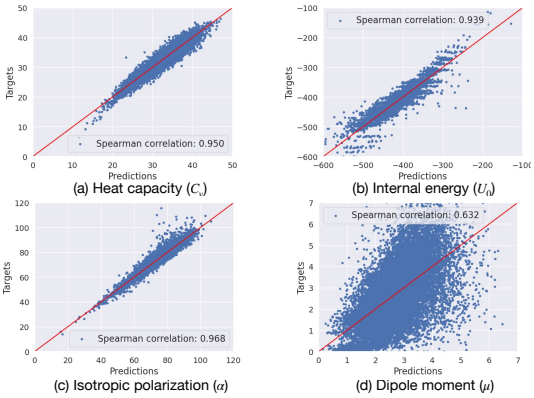

Figure 7: Performance of linear regression model (a.k.a linear probing) trained on modulation codes to predict molecular properties on QM9. We show the scatter plots and Spearman correlation for four different properties: (a) heat capacity (C_v), (b) internal energy (U_0), (c) isotropic polarization (α) and (d) dipole moment (μ).

but we were not successful (the generated molecules had very low stability). Table 4 shows the performance of the diffusion model relative to our other models. It performed worse than walk-jump sampling but was still able to generate good quality molecules.

Table 4: Comparing walk-jump sampling and diffusion model on QM9 results (10000 samples per model). ↑/↓ indicate that higher/lower numbers are better. The row data are randomly sampled molecules from the validation set.

	stable mol %↑	stable atom%↑	valid %↑	unique %↑	valency W_1↓	atom TV↓	bond TV↓	bond len W_1↓	bond ang W_1↓	time s/mol.↓
FuncMol	89.4	99.1	100.	93.1	.021	.012	.006	.004	1.49	0.05
FuncMol$_{\text{dec}}$	88.6	99.2	100.	81.1	.022	.066	.032	.006	1.21	0.05
FuncMol$_{\text{dec, diff}}$	70.8	97.3	95.8	81.1	.007	.034	.021	.006	1.25	0.07

E Ablations

E.1 Reconstruction quality

We analyze the reconstruction quality of the fields and molecules we compressed with latent codes.

Field reconstruction. We report in Table 5 the reconstruction performance of the molecular fields using our conditional MFN architecture and learned codes.

Table 5: Ablation: field reconstruction (whole training set).

dset	MSE ↓	PSNR ↑
GEOM-drugs	$2.8 \cdot 10^{-6}$	55.5
CREMP	$2.9 \cdot 10^{-6}$	55.4
QM9	$6.1 \cdot 10^{-6}$	52.1

Molecule reconstruction. To make more sense out of these raw reconstruction metrics, we show that the 3D molecules decoded from learned training codes are valid and stable molecules. This means that we successfully compressed the training data into low-dimensional latent vectors via our field-based representation. Table 6 reports metrics of molecules decoded from the learned training codes for GEOM-drugs and QM9. For each dataset, we display the metrics of the training molecules, the molecules decoded from our training codes and the molecules derived from a voxelized representation of the field at a resolution of .25Å. We observe that the molecules rendered from codes have better metrics than those derived from voxels showing the validity of our new representation. These results are an upper bound to the results in Table 2.

Table 6: Ablation: molecule reconstruction (sample of 4k).

dset	type	stable mol %↑	stable atom%↑	valid %↑	valency $W_{1\downarrow}$	atom TV↓	bond TV↓	bond len $W_{1\downarrow}$	bond ang $W_{1\downarrow}$
GEOM-drugs	*data*	99.9	99.9	100.	.001	.001	.025	.000	.05
	code	83.1	99.5	100.	.188	.007	.026	.004	.19
	voxel	83.7	99.4	93.8	.252	.006	.026	.001	.43
QM9	*data*	98.7	99.8	98.9	.001	.003	.000	.000	.12
	code	95.5	99.7	100.	.010	.009	.003	.001	.16
	voxel	92.5	99.4	98.8	.017	.009	.002	.002	.30

E.2 Atomic coordinate refinement

A big advantage of using neural fields is that we can represent signals, here molecules, in continuous space rather than in discrete space as in voxel representations. The continuous refinement introduced in Section 3.3 improves the quality of the molecules by finding more precise atomic coordinates. Table 7 shows the improvement of this continuous refinement over the refinement used in [5] that operates in discrete space: by going to continuous space, we overcome the limitations of discrete grids and substantially improve the stability of the molecules and the angle between bonds.

E.3 Neural field robustness to noise

Here, we analyze how the neural field is robust to noise on the modulation code space. Figure 8 illustrates how molecular stability and the distance between the distribution of bond angles change as we increasingly add noise to the codes. Each point consists of the average of the metric over 4000 random modulation codes from the validation set. Interestingly, the neural field is quite robust to noise as we see that the metrics unchanged even at a reasonable amount of noise. We believe that this code robustness to noise helps better learn the denoiser.

Table 7: Ablation: continuous refinement improvement on code reconstruction performance. Metrics computed with 4000 generated samples on validation reference set.

dset	FuncMol coord refine	stable mol %↑	stable atom%↑	valid %↑	valency $W_{1\downarrow}$	atom TV↓	bond TV↓	bond len $W_{1\downarrow}$	bond ang $W_{1\downarrow}$
GEOM-drugs	✓	83.1	99.5	100.	.188	.007	.026	.004	.189
	✗	78.8	96.2	100.	.090	.007	.018	.011	2.71
QM9	✓	95.5	99.7	100.	.010	.009	.003	.001	.158
	✗	79.1	96.4	100.	.009	.008	.002	.011	2.76

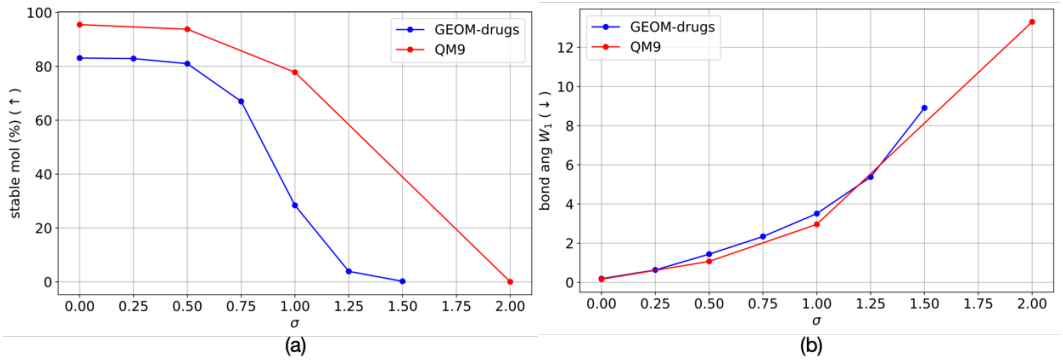

Figure 8: Ablation: code robustness to noise on GEOM-drugs (blue) and QM9 (red). Stable molecule (a) and bond angle distance (b) metrics as we increasingly add noise to the codes. Metrics are computed with 4000 generated samples on validation reference set.

E.4 Number of walk steps

Table 8 shows how the quality of molecules changes as we increase the number of walk steps K in the walk-jump sampling for FuncMol on GEOM-drugs. In this experiment, we sample 2000 samples and compute metrics w.r.t the validation set. First, we observe that some metrics improve (e.g., molecular stability) while other get worse (e.g., uniqueness). Second, we observe that the walk-jump chain is extremely stable, allowing us to perform as much as 50000 MCMC steps in the chain without breaking it. Finally, sampling time does not increase significantly: going from 500 to 50000 steps only results in a 10× increase in sampling time (this is because the sampling bottleneck is on finding the atomic coordinates).

Table 8: Ablation on the number of walk steps K on GEOM-drugs. Metrics computed with 2000 generated samples on test reference set.

K (n steps)	stable mol %↑	stable %↑	unique %↑	valency $W_{1\downarrow}$	atom TV↓	bond TV↓	bond len $W_{1\downarrow}$	bond ang $W_{1\downarrow}$	avg. t s/mol.↓
500	52.8	98.1	99.8	.235	.116	.031	.003	2.58	.279
1000	68.8	98.8	97.4	.246	.109	.051	.003	2.51	.298
2000	77.1	99.0	93.7	.247	.108	.068	.003	2.86	.337
5000	80.6	99.0	85.6	.247	.150	.091	.003	3.37	.456
10000	82.4	99.0	77.9	.247	.162	.109	.003	3.52	.654
20000	83.8	98.9	73.6	.252	.154	.130	.004	3.52	1.05
50000	84.9	99.0	66.6	.259	.166	.158	.003	3.44	2.24

E.5 Impact of resolution at decoding time

Finally, we measure the impact of resolution on sampling time and quality. As expected, sampling time becomes slower as we we have finer resolution. We also notice that finer resolution has better results than coarse ones (although the results stop improving). We chose resolution 0.25Å as it provides a good trade-off between performance and speed. The table below shows the results as a function of resolution.

Table 9: Ablation on the impact of resolution on sampling quality on GEOM-drugs. Metrics computed with 2000 generated samples on test reference set.

resolution Å	stable mol %↑	stable atom %↑	valid %↑	unique %↑	valency W_{1}↓	atom TV↓	bond TV↓	bond len W_{1}↓	bond ang W_{1}↓	avg. t s/mol.↓
0.167	68.6	98.8	100.	97.4	.246	.109	.052	.003	2.52	.89
0.25	68.8	98.8	100.	97.4	.250	.109	.052	.003	2.51	.29
0.5	58.8	97.7	100.	98.0	.247	.096	.051	.003	2.46	.08

F Additional quantitative results

We report some additional plots for evaluation on GEOM-drugs. See Section 5 for more details.

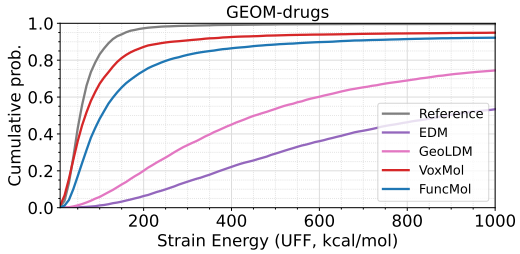

Figure 9: Cumulative distribution function of strain energy of generated molecules on GEOM-drugs based on 10000 molecules.

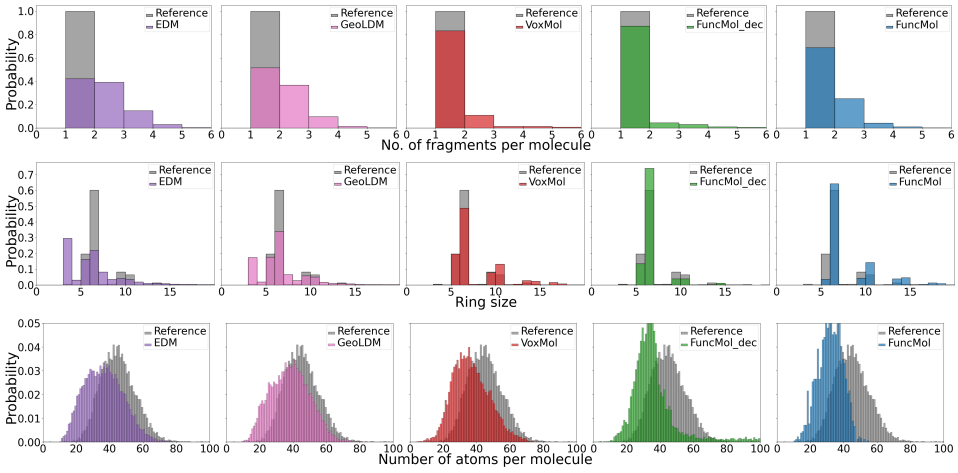

Figure 10: Histograms (over 10000 samples) showing (first row) distribution of number of fragments, (second) distribution of ring size, and (third) distribution of number of atoms per molecule.

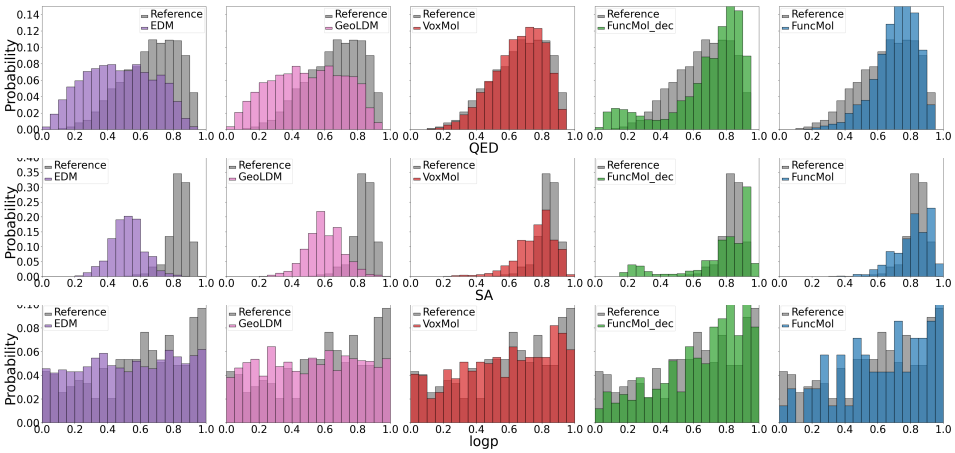

Figure 11: Histograms (over 10000 samples) showing (top) distribution of QED, (mid) SA and (bottom) log p.

G Comparison to bond-diffusion baselines

Some recent papers e.g., MolDiff [56], MiDi [54], show that incorporating extra information such as bonds and formal charges in point cloud-based approaches (e.g. EDM) improves the quality of the generated samples. These contributions are orthogonal to ours and are an interesting future addition to FuncMol, e.g. via additional channels in the molecular field.

We compare FuncMol to MolDiff despite different training assumptions, for completeness. MolDiff only incorporates bond information into the diffusion process, making it a simple representative baseline for this class of model. Since the weights for MolDiff with hydrogens were unavailable, we compared FuncMol using MolDiff's metrics and the MolDiff performance reported in their Appendix D.1, Table 8. The results are reported in Table 10. We observe that FuncMol achieves competitive results in most metrics despite not leveraging bond information.

Table 10: Comparison of MolDiff with H and FuncMol

	MolDiff with H	FuncMol
Validity ↑	0.957	1.000
Connectivity ↑	0.772	0.739
Succ. Rate ↑	0.739	0.739
Novelty ↑	1.000	0.992
Uniqueness ↑	1.000	0.977
Diversity ↑	0.427	0.810
Sim. Val. ↑	0.695	0.554
QED ↑	0.688	0.715
SA ↑	0.806	0.815
Lipinski ↑	4.868	5.000
RMSD ↓	1.032	1.088
JS bond lengths ↓	0.414	0.529
JS bond angles ↓	0.182	0.217
JS dihedral angles ↓	0.244	0.232

H Additional qualitative results

We display some generated molecules in Figures 12 and 13 and display a MCMC chain in Figure 14.

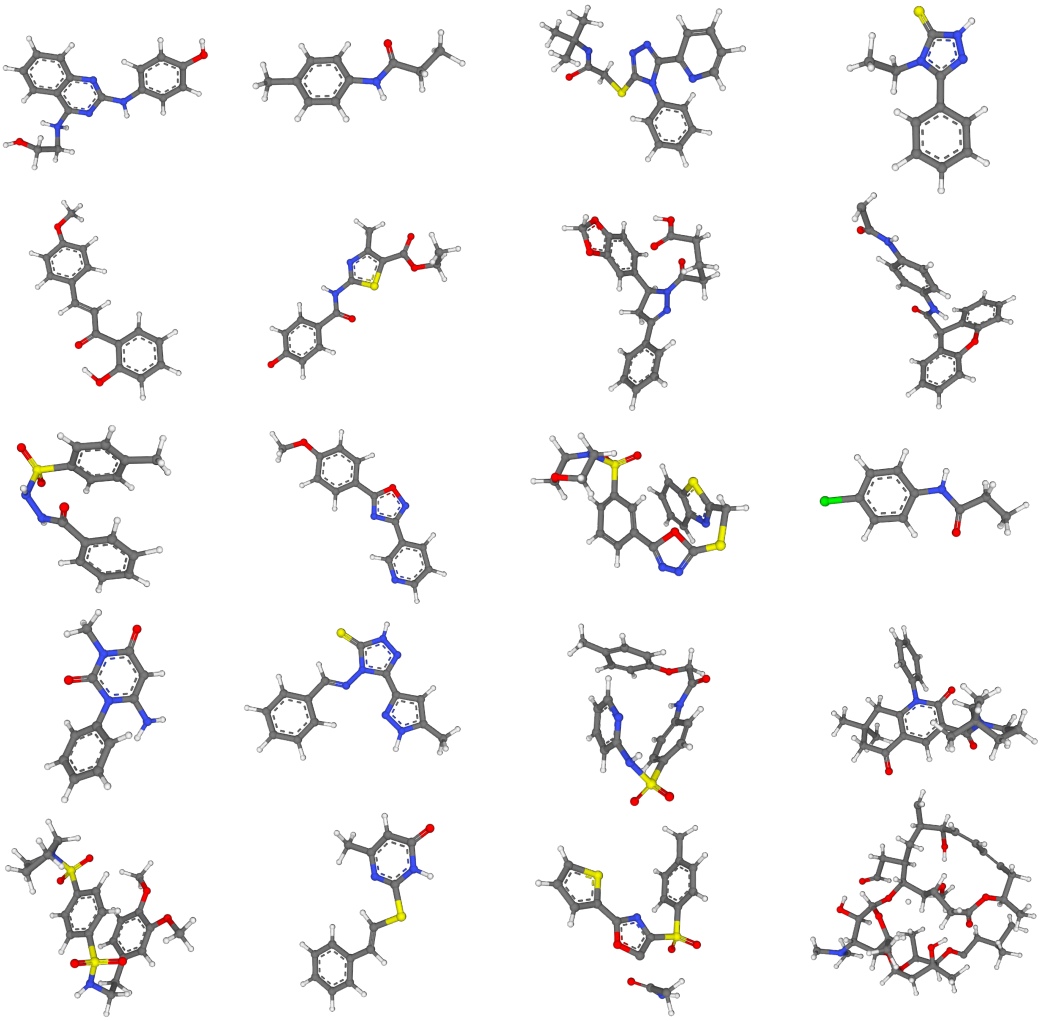

Figure 12: Generated samples from FuncMol trained on GEOM-drugs.

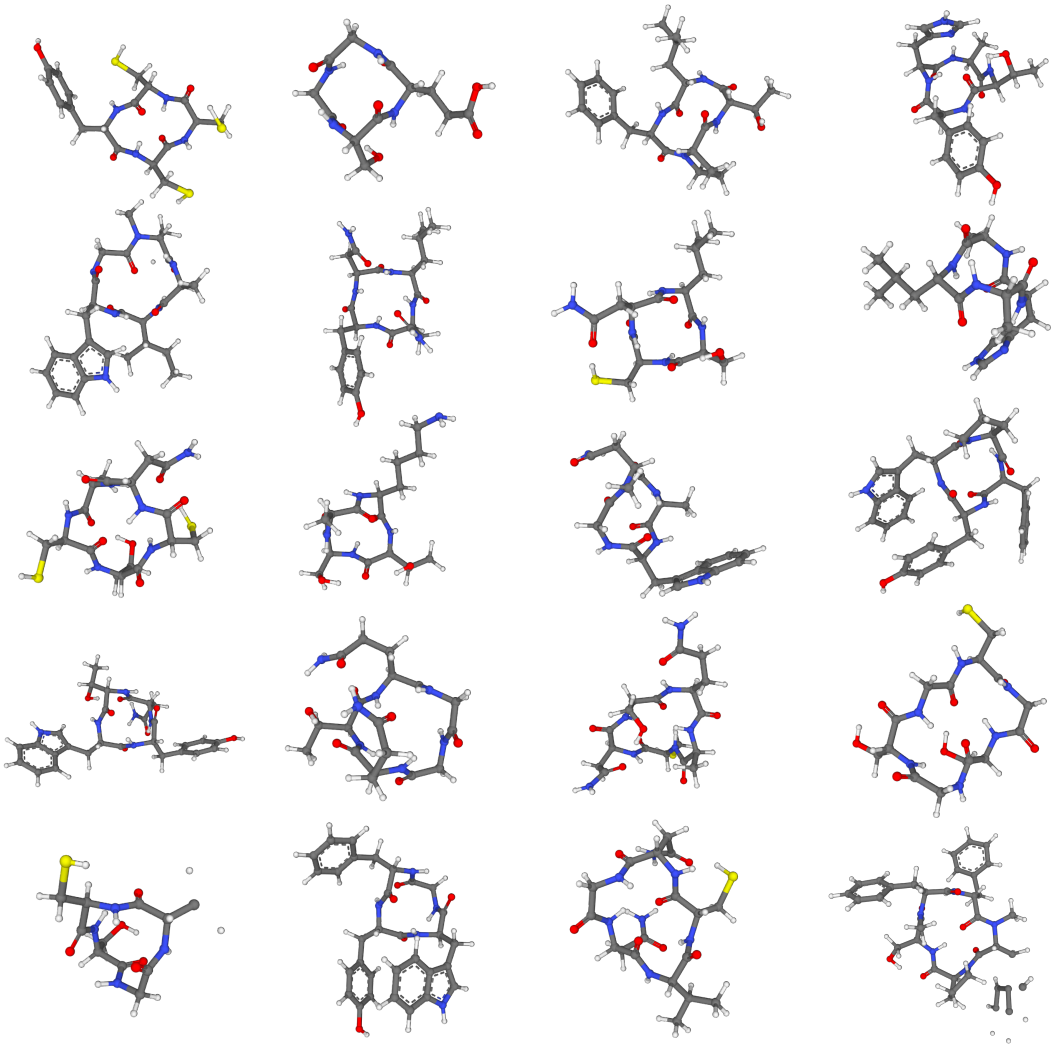

Figure 13: Generated samples from FuncMol trained on CREMP.

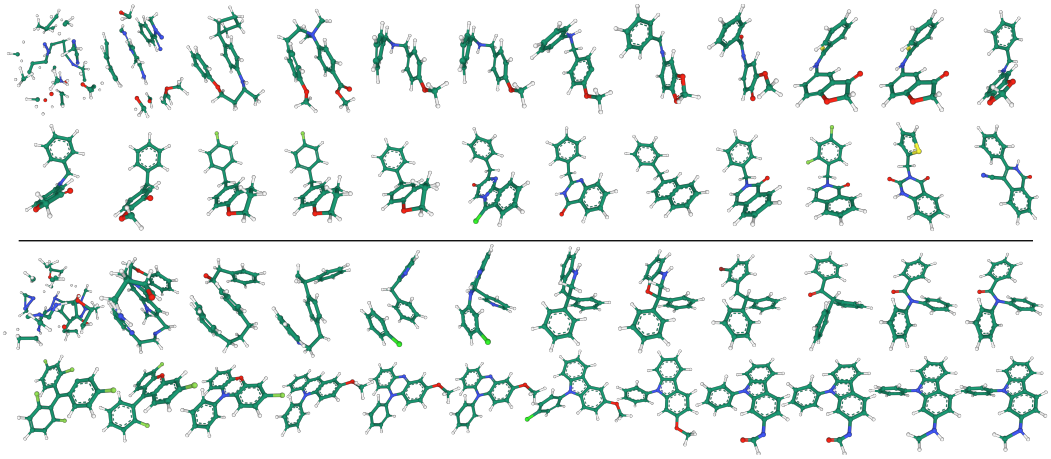

Figure 14: Two single MCMC chains generated by FuncMol, initialized randomly with different seeds (seen from left to right, top to bottom). Molecules are generated after each 200 "walk" steps.

I NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: we clearly state in the introduction in Section 1 our contributions and scope. Our paper does not make any important assumptions and we provide a discussion in Section 6.

   Guidelines:

   - The answer NA means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A No or NA answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: We provide a discussion on our model including limitations in Section 6

   Guidelines:

   - The answer NA means that the paper has no limitation while the answer No means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory Assumptions and Proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

Answer: [NA]

Justification: we do not provide theoretical results in this paper.

Guidelines:

- The answer NA means that the paper does not include theoretical results.
- All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
- All assumptions should be clearly stated or referenced in the statement of any theorems.
- The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
- Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
- Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental Result Reproducibility**

   Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

   Answer: [Yes]

   Justification: We detail all the hyperparameters, architectures and model training details in Appendix B.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - If the paper includes experiments, a No answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
   - If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
   - Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
   - While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
     (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
     (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
     (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
     (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

Answer: [Yes]

Justification: We will provide our code at https://github.com/prescient-design/funcmol. We provide all the hyperparameters used to run our experiments in Appendix B.

Guidelines:

- The answer NA means that paper does not include experiments requiring code.
- Please see the NeurIPS code and data submission guidelines (https://nips.cc/public/guides/CodeSubmissionPolicy) for more details.
- While we encourage the release of code and data, we understand that this might not be possible, so "No" is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
- The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (https://nips.cc/public/guides/CodeSubmissionPolicy) for more details.
- The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
- The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
- At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
- Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental Setting/Details**

Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer, etc.) necessary to understand the results?

Answer: [Yes]

Justification: we provide a succinct description of the experimental setting in Section 5 and provide more details in Appendix B.

Guidelines:

- The answer NA means that the paper does not include experiments.
- The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
- The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment Statistical Significance**

Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

Answer: [Yes]

Justification: we report 1-sigma error bars after repeating 3 times sampling in our main Tables 2 and 3.

Guidelines:

- The answer NA means that the paper does not include experiments.
- The authors should answer "Yes" if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
- The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).

- The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
- The assumptions made should be given (e.g., Normally distributed errors).
- It should be clear whether the error bar is the standard deviation or the standard error of the mean.
- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g. negative error rates).
- If error bars are reported in tables or plots, The authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments Compute Resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

   Justification: We report in Appendix B the compute resources required for for the experimental runs. In addition, we had access to around 20 A100 GPUs on an internal cluster for prototyping over the last 5 months.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code Of Ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

   Answer: [Yes]

   Justification: we reviewed the Code Of Ethics and confirm that the research in the paper conform to it.

   Guidelines:

   - The answer NA means that the authors have not reviewed the NeurIPS Code of Ethics.
   - If the authors answer No, they should explain the special circumstances that require a deviation from the Code of Ethics.
   - The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader Impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [Yes]

    Justification: cf Appendix A

    Guidelines:

    - The answer NA means that there is no societal impact of the work performed.
    - If the authors answer NA or No, they should explain why their work has no societal impact or why the paper does not address societal impact.

- Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

    Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pretrained language models, image generators, or scraped datasets)?

    Answer: [NA]

    Justification: the paper poses no such risks. the considered datasets are commonly used to benchmark generative models in the field.

    Guidelines:

    - The answer NA means that the paper poses no such risks.
    - Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
    - Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
    - We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

    Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

    Answer: [Yes]

    Justification: we included a discussion of all assets used in Appendix B.

    Guidelines:

    - The answer NA means that the paper does not use existing assets.
    - The authors should cite the original paper that produced the code package or dataset.
    - The authors should state which version of the asset is used and, if possible, include a URL.
    - The name of the license (e.g., CC-BY 4.0) should be included for each asset.
    - For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.

- If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
- For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New Assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

    Answer: [Yes]

    Justification: We provide all the required information to reproduce our models and the assest we used in Appendix B. Our license is CC-BY 4.0.

    Guidelines:

    - The answer NA means that the paper does not release new assets.
    - Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
    - The paper should discuss whether and how consent was obtained from people whose asset is used.
    - At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and Research with Human Subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

    Answer: [NA]

    Justification: the paper does not involve crowdsourcing nor research with human subjects.

    Guidelines:

    - The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
    - Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
    - According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional Review Board (IRB) Approvals or Equivalent for Research with Human Subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

    Answer: [NA]

    Justification: the paper does not involve crowdsourcing nor research with human subjects

    Guidelines:

    - The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
    - Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.

- We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
- For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.