Equivariant Diffusion for Molecule Generation in 3D

Pierre Aguié, Tess Breton and Mathias Grau

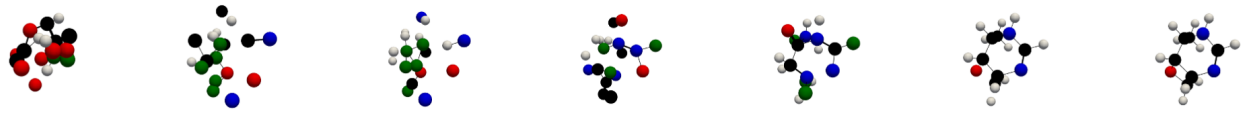


Figure 1: Visualization of different steps of the denoising process during the sampling of a molecule using an Equivariant Diffusion Model, from $t = T = 1000$ on the left to $t = 0$ on the right.

Introduction

This project is a study of the paper "Equivariant Diffusion for Molecule Generation in 3D" by Hoogeboom et al. [4], which introduces E(3) Equivariant Diffusion Models (EDMs) as a novel framework for 3D molecule generation. This approach addresses long-standing challenges such as stability, scalability and inference speed, representing a significant improvement over previous methods.
In this report, we first contextualize the paper and outline the methods and experiments conducted by the authors. Then, we highlight limitations of the proposed approach, investigate some decisions of the authors and present the findings of our own experiments. More specifically, Section 1 provides background on the paper, incuding its main contributions and related work. Next, Section 2 focuses on technical aspects of the authors' method, especially on the diffusion model. Section 3 presents the authors' main results, introducing the relevant datasets, metrics and methodology. Finally, in Section 4, we provide the results of our experiments and explore some limitations of the paper, exhibiting failure cases and questioning some of the authors' choices.

1 Context of the Paper

1.1 Molecule Generation

The paper is part of a body of research works exploring deep generative models for molecule generation. These models have been applied to molecular structures encoded as graphs, with the goal of generating stable, diverse and new molecules. These features are vital for applications such as *de novo* design (creating novel molecules from scratch with desired properties).

1.2 Related Work

The authors provide a comprehensive literature review to contextualize their method. They benchmark their model mainly against two alternative approaches for 3D molecule generation, which both exploit E(3) equivariance properties: G-SchNet [2] and Equivariant Normalizing Flows (E-NFs) [9]. E(3) equivariance refers to the physical symmetries of 3D molecules. It is introduced and formally defined in Subsection 2.2.
More precisely, G-SchNet uses autoregressive models to generate molecules, while E-NFs rely on continuous-time normalizing flows. Both approaches demonstrate the benefits of enforcing E(3) equivariance directly into the models' architecture. In E-NFs, the equivariance is achieved through Equivariant Graph Neural Networks (EGNNs), introduced in [10]. Similarly, EDMs leverage EGNNs to learn distributions, but differ by using diffusion as the generative process instead of normalizing flows. Yet, both of these methods have their limitations: G-SchNet introduces an artificial atom ordering, and E-NFs requires solving computationally demanding Ordinary Differential Equations (ODEs).
Beyond these 3D-based methods, previous literature on molecule generation also includes techniques that do not rely on 3D molecular representations. Such methods include one-shot graph-based frameworks, such as GraphVAE [6], to which the authors also compare their method.

1.3 Main Contributions

The EDM described in the paper is the first Denoising Diffusion Probabilistic Model (DDPM) to leverage E(3) equivariance. Upon publication, it achieved state-of-the-art performance on several molecule generation benchmarks. It ensures equivariance to physical symmetries, generates more stable molecules than previous methods and scales efficiently to larger datasets.
A key feature of the method is its flexibility in handling hydrogen atoms, either explicitly or implicitly. Unlike most models that treat hydrogens as standard elements, this approach was designed to work effectively without them by focusing exclusively on heavy atoms. In that case, hydrogens are added in a post-processing step to heavy atoms with incomplete valency, aligning with their expected reference valency. This strategy significantly reduces computational complexity by minimizing the amount of data to process, resulting in faster training and sampling.
The EDM was also designed to support conditional molecule generation, which is very promising for drug discovery where generating molecules with specific properties is often a key objective. The conditions available correspond to chemical properties provided in the training dataset, such as those in QM9[1]. Examples of such properties include polarizability α and dipole moment μ. However, for this project, we decided not to focus on this particular feature.

Geometric Data Analysis, December 2024, ENS Paris Saclay, M2 MVA

[1]The QM9 dataset is presented thoroughly in Subsection 3.1.

2 EDM Method

In this Section, we present the method proposed by the authors and provide details on the diffusion model, equivariance and EGNNs.

2.1 The Diffusion Model

2.1.1 Background on diffusion models. Diffusion models generate new data by transforming pure random noise into meaningful samples through a denoising process. The model is trained to reverse a noising procedure, called the diffusion process.

A diffusion process is a Markov process that progressively adds Gaussian noise to an input sample. For a sample x, at every time step $t \in \{0, \dots, T\}$, the noising process is defined as follows:

$$q(z_t|x) = \mathcal{N}(z_t|\alpha_t x, \sigma_t^2 I) \tag{1}$$

The parameters $\alpha_t \in \mathbb{R}^+$ and $\sigma_t \in \mathbb{R}^+$ respectively determine how much of the input signal is retained, and the amount of noise added. They usually follow a predefined schedule, which ensures that the signal-to-noise ratio (SNR) α_t^2/σ_t^2 gradually decreases[2].

From Equation (1), we can derive transition probabilities $q(z_t|z_s)$ between latent states, for $t > s$, which are also Gaussian. Then the complete noising process can be defined as:

$$q(z_0, z_1, \dots, z_T|x) := q(z_0|x) \prod_{t=1}^{T} q(z_t|z_{t-1}) \tag{2}$$

Using Bayes rule, we can show that the true denoising process, defined by the conditional posteriors $q(z_s|x, z_t)$, is also Gaussian:

$$q(z_s|x, z_t) = \mathcal{N}(z_s \mid \mu_{t\to s}(x, z_t), \sigma_{t\to s}^2 I) \tag{3}$$

Now, when generating new samples, the variable x is unkwown and precisely what we want to reach by the end of the process. It is thus approximated using a neural network ϕ, which provides $\hat{x} = \phi(z_t, t)$. This defines a generative transition distribution p:

$$p(z_s|z_t) = q(z_s|\hat{x}, z_t) = \mathcal{N}(z_s \mid \mu_{t\to s}(\hat{x}, z_t), \sigma_{t\to s}^2 I) \tag{4}$$

The denoising is then done step by step, starting from a random latent state $z_T \sim \mathcal{N}(0, I)$ and iteratively transitioning from z_t to z_{t-1}. The generated sample x is eventually derived from z_0.

2.1.2 Noising procedure for molecules. The EDM method generates molecules as sets of points, where each point represents an atom with its own position and features. Specifically, a molecule with M atoms is encoded as a set $\{(x_i, h_i)\}_{i=1}^M$, with $x_i \in \mathbb{R}^3$ representing the 3D coordinates of each atom, and $h_i \in \mathbb{R}^{n_f}$ the atomic features. These features typically include atom types (H, C, O…) and integer-valued atom charges. The bonds between atoms are not taken into account through the process, and every latent state is encoded similarly as molecules: $z_t = \{(z_{t,i}^x, z_{t,i}^h)\}_{i=1}^M$. Figure 2 provides a schematic representation of the noising process.

We now provide further details on how both continuous and categorical features are handled during the noising process. Adding noise to 3D coordinates is relatively straightforward, since they are continuous variables. However, the EDM requires the distribution p to be E(3) equivariant w.r.t. the variable x. Hence, p must be invariant to translations, which is impossible for a distribution on $\mathbb{R}^{3\times M}$ since it must integrate to one. Thus, the authors propose to

[2]Additional information and missing analytical expressions from this Subsection are provided in Appendix A.

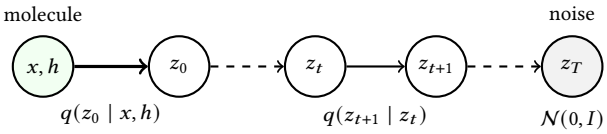
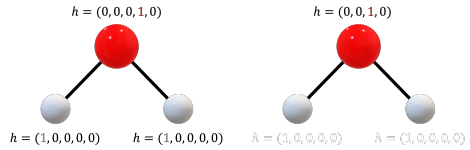


Figure 2: Noising process

make the center of gravity of the point clouds $\{z_{t,i}^x\}_{i=1}^M$ be zero at each time step, by setting $\sum_i z_{t,i}^x = 0$. This comes down to sampling z_t^x from the normal distribution on the subspace of $\mathbb{R}^{3\times M}$ where $\sum_i z_{t,i}^x = 0$. From now on, this distribution will be noted $\mathcal{N}_x$.

On the other hand, handling categorical features such as atom type is more complex. Representing them as integers is problematic, since it introduces irrelevant ordering biases. To address this, categorical features are represented as one-hot vectors. The example of H_2O is given in Figure 3 with QM9 encoding.



(a) With explicit hydrogens (b) With implicit hydrogens

Figure 3: Atom type one-hot encoding on H_2O

In Figure 4, we visualize different steps of this noising process on a butanol molecule ($C_4H_{10}O$).

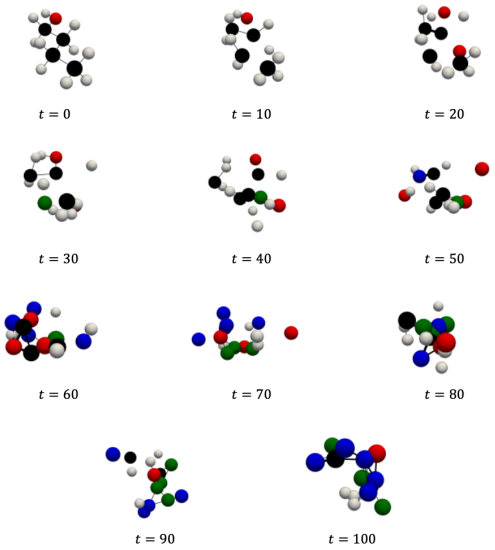


Figure 4: Molecular representations of latent states z_t for selected time steps $t \in \{0, \dots, T\}$, with $T = 100$, during the noising process of a butanol molecule. Bonds are shown indicatively, to give a sense of inter-atom distances.

2.1.3 Generative denoising procedure. To generate new samples, the noising process is reversed. An EGNN is trained to learn the denoising procedure, by predicting the noise at each step. More information on this network is given in Subsection 2.3. The generative process is described schematically in Figure 5.

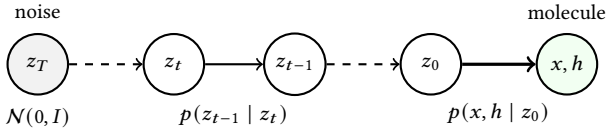


Figure 5: Generative denoising process

It is worth mentioning that this framework assumes that the number of atoms in the generated molecules is fixed beforehand. In practice, models are designed to handle the maximal number of atoms found in molecules of the training set. They pad (x, h) with zeros to ensure that all sets belong to the same space. During sampling, we first draw the number of atoms $M \sim p(M)$, with p computed on the training dataset. Then, we generate the coordinates and features $x, h \sim p(x, h|M)$ using the model.

2.1.4 Optimization Objective. Conceptually, the neural network ϕ should maximize the log-likelihood of the training dataset given the model: $\log p(\{(x, h)_i\}_i|\phi)$. But this objective is untractable, and the one used to optimize the model is a lower bound of the log-likelihood, derived in [3]. For the EDM, the lower bound is simplified by neglecting terms that are close to 0 due to specificities of the diffusion process and the discrete nature of the features h. The lower bound is further simplified by approximating some multiplicative factors by 1, and by having the neural network ϕ predict the noise $\hat{\varepsilon} = \phi(z_t, t)$ to add to the latent state z_t, instead of the next latent state. The final objective to minimize is then, for a predicted noise $\hat{\varepsilon}$ at time step t:

$$\mathcal{L}_t = \mathbb{E}_{\varepsilon_t \sim \mathcal{N}_x(0,I) \times \mathcal{N}(0,I)} \left[\|\varepsilon_t - \hat{\varepsilon}\|^2 \right] \tag{5}$$

In practice, at each iteration of the optimization algorithm that updates the parameters of ϕ, a time step t is chosen uniformly from $\{0, \dots, T\}$ and a noise ε_t is sampled from $\mathcal{N}(0, I)$. The x component of the noise is then shifted so that its center of gravity is 0: we replace ε_t^x with $\varepsilon_t^x - \sum_i \varepsilon_{t,i}^x$. Given the latent state $z_t = \alpha_t(x, h) + \sigma_t \varepsilon_t$, the model's parameters are then updated in a direction chosen to minimize $\|\varepsilon_t - \phi(z_t, t)\|^2$. More details on the derivation of this objective are given in Appendix B. Intuitively, the goal is to train the model to accurately predict the noise added at each step of the diffusion process.

2.1.5 From diffusion outputs to molecules. One critical step in Figure 5 is the last one, which transforms the denoised version z_0 into a real molecule. Indeed, z_0 only contains the denoised 3D coordinates x_i and features h_i of each atom $i \in \{1, \dots, M\}$. At this point, the h_i are not one-hot encoded and the bonds between atoms are unknown. We describe the authors' procedure below.
First, the categorical features are converted to one-hot vectors by taking the argmax of each feature. Next, building a real molecule requires determining the bonds between atoms, along with their type: simple, double, triple or none. The authors achieve this by looking at distances between the atoms and at their types, using a lookup table to map those to the appropriate bond type for each atom pair. This process is represented schematically in Figure 6, assuming that atom type is the only feature and using QM9 encoding.

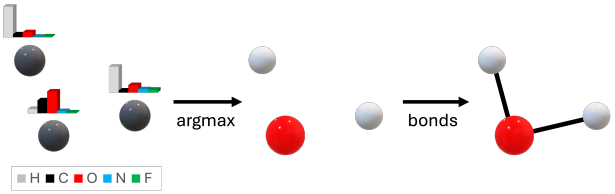


Figure 6: Last step of the generation, from z_0 to the molecule

2.2 Background on Equivariance

The method introduced in the paper leverages E(3) equivariance to generate molecules. We provide some background on this notion. The group of isometries of $\mathbb{R}^3$, which includes translations, symmetries and rotations, in noted $E(3)$. A conditional distribution $p(x|z)$ is said to be $E(3)$-equivariant if, for all actions $R \in E(3)$, we have:

$$p(Rx|Rz) = p(x|z). \tag{6}$$

The authors decided to have the conditional distribution of the generated molecule x given the input latent variable z verify this equivariance property, as chemical properties of molecules are invariant under the transformations of $E(3)$. This is why invariance to translations was ensured in Subsection **??**, by enforcing centers of gravity equal to zero. Similarly, for the invariance to the group $O(3)$ of rotations and symmetries, the authors used an EGNN, which we describe in the following Subsection.

2.3 Equivariant Graph Neural Networks

The generative denoising process is learned using a neural network ϕ. To ensure that the denoising distribution $p(z_s|z_t)$ is E(3) equivariant, the neural network's architecture must be designed to also respect equivariance properties. More specifically, ϕ should verify that for all $R \in O(3)$, for all (z_t^x, z_t^h), if

$$\hat{\varepsilon}^x, \hat{\varepsilon}^h = \phi(z_t^x, z_t^h, t), \tag{7}$$

then

$$R\hat{\varepsilon}^x, \hat{\varepsilon}^h = \phi(Rz_t^x, z_t^h, t). \tag{8}$$

As shown by the authors, if this property is respected, then the denoising process p is equivariant. The EGNNs introduced in [10] respect this property, and are used in the EDM architecture. In this setting, molecules are seen as a fully connected graph with M nodes, where each node's attributes are the position x_i and the features h_i of the corresponding atom.
An EGGN is a sequence of Equivarient Graph Convolutional Layers (EGCLs). In each of these layers, for an input (x^l, h^l), messages $m_{i,j}$ between atoms i and j are computed as the output of a neural network taking as input h_i^l, h_j^l and $\|x_i^l - x_j^l\|^2$, which is invariant to transformations in $O(3)$. The output feature vector h^{l+1} is then computed as the output of another neural network, taking as input

h_i^l and a weighted average of the $m_{i,j}$s, where the weights are estimated through another neural network. The output position x_i^{l+1} is not computed as the direct output of a neural network, in order to ensure that (8) holds. Instead, a neural network ϕ_x estimates how much each atom at x_i^l is pulled towards or pushed away from the other atoms at x_j^l:

$$x_i^{l+1} = x_i^l + \sum_{j \neq i} \frac{x_i^l - x_j^l}{\|x_i^l - x_j^l\|^2 + 1} \phi_x(h_i^l, h_j^l, \|x_i^l - x_j^l\|^2) \qquad (9)$$

This formula ensures that transforming an input with the action $R \in O(3)$ results in the same transformation for the output, and gives a physical interpretation of the neural network ϕ_x. It could be interpreted as proportional to a force, which would move atoms to minimize the energy of the system, by taking into account atom types and charges.

3 Experimental results

In this Section, we present the main results obtained by the authors, along with their evaluation set-up and methodology.

3.1 Datasets

The authors conducted their experiments on the QM9 and GEOM-Drugs datasets, both described below.

QM9 [8] is a widely used benchmark in computational chemistry and machine learning, consisting of approximately 134k stable small molecules. Each molecule contains up to nine heavy atoms[3] (C, N, O and F) and comes with over a dozen chemical properties such as dipole moment μ or polarizability α. Figure 7 shows the distribution of molecule sizes, which is concentrated around the average $M = 18$. This imbalance already raises concerns about its potential impact on performance, which we will explore in Subsection 4.2.

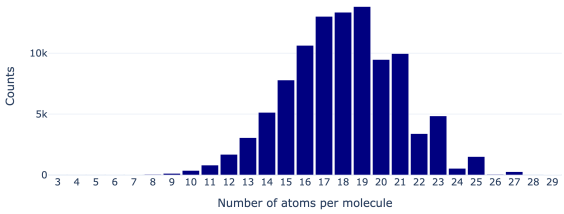


Figure 7: Histogram of the number of atoms per molecule in QM9 (including hydrogens)

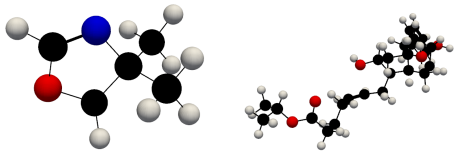
On the other hand, GEOM-Drugs [1] was designed to provide comprehensive molecular conformations for drug-like molecules. It contains larger and more complex molecules than QM9, along with some of their conformers and respective energies, but no chemical properties. Conformers are different spatial arrangements of a molecule's atoms, resulting from rotations around single bonds. All these conformers have the same molecular formula and the same bonds between atoms, but differ in their three-dimensional structure.

Table 1 summarizes the main metrics measuring the size of the molecules in QM9 and GEOM-Drugs, showing that the two datasets contain on molecules of different scales. Figure 8 shows one molecule from each training dataset.

[3] "Heavy atoms" are atoms other than hydrogen.

Table 1: Some metrics on QM9 and GEOM-Drugs

Attribute	QM9	GEOM-Drugs
Maximum number of atoms	29	181
Maximum number of heavy atoms	9	91
Average number of atoms	18	44
Number of molecules	134k	430k



(a) Example from QM9 (b) Example from GEOM-Drugs

Figure 8: Example molecules from each training dataset

3.2 Metrics

To benchmark their model, the authors refer mostly to metrics related to chemical properties of the molecules. More specifically, across a fixed number of generated molecules, models are compared on atom stability, molecule stability, and molecule validity. Atom stability is defined as the proportion of atoms with the right valency, and molecule stability is the proportion of molecules for which all atoms are stable. Molecule validity is the proportion of molecules that are chemically valid, as measured by RDKit [5].

The authors also compare models on their ability to generate unique and novel molecules. Uniqueness is measured in RDKit as the proportion of valid molecules that are structurally distinct from one another. RDKit also includes a novelty metric, which is discussed in Subsection 4.3.

When applicable, the authors compare the models on the final value of the objective at the end of training. This objective is the negative log-likelihood of the train dataset given the model. Note that, as explained in Subsection 2.1, the objective of the diffusion model is not exactly the negative log-likelihood, but an upper bound of it.

3.3 Performance

The authors' experiments show that the EDM outperforms all other 3D generative models trained on QM9 in terms of negative log-likelihood, atom and molecule stability, validity and uniqueness. These results show the benefits of using DDPMs over other methods, as well as the significantly positive impact of incorporating equivariance directly into the DDPM architecture.

Moreover, the EDM method is computationally more effective than previous approaches. This quicker training allows for extension to larger datasets with larger molecules, which can be interesting for applications of 3D molecule generation.

4 Limitations and Experiments

This last section explores limitations of the EDM method through experiments and questions some of the authors' choices.

4.1 Quality of Generated Molecules and Failure Cases

As acknowledged by the authors, despite great performance on all reported metrics, some generated molecules are of poor quality. Several generations are not chemically valid, and even among the valid ones, some samples show unrealistic structures such as unusually long or short cycles.
Starting with QM9, we first proposed the following experiment focusing on cycles. We generated 200 molecules, and examined the number and size of cycles within each molecule. The results are plotted in Figure 9 below.

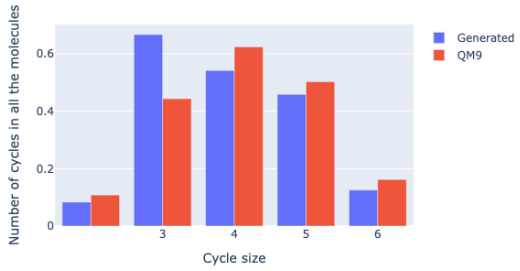


Figure 9: Normalized distribution of cycles size (None, 3, 4, 5, 6) in the generated outputs vs in QM9

The comparison between the generated molecules and the molecules from QM9 shows a clear difference in distribution, with a noticeable shift towards a higher number of small cycles in the generated outputs. While cycles play a crucial role in determining a molecule's chemical properties , they also introduce tension and impact its conformational stability. Small cycles, particularly those with fewer than four atoms, are highly strained. An example of unrealistic output featuring such small cycles is shown in Figure 10.

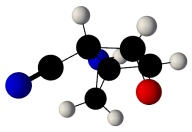


Figure 10: Unrealistic generated output of the EDM trained on QM9 (two cycles of length 3, and one cycle of length 4)

Next, on GEOM-Drugs, the authors expose some failure cases. Several generated molecules contain unrealistically long cycles, and a few are made of multiple disjoint components that do not form a single molecule. The authors point out that similar issues were observed in early stages of the training on QM9, suggesting that further training on GEOM-Drugs could potentially solve this issue. As explained in [7], these failure cases are, to some extent, due to the fact that the diffusion model first chooses atom types and positions, adding bonds only later using a lookup table. We also hypothesize that the number of atoms in the molecule might impact performance. The EDM has indeed significantly more failure cases when trained on GEOM-Drugs than on QM9, the key difference between the datasets being the number of atoms in their molecules. It seems that a higher atom count in latent states makes it more challenging for the model to generate chemically coherent structures. In the next subsection, we propose an experiment to test this hypothesis.

4.2 Impact of Molecule Size on Performance

Based on Figure 7, we built an experiment to evaluate the performance of the model on each number of atoms individually. For every $M \in \{3, 4, \ldots, 29\}$, we generated $N = 100$ molecules and computed the following metrics over each N-sample: atom stability, molecule stability, novelty, uniqueness and validity. The results are displayed in Figure 11.

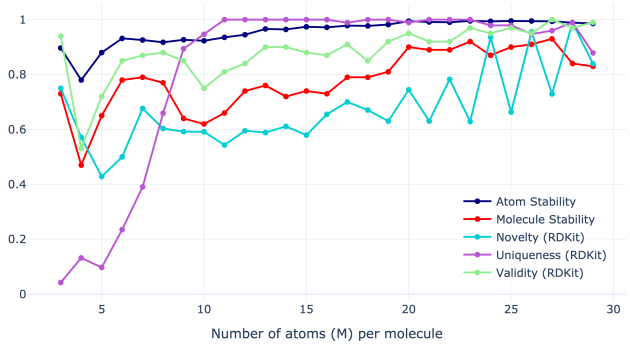


Figure 11: Metrics over N=100 samples for each M

While we expected more failure cases with larger values of M, we were surprised to see that the model performed well on all metrics for $M \geq 25$ despite few training examples. Notably, the generation was successful for $M = 28$, even though the training set contained no molecule of that size. This can be attributed to the convolutional structure of EGCLs: rather than learning M distinct operations for each node at each layer, EGCLs learn a single "filter" which operates locally over neighborhoods in the input graph.
On the other hand, for small molecule sizes ($M \leq 10$), the results were not as good. However, some of them should be nuanced, especially regarding uniqueness and novelty, as there are only a limited number of valid outcomes to sample for such small sizes.

4.3 Choice of Metrics

Another important consideration is the choice of metrics used to benchmark the EDM and assess the quality of the generated molecules.
Evaluating the quality of generated samples in generation tasks is inherently challenging, and molecule generation is no exception. The primary concern is ensuring that chemical requirements are met, which stability metrics aim to address. However, we believe that the metrics chosen by the authors may overlook practical relevance, particularly in downstream applications. For instance, in drug discovery, metrics like synthetizability and novelty might

provide more valuable insights. Yet novelty can be questioned, as explained below, and synthesizability is hard to evaluate.
Moreover, it appears that evaluation metrics must currently be tailored to the specificities of each dataset. Finding a metric that applies well to both QM9 and GEOM-Drugs seems indeed particularly difficult. As noted in Appendix C of the paper, novelty is not meaningful on QM9 since this dataset is the exhaustive enumeration of all molecules that satisfy a predefined set of constraints. In this context, high novelty scores would suggest that the model somehow fails to capture some properties of the training data. Conversely, molecule stability is of limited relevance on GEOM-Drugs, where only 2.8% of molecules are stable as reported in Table 2. This raises questions about the relevance of molecule stability as a performance metric for drug discovery applications, and it could be argued that validity, as measured by RDKit, would be more appropriate in that case. It is less strict and more flexible than molecule stability, accepting less conventional structures such as carbon monoxyde (CO).

Table 2: Metrics on QM9 and GEOM-Drugs

Metric	QM9	GEOM-Drugs
Atom Stability	99.0%	86.5%
Molecule Stability	95.2%	**2.8%**

4.4 Scalability

One notable limit of EDM is its scalability with respect to the maximal number of atoms in the molecules of the training dataset. This is due to the fact that the latent states given as input to the EGNNs are modeled as fully-connected graphs, which requires message passing between all the nodes of the graph. Thus, the computational cost of computing the output of an EGNN is quadratic in the number of atoms in the molecule.
One potential solution is to limit the size of the molecules by handling hydrogens implicitly. This drastically reduces the maximal number of nodes in the graphs on both QM9 and GEOM-Drugs, as shown in Table 1. However, this simplification might limit the model's ability to handle applications where explicit hydrogen interactions, such as hydrogen bonding or protonation states, are significant.
Another possible solution would be to limit message-passing operations by avoiding fully-connected latent graphs (e.g., restricting messages to bonded atoms). But this would be impractical, as denoising latent states requires capturing relationships between distant atoms.
This scaling issue leads to long training times for EDMs, often spanning several days for the models described in the paper. This made us wonder whether scaled-down models could maintain great performance on stability metrics. To explore this, we decided to train smaller versions of EDM and GDM on QM9 with fewer iterations. The number of layers and hidden dimension used are provided in Appendix D, along with comparisons to the reference EDM. The stability results obtained with our models are summarized in Table 3. While our models are significantly outperformed by the reference EDM from the paper, we were surprised by the ability of EDM to generate valid atoms at such small scales. Notably, our small EDM yields similar results as the E-NF model introduced in [9], which had significantly more parameters . This demonstrates that even with fewer parameters, EDMs outperform E-NFs for molecule generation.

Table 3: Atom and molecule stability of $N = 10^4$ samples generated by our small GDM and EDM

	Atom stab. (%)	Mol. stab. (%)
Small GDM	46.5%	0.0%
Small EDM	82.3%	4.8%

4.5 The Impact of Equivariance

Our trained EDM and GDM gave us a more intuitive feeling of how the equivariance constraint makes EDMs learn the denoising process better than GDMs. While the equivariance property is justified by chemistry arguments, it is not clear how imposing it to the model architecture would improve performance. Indeed, the EGNN architecture could be less expressive than its non-equivariant counterpart, since the operations allowed on 3D coordinates are limited. To better visualize what happens, we show in Figure 12 examples of the denoising processes learned by our EDM and GDM at early training epochs.

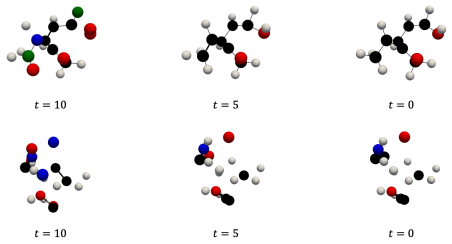


Figure 12: Examples of denoising at selected late time steps of our small EDM (top) and GDM (bottom) after 50 epochs

From this Figure, it seems that the restriction imposed by (9) on the evolution of z_t^x during the denoising process helps the EDM find chemically valid structures more easily. Atoms which are not connected by bonds seem to be "pulled" away from each other, which is not the case for our GDM. This justifies our interpretation of Equation (9) as the operation of a "force" on an atom.
While this experiment highlights the benefits of imposing equivariance for a low number of iterations on a small dataset, equivariant models may not perform as well when trained longer on larger datasets. Equation (9) imposes a strong restriction on the way z_t^x evolves during the denoising process, which helps the model make correct predictions when it has been exposed to a low number of examples. But it may be detrimental to cases where the model has more training examples.

Conclusion

This project analyzed and extended the framework introduced by Hoogeboom et al. in their paper "E(3) Equivariant Diffusion Models

(EDMs) for 3D molecule generation". EDMs demonstrated state-of-the-art performance in generating stable and diverse molecules, significantly outperforming previous methods on both QM9 and GEOM-Drugs.

Our experiments revealed that the model performed surprisingly well even for large molecules in terms of stability and novelty. This is a promising outcome for practical applications in drug discovery. However, the generated molecules occasionally display chemically infeasible structures, such as small cycles or stressed configurations, which are less frequent in the training dataset. These issues arise from the coordinate diffusion process, which does not fully account for chemical bonding properties or steric constraints.

To address scalability challenges, we experimented with reducing the number of equivariant layers. Even with fewer parameters, the equivariant model outperformed its non-equivariant counterpart in atom and molecule stability. This highlights the importance and relevance of equivariance in the design of generative models.

In conclusion, while EDMs represent a major advancement in 3D molecule generation, challenges remain in scaling the method to larger molecules and datasets, as well as ensuring the practical validity of generated structures. Future work should focus on refining the generative process to better incorporate chemical properties, while also addressing the computational demands of handling larger and more complex molecular systems.

References

[1] Simon Axelrod and Rafael Gómez-Bombarelli. 2022. Geom, energy-annotated molecular conformations for property prediction and molecular generation. *Scientific Data*, 9, 1, 185. ISBN: 2052-4463. DOI: 10.1038/s41597-022-01288-4.

[2] Niklas W. A. Gebauer, Michael Gastegger, and Kristof T. Schütt. 2020. Symmetry-adapted generation of 3d point sets for the targeted discovery of molecules. (2020). https://arxiv.org/abs/1906.00957 arXiv: 1906.00957 [stat.ML].

[3] Jonathan Ho, Ajay Jain, and Pieter Abbeel. 2020. Denoising diffusion probabilistic models. (2020). https://arxiv.org/abs/2006.11239 arXiv: 2006.11239 [cs.LG].

[4] Emiel Hoogeboom, Victor Garcia Satorras, Clément Vignac, and Max Welling. 2022. Equivariant diffusion for molecule generation in 3d. (2022). https://arxiv.org/abs/2203.17003 arXiv: 2203.17003 [cs.LG].

[5] Greg Landrum. 2013. RDKit: Open-source cheminformatics. *Journal of Cheminformatics*, 5, 33. DOI: 10.1186/1758-2946-5-33.

[6] Joshua Mitton, Hans M. Senn, Klaas Wynne, and Roderick Murray-Smith. 2021. A graph VAE and graph transformer approach to generating molecular graphs. *CoRR*, abs/2104.04345. https://arxiv.org/abs/2104.04345 arXiv: 2104.04345.

[7] Xingang Peng, Jiaqi Guan, Qiang Liu, and Jianzhu Ma. 2023. Moldiff: addressing the atom-bond inconsistency problem in 3d molecule diffusion generation. (2023). https://arxiv.org/abs/2305.07508 arXiv: 2305.07508 [q-bio.BM].

[8] Raghunathan Ramakrishnan, Pavlo Dral, Matthias Rupp, and Anatole von Lilienfeld. 2014. Quantum chemistry structures and properties of 134 kilo molecules. *Scientific Data*, 1, (Aug. 2014). DOI: 10.1038/sdata.2014.22.

[9] Victor Garcia Satorras, Emiel Hoogeboom, Fabian B. Fuchs, Ingmar Posner, and Max Welling. 2022. E(n) equivariant normalizing flows. (2022). https://arxiv.org/abs/2105.09016 arXiv: 2105.09016 [cs.LG].

[10] Victor Garcia Satorras, Emiel Hoogeboom, and Max Welling. 2022. E(n) equivariant graph neural networks. (2022). https://arxiv.org/abs/2102.09844 arXiv: 2102.09844 [cs.LG].

A Precisions on Diffusion Models

We provide additional information on the diffusion model from 2.1.1.

Diffusion Process: A common schedule for the parameters α_t and σ_t is the variance preserving process, which links them as follows: $\alpha_t = \sqrt{1-\sigma_t^2}$. Usually, α_t is defined as a function that gradually decreases from $\alpha_0 \approx 1$ to $\alpha_T \approx 0$ throughout the noising process.

Regarding the transition probabilities between two latent states z_s and z_t, for $s < t$, the analytical formulas are given by :

$$q(z_t|z_s) = \mathcal{N}(z_t|\alpha_{t|s}z_s, \sigma^2_{t|s}I)$$

with $\alpha_{t|s} = \alpha_t/\alpha_s$ and $\sigma^2_{t|s} = \sigma^2_t/\sigma^2_s$.

Denoising Process: The mean $\mu_{t\to s}$ and standard deviation $\sigma_{t|s}$ of the true denoising process between latent states z_s and z_t are :

$$\mu_{t\to s}(\mathbf{u}) = \frac{\alpha_{t|s}\sigma^2_s}{\sigma^2_t}z_t + \frac{\alpha_s\sigma^2_{t|s}}{\sigma^2_t}\mathbf{u} \quad \text{and} \quad \sigma_{t\to s} = \frac{\sigma_{t|s}\sigma_s}{\sigma_t}$$

B Variational Lower Bound

The evidence lower bound (ELBO) is a variational approach to optimizing the log-likelihood of the training dataset given the model.

$$\log p([x,h]) \geqslant \mathcal{L}_T + \sum_{t=1}^{T}\mathcal{L}_t + \mathcal{L}_0 \tag{10}$$

This formulation divides the loss into three components:

- $\mathcal{L}_T$ measures the KL divergence between the prior $p(z_T)$ and the forward process posterior $q(z_T|[x,h])$. In practice, it is close to zero when the noising schedule is defined such that $\alpha_T \approx 0$.
- $\mathcal{L}_t$ compares $q(z_{t-1}|z_t,[x,h])$ (a tractable forward process posterior) with the reverse process model $p(z_{t-1}|z_t)$.
- $\mathcal{L}_0$ is a reconstruction term for $[x,h]$ given z_0. In practice, if $\alpha_0 \approx 1$ and x is discrete, then $\mathcal{L}_0$ is close to zero

The optimization thus focuses on $\mathcal{L}_t$. In our formulation, the network predicts $\hat{\epsilon} = \phi(z_t,t)$, from which we can compute $[\hat{x},\hat{h}]$, by $[\hat{x},\hat{h}] = (1/\alpha_t)z_t - (\sigma_t/\alpha_t)\hat{\epsilon}$. Then, the bound $\mathcal{L}_t$ becomes:

$$\mathcal{L}_t = \mathbb{E}_{\epsilon\sim\mathcal{N}(0,\mathbb{I})}\left[\frac{1}{2}\left(1-\mathrm{SNR}(t-1)/\mathrm{SNR}(t)\right)\parallel\epsilon-\hat{\epsilon}\parallel^2\right]. \tag{11}$$

In practice, the factors $1-\frac{\mathrm{SNR}(t-1)}{\mathrm{SNR}(t)}$ are set to 1 during training.

C Structure of Equivariant Graph Neural Networks

We provide schematic representations of the operations performed by the Equivariant Graph Convolutional Layers used in the EGNNs. These are shown in Figures 13, 14 and 15.

D Details on our Smaller EDM and GDM

We give some details regarding the experiment presented in 4.4. Our smaller EGNN and GNN are made of 2 layers with 64 hidden features, and were trained for 500 epochs on QM9 using a batch size of 64 (which represents about 780000 iterations). In comparison, the EDM trained on QM9 provided by the authors had an EGNN with 9 layers of 256 features each, and the E-NF introduced in [9] had 6 layers with 64 hidden features. For our small models, the diffusion process is of length $T = 100$, while the EDM in [4] has $T = 1000$.

The training of our models required around 6 hours on a single NVIDIA RTX A2000 GPU. In comparison, the authors report that training their EDM on QM9 with explicit hydrogens for 1.7 million iterations took them 7 days, while training it for 1.2 million iterations on GEOM-Drugs took 5.5 days.

We tested our models every 10 training epochs, and saved the one with the best validation loss.

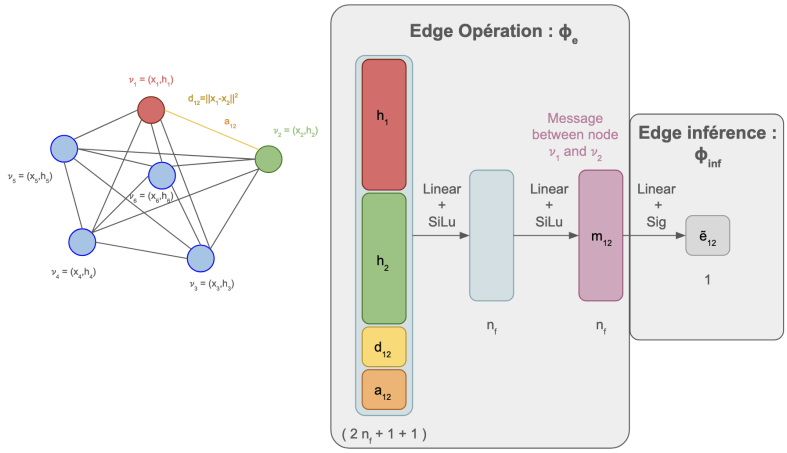


Figure 13: Representation of the EGCL's edge update operation

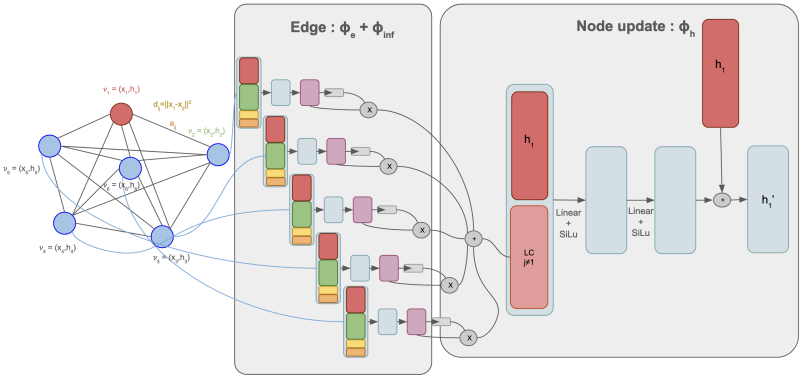


Figure 14: Representation of the EGCL's node update operation

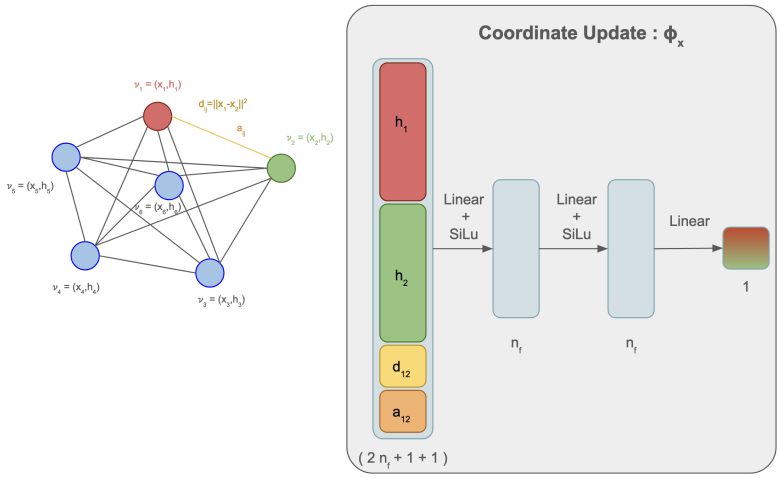


Figure 15: Representation of the EGCL's coordinate update operation