Summary

In this project, we work within the field of de novo drug design by applying categorical and continuous diffusion models on graphs to generate new molecules. The use of diffusion models for de novo drug design is an active field of research seeing rapid publications improving on the previous ones. In recent research, the focus has been on applying different representations of molecules, using different neural networks, and tweaking the noise schedule to improve the generated molecules. Additionally, the research has moved to focus on generation of 3D conformations of the molecules as well. In this project, we investigate the use of both continuous and categorical diffusion for generating valid, unique and novel molecules. Moreover, we investigate how the use of molecular internal coordinates rather than Cartesian coordinates for representing the 3D conformation of a molecule affects the generation of new molecules.

For the categorical diffusion, we follow the approach proposed by Hoogeboom, Nielsen, et al., 2021 that is yet to be applied for de novo drug design using a graph representation. Additionally, we propose a novel spatial graph representation that captures the molecular internal coordinates of a molecule, which we employ in a diffusion model for generation of new molecules in 3D-space. We evaluate the generated molecules based on well known metrics within the field of de novo drug design: validity, uniqueness and novelty. As these metrics do not consider the 3D conformation of a generated molecule, we use additional metrics to evaluate the generated 3D conformations: molecular potential energy and root mean square deviation (RMSD) of the coordinates. As our spatial graph representation is novel, we propose an conversion algorithm from our representation into the well known Z-matrix that also captures the molecular internal coordinates but is structured differently. From the Z-matrix, we can compute the Cartesian coordinates, using NeRF (Parsons et al., 2005), that are used for the energy and RMSD evaluation metrics.

Our experiments show that in the domain of de novo drug design and representing molecules as graphs, the choice of noise schedule is not a major contributing factor to the results, as the two noise schedules perform comparably. When generating simple graphs without 3D conformations, our results

show that the categorical diffusion process performs better than the continuous diffusion process when looking at the validity of the generated molecules. However, looking at uniqueness and novelty, the continuous diffusion process shows similar results to the categorical or even outperforming the categorical slightly. Our diffusion models working on simple graphs outperform state of the art models when generating valid, unique and novel molecules.

Using our spatial graph representation, we are unable to, in general, generate energy minimised 3D conformations of molecules resulting in high energies instead. However, our generated 3D conformations are close in 3D-space to energy minimised conformations based on our RMSD scores. Our high energy of the generated 3D conformations reflects some inherent problems in our spatial graph representation and proposed conversion algorithm.

Based on our results, we see that the categorical diffusion approach from Hoogeboom, Nielsen, et al., 2021 is applicable when representing molecules as graphs with our model outperforming state of the art models. Additionally, the use of our spatial graph representation and conversion algorithm shows great potential in generating sensible 3D conformations of molecules. During both experiments, the use of a distribution inherent to the values being diffused yielded better results and shorter training and convergence times.

Molecule Generation using Diffusion Models on Graphs

De Novo Drug Design

cs-24-mi-10-06

AALBORG UNIVERSITY
STUDENT REPORT

Department of Computer Science
Aalborg University
http://www.aau.dk

Title:
Molecule Generation using Diffusion Models on Graphs

Theme:
De Novo Drug Design

Project Period:
Spring Semester 2024

Project Group:
cs-24-mi-10-06

Participant(s):
Mourits Johannes Jørgensen
Simon Søndergaard Holm

Supervisor(s):
Manfred Jaeger

Page Numbers: 56

Date of Completion:
June 6, 2024

Abstract:

This project introduces how diffusion models coupled with a graph representation of molecules can be used within the field of de novo drug design to generate valid, novel and unique molecules. The project covers how a graph, an inherently discrete data type, can be modelled using both continuous and categorical distributions during the diffusion processes. Additionally, this project introduces how molecules can be generated in 3D-space by introducing a novel spatial graph representation. The representation extends a graph with triplet angles and dihedral angles to capture the 3D conformation of a molecule. The application of categorical diffusion outperforms the continuous version and both achieve state of the art performance. The generated 3D conformations of the molecules using the novel representation show promise in sampling new and diverse conformations. Based on our experiment results, we conclude that using categorical diffusion for discrete values improves the results, and the use of molecular internal coordinates has great potential for generating sensible 3D conformations.

Acknowledgements

This project is the result of a collaboration with AstraZeneca in Gothenburg, Sweden. We would like to express our gratitude to Alessandro Tibo, who was our external supervisor at AstraZeneca during this project, for always providing great support and guidance throughout this project. We would also like to thank our supervisor Manfred Jaeger for his help and review of our work. Finally, we would like to thank AstraZeneca for hosting us with a special thanks to the people at the Molecular AI department, which we were a part of during our stay, for showing interest in our project and sharing their insights and perspectives on our project.

Contents

Chapter 1

Introduction

In recent years, both diffusion models and de novo drug design have seen a lot of attention from the research community. The paper from Ho, Jain, and Abbeel, 2020 showed promising results in the field of image generation using diffusion models. Since, multiple papers regarding diffusion models have been published (Hoogeboom, Nielsen, et al., 2021; Nichol and Dhariwal, 2021). Diffusion models used for de novo drug design have also seen a spike in interest and is an active area of research (Vignac et al., 2023; Guan, Qian, et al., 2023; Guan, Zhou, et al., 2023).

Previous work within the field of de novo drug design using diffusion models has focused on representing the molecules as graphs. Ho, Jain, and Abbeel, 2020 worked on images where each pixel is often represented using continuous values, which means during diffusion continuous noise is applied. However, molecules are inherently discrete, as the atoms and bonds in a molecule belong to one of many categories, which is why papers (Vignac et al., 2023; Guan, Qian, et al., 2023; Guan, Zhou, et al., 2023) focus on how molecules can be represented using discrete values and applying categorical noise during the diffusion process, as it is assumed that this is the best approach. In this project, we investigate how representing a molecule as a graph using continuous values and adding continuous noise compares to using discrete values and adding categorical noise.

Guan, Qian, et al., 2023 and Hoogeboom, Satorras, et al., 2022 investigate how Cartesian coordinates can be used to represent a molecule in 3D-space. For a graph representation of a molecule, the Cartesian coordinates can be implemented as node features. However, when using the Cartesian coordinates representation of a molecule, the model and the loss have to be equivariant to E(3) transformations to preserve the geometric symmetries of the molecule

(Hoogeboom, Satorras, et al., 2022). Instead of using Cartesian coordinates to represent a molecule in 3D-space, a molecule can also be represented using internal coordinates. Representing a molecule using internal coordinates does not have the problem of needing a model and a loss that are equivariant to rotation and translation. Therefore, we also investigate the use of a spatial graph representation for the internal coordinates of a molecule to generate new molecules in 3D-space using diffusion.

In the following, we cover some preliminaries in relation to this project. First, we cover some metrics that we use for evaluation of the generated molecules. Second, we cover how molecules can be represented in 3D-space in different ways with a focus on the internal coordinates representation of a molecule. Third, we cover how we represent molecules in 3D-space as graphs with added features needed for the internal coordinates of the molecule.

1.1 Evaluation Metrics

To evaluate the generated molecules, we define multiple metrics that cover different aspects of the molecules and how they relate to the training dataset. We define the molecules used during training as the multiset $\mathcal{M}_t$, and the molecules generated by the diffusion model as the multiset $\mathcal{M}_g$. In the following we use $[]$ when describing a multiset and the standard $\{\}$ notation for sets. We define an operation $\mathbf{Set} : [\,] \mapsto \{\}$ that converts a multiset into a set by keeping only one of each element in the multiset, e.g. $\mathbf{Set}([1, 2, 2, 3, 4, 5, 5]) = \{1, 2, 3, 4, 5\}$. Furthermore, we define the cardinality of a multiset $|[\,]|$ as the sum of multiplicities in the multiset, e.g. $|[1, 2, 2, 3, 4, 5, 5]| = 7$.

1.1.1 Validity

We use RDKit[1] to determine if a generated molecule is valid. For a molecule to be valid according to RDKit, the molecule must pass a series of sanitisation[2] checks. Two of the most important sanitisation checks is a valence check and a check on the Kekule form of aromatic rings. When checking valency, the intu-

[1] `https://www.rdkit.org/`

[2] `https://www.rdkit.org/docs/RDKit_Book.html#molecular-sanitization`

ition is that each atom must not have a too high valency, which occurs when it binds with more atoms than chemically possible. The kekulization of the aromatic rings will fail if a molecule contains rings but the bonds are not marked as aromatic or there are found chemically impossible bonds outside an aromatic ring. The multiset of valid molecules is defined as $\mathcal{M}_v = [x \in \mathcal{M}_g | x \text{ is valid}]$. The validity metric is defined as $V = \frac{|\mathcal{M}_v|}{|\mathcal{M}_g|}$.

1.1.2 Uniqueness

For de novo drug design, discovering new molecules is one of the most important aspects. Intuitively, to make the discovery process efficient, the generated molecules $\mathcal{M}_g$ should be diverse and unique such that each generation process discovers as many new molecules as possible. Depending on the generation process, the uniqueness of the generated molecules is not guaranteed in which case the uniqueness metric is often used to report the diversity of the generated molecules. We define the unique molecules, based on the multiset of valid molecules, as the set $\mathcal{M}_u = \mathbf{Set}(\mathcal{M}_v)$. We introduce two types of uniqueness defined as $U_g = \frac{|\mathcal{M}_u|}{|\mathcal{M}_g|}$ and $U_v = \frac{|\mathcal{M}_u|}{|\mathcal{M}_v|}$. U_g is the unique and valid molecules w.r.t. $\mathcal{M}_g$, while U_v is the unique and valid molecules w.r.t. $\mathcal{M}_v$. As it can be seen from the definition of U_g and U_v, it is expected that the uniqueness metrics will drop as the number of generated molecules increases.

In practice checking whether a molecule is unique is often done using a canonical SMILES representation, such that if two or more molecules generate the same SMILES string only one of them is considered a unique sample. Uniqueness is also a metric that can be used to check mode collapse of a trained generative model. Here, the generative model is said to have encountered a mode collapse if it keeps generating the same small subset of molecules, resulting in a very low uniqueness score.

1.1.3 Novelty

As mentioned above, discovering new molecules in de novo drug design is important. Novelty is a metric that measures how many of the generated molecules $\mathcal{M}_g$ are actually unseen and new. Here, a molecule is considered

new if it is not found in the molecules $\mathcal{M}_t$ that are used during training. Additionally, in de novo drug design you want the novel molecules to be both valid and unique and as such the novelty is often defined over the the set of valid and unique molecules rather than just the set of valid molecules. Following this, we define novel molecules as the set $\mathcal{M}_n = \{x \in \mathcal{M}_u | \, x \notin \mathcal{M}_t\}$. Furthermore, we introduce three types of novelty to report: $N_g = \frac{|\mathcal{M}_n|}{|\mathcal{M}_g|}$, $N_v = \frac{|\mathcal{M}_n|}{|\mathcal{M}_v|}$ and $N_u = \frac{|\mathcal{M}_n|}{|\mathcal{M}_u|}$. All three types of novelty is a measure of the number of valid, unique and novel molecules w.r.t. to a certain set or multiset, i.e. N_g w.r.t. $\mathcal{M}_g$, N_v w.r.t. $\mathcal{M}_v$, and N_u w.r.t. $\mathcal{M}_u$. As with uniqueness, it is expected that the novelty metrics will drop as the number of generated molecules increases.

In practice, determining whether a molecule is novel is often done using a canonical SMILES representation, such that, if the SMILES string of a molecule in $\mathcal{M}_g$ is the same as a SMILES string of a molecule in $\mathcal{M}_t$ then the molecule is not novel. Additionally, novelty can be used to detect mode collapse of a trained generative model, i.e. with a low N_u score, the model does not generate unseen molecules.

1.1.4 Root Mean Square Deviation

Root Mean Square Deviation (RMSD) is a distance based similarity measure between two molecules. In this project, RMSD is calculated between a generated molecule $\mathcal{M}$ and a reference molecule $\mathcal{R}$ with 3D Cartesian coordinates and with both molecules consisting of N atoms. The RMSD between two molecules can be calculated as (Kufareva and Abagyan, 2012):

$$RMSD = \sqrt{\frac{1}{N}\sum_{i=1}^{N} d_i^2}$$

where d_i is the euclidean distance between the i^{th} pair of equivalent atoms from $\mathcal{M}$ and $\mathcal{R}$. Determining which pairs of atoms are equivalent can be hard. However, in this project, we only use the RMSD metric in contexts where we compare a 3D conformation of a molecule with its energy minimised conformation. Therefore, we always know which pairs of atoms are equivalent, as the molecules we compare are the same, just with different 3D coordinates. In this project, the RMSD score is measured in angstrom.

1.1.5 Energy of a Molecule

The potential energy of a specific 3D conformation of a molecule, measured in kcal/mol, can be used to describe the stability of the 3D conformation when compared with all possible 3D conformations of the same molecule. All possible 3D conformations of a molecule characterise the energy surface of the molecule, where a specific 3D conformation of the molecule is a single point on its energy surface. When the potential energy, hereafter referred to as just energy in this project, is at a local minima on the energy surface the 3D conformation of the molecule is considered stable. With this in mind, the energy of a molecule can be seen as a function of its geometry in 3D-space that considers distances between atoms, bond types, triplet angles, dihedral angles and charges. Different definitions of the energy function exist, and they employ different penalties for the geometry of a molecule. For example, 2 carbon atoms in a single bond usually have a distance of 1.54 angstroms (Leach, 2001, p. 3). If the distance is different in the 3D conformation of the molecule, it is penalised with a larger energy. For any molecule with a specific 3D conformation, its energy can be minimised by changing the 3D conformation (Leach, 2001, p. 4-5, 253).

When comparing the energy states of two molecules, we take inspiration from the approach by Tong and Zhao, 2021, where they calculate the Ligand conformational strain energy (LCSE) between two molecules. In our case, we want to calculate the strain energy between a generated 3D conformation of a mole and its energy minimised counterpart. Following Tong and Zhao, 2021, this gives us the following formula:

$$E_{LCSE} = E_{generated} - E_{minimised}$$

However, this formula is not normalised and to be able to compare the results, where molecules of different sizes are used, we normalise the strain energy based on the number of atoms in the molecule, N:

$$E_{strain} = \frac{E_{generated} - E_{minimised}}{N}$$

1.2 Molecular Internal Coordinates

In de novo drug design it can be desirable that the generated molecules have a certain conformation in 3D-space, s.t. they bind easier with certain protein target binding sites. The most prominent representation is the use of Cartesian coordinates specified for each of the atoms. The use of Cartesian coordinates introduces a problem in the model and loss when it comes to translation, rotation and reflection in 3D-space, E(3) transformations, as we want to preserve the geometric symmetries of the molecules (Guan, Qian, et al., 2023; Hoogeboom, Satorras, et al., 2022). These problems are not present when using internal coordinates of the molecule to represent its conformation in 3D-space.

The internal coordinates of a molecule, called the molecular internal coordinates, are represented using the bond length between atoms, the angles produced by sets of 3 connected atoms and dihedral angles between 2 sets of 3 connected atoms, where 2 atoms are in common. The dihedral angle is the angle between the 2 planes passing through the 2 sets of 3 connected atoms in 3D-space. As such, the dihedral angle is also called the torsion angle, as it represent how we can get the position of one atom from a second atom based on a torsion in 3D-space. For convenience, the bond lengths are measured in angstroms. The angles produced by the sets of 3 atoms range from 0 to 180 degrees, $(0, 180)$, and the dihedral angles range from -180 to 180 degrees, $[-180, 180]$ (Leach, 2001, p. 2-4). A simple example of the molecular internal coordinates for HOOH can be seen in Figure 1.1.

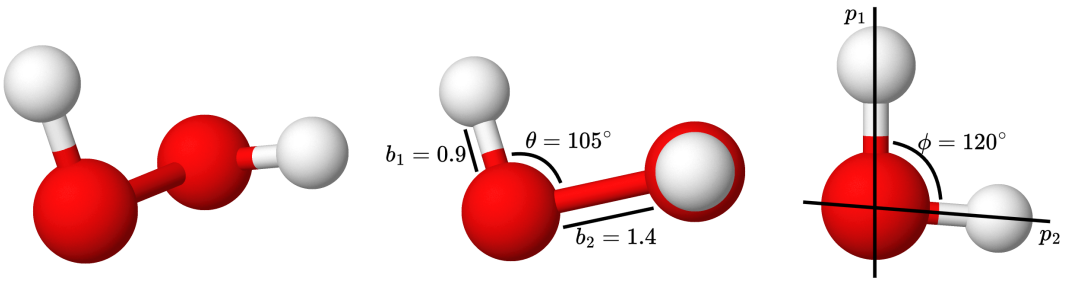

Figure 1.1: HOOH in 3D-space from different viewpoints with illustrations of the bond lengths, b, angles between sets of 3 connected atoms, θ, and dihedral angles, ϕ.

In Figure 1.1, HOOH is placed in 3D-space and illustrated from three different viewpoints. The first viewpoint gives a general view of how the atoms are placed in relation to each other. The second viewpoint illustrates the bond lengths of 0.9 and 1.4 between the atom pairs HO and OO, and the angle of 105° between the 3 connected atoms HOO. The third viewpoint illustrates how the plane, p_1, formed by the atoms HOO and the plane, p_2, formed by the atoms OOH has a dihedral angle of 120°.

When it comes to dihedral angles, there exists 2 different kinds in the molecular internal coordinate representation. The difference lies in whether the non-common atoms, in the 2 sets of 3 connected atoms, are connected to the same or different atoms. These 2 variants of dihedral angles are seen in Figure 1.2.

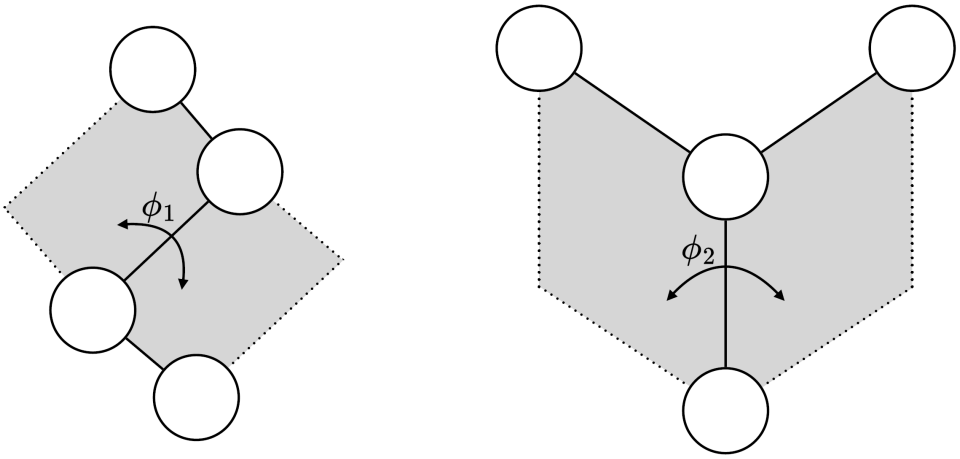

Figure 1.2: The two different kinds of dihedral angles.

The first kind of dihedral angle, ϕ_1 in Figure 1.2, is formed by 4 sequentially connected atoms, as the non-common atoms are connected to different common atoms. For the second kind of dihedral angle, ϕ_2 in Figure 1.2, the non-common atoms are connected to the same common atom to form a Y-like shape.

1.2.1 Representing the Internal Coordinates in a Z-Matrix

A common way of representing the molecular internal coordinates is in a Z-matrix (Leach, 2001, p. 2-4). The Z-matrix uses an ordering between atoms identified by the rows in the matrix, and the following structure for each row

in the matrix: *label, atom1, bond length, atom2, angle, atom3, dihedral angle.* The label is an indication of the atom either by its atomic symbol or atomic number, the atom numbers, *atom1, atom2, atom3*, reference specific rows in the matrix and the bond length, angle and dihedral angle are decimal numbers. Building the Z-matrix is done by adding one atom/row at a time, which results in the first 3 rows being special as they do not contain all the information listed above. This is easily seen by studying an example of the Z-matrix for hydrogen peroxide (HOOH) (Roskilde Universitet, 2009):

```
1: H
2: O 1 0.9
3: O 2 1.4 1 105.0
4: H 3 0.9 2 105.0 1 120.0
```

The above Z-matrix captures the molecular internal coordinates illustrated in Figure 1.1. In the Z-matrix, the anchor atom H is added in the first row. The second row specifies that O is connected to atom 1, H, with a distance of 0.9 between them. In the third row, another O is added, and it is specified that it is connected to atom 2, O, with a distance of 1.4 between them. Moreover, the third row specifies an angle of 105.0 between atoms 1, 2 and 3. The fourth row adds the last H, specifies a distance of 0.9 to atom 3, O, an angle of 105.0 between atoms 2, 3, 4 and a dihedral angle of 120.0 between the planes generated by atoms 1, 2, 3 and 2, 3, 4.

1.2.2 Converting Between Cartesian and Z-Matrix

When converting from Cartesian coordinates to molecular internal coordinates, given Cartesian coordinates for 2 atoms, a and b, the bond length is the norm of the vector $\vec{ab}$ going from a to b, $\|\vec{ab}\|$. Given 3 sequentially connected atoms, a, b, and c, the triplet angle needed for the molecular internal coordinates is the angle between the vectors $\vec{ba}$ and $\vec{bc}$. Given 4 connected atoms a, b, c, and d, the dihedral angle needed for the molecular internal coordinates is the angle between 2 planes. The first plane passing through the atoms a, b, c and the second plane passing through the atoms b, c, d. In summary, given 4 connected atoms a, b, c, and d, the bond length, l, and the triplet angle, θ,

can be computed as:

$$l = \|b - a\| \tag{1.1}$$

$$\theta = \arccos\left(\frac{(a-b)\cdot(c-b)}{\|(a-b)\|\cdot\|(c-b)\|}\right) \tag{1.2}$$

The dihedral angle, ϕ, can be computed as:

$$\begin{aligned} \phi &= \text{arctan2}(\omega, \psi) \\ \text{where } \omega &= \frac{(\mathbf{v}_0 \times \mathbf{v}_1) \times (\mathbf{v}_1 \times \mathbf{v}_2)}{\|(\mathbf{v}_0 \times \mathbf{v}_1)\| \cdot \|(\mathbf{v}_1 \times \mathbf{v}_2)\|} \cdot \frac{\mathbf{v}_1}{\|\mathbf{v}_1\|}, \\ \text{and } \psi &= \frac{(\mathbf{v}_0 \times \mathbf{v}_1) \cdot (\mathbf{v}_1 \times \mathbf{v}_2)}{\|\mathbf{v}_0 \times \mathbf{v}_1\| \cdot \|\mathbf{v}_1 \times \mathbf{v}_2\|} \end{aligned} \tag{1.3}$$

Here, we use 3 vectors defined by the Cartesian coordinates of the 4 atoms. If the dihedral angle is similar to the first kind, ϕ_1 in Figure 1.2, the vectors are $\mathbf{v}_0 = b - a$, $\mathbf{v}_1 = c - b$, and $\mathbf{v}_2 = d - c$. If the dihedral angle has the Y-shape of the second kind, ϕ_2 in Figure 1.2, $\mathbf{v}_0$ and $\mathbf{v}_1$ are the same but for $\mathbf{v}_2$ we use $\mathbf{v}_2 = d - b$.

One prominent way to convert back from molecular internal coordinates in a Z-matrix to Cartesian coordinates is using the Natural Extension Reference Frame (NeRF) method proposed by Parsons et al., 2005. The NeRF method considers 4 cases; the first 3 cases being the first 3 rows in the Z-matrix and the 4^{th} case being every row after the first 3 in the Z-matrix.

The first case places the atom in the first row of the Z-matrix in the origin of the 3D-space, $a_1 = (0, 0, 0)$. The second case places the second atom from the Z-matrix by translating a_1 by the bond length on the x-axis, $a_2 = (l, 0, 0)$. The third case uses the bond length and triplet angle to place the third atom from the Z-matrix on the xy-plane as follows: $a_3 = a_2 + (l \cdot \cos(\pi - \theta), l \cdot \sin(\theta), 0)$, where $+$ is element-wise addition of the Cartesian coordinates. For the fourth case the spherical representation, Q, of the fourth atom is first calculated as (Parsons et al., 2005):

$$Q = (l \cdot \cos(\pi - \theta), \quad l \cdot \cos(\phi) \cdot \sin(\theta), \quad l \cdot \sin(\pi + \phi) \cdot \sin(\theta))$$

Next, a transformation matrix, R, is calculated based on the vectors defined by the previous 3 atoms, $\mathbf{n}_1 = a_2 - a_1$ and $\mathbf{n}_2 = a_3 - a_2$, as (Parsons et al.,

2005):

$$R = [r_1, \quad r_2 \times r_1, \quad r_2]$$
$$\text{where } r_1 = \frac{\mathbf{n}_2}{\|\mathbf{n}_2\|} \text{ and } r_2 = \frac{\mathbf{n}_1 \times \mathbf{n}_2}{\|\mathbf{n}_1 \times \mathbf{n}_2\|}$$

Using Q and R, the Cartesian coordinates for the fourth atom from the Z-matrix is given as (Parsons et al., 2005):

$$a_4 = R \times Q + a_3 \tag{1.4}$$

1.3 Graph Representation with Angles

A molecule with N atoms can be represented as a simple graph $\mathcal{M} = (V, E)$, where $V \in \mathbb{R}^N$ is the set of vertices/atoms, and $E \in \mathbb{R}^{N \times N}$ is the set of edges/bonds. To extend this graph representation to molecules with internal coordinates, we define the following injective functions that maps from ordered sets of atoms to angle indices:

$$\text{IndexA} : \Gamma_3 \mapsto [1, \ldots, N_a]$$
$$\text{IndexD} : \Gamma_4 \mapsto [1, \ldots, N_d]$$

where N_a is the maximum number of triplet angles for any given atom in the molecule, N_d is the maximum number of dihedral angles for any given atom in the molecule, and:

$$\begin{aligned}\Gamma_3 &= \{(v_1, v_2, v_3) \mid v_2 \in V, (v_1, v_3) \in \text{Comb}(N(v_2), 2)\} \\ \Gamma_4 &= \{(v_1, v_2, v_3, v_4) \mid (v_1, v_2), (v_2, v_3), (v_x, v_4) \in E \\ &\qquad \wedge v_1 \neq v_3 \\ &\qquad \wedge (v_x = v_2 \vee v_x = v_3)\}\end{aligned}$$

where $N(v) = \{x \mid (v, x) \in E\}$ is the neighbourhood of v and $\text{Comb}(N(v), 2)$ is all combinations of unordered pairs in the neighbourhood of v.

With these functions, we represent a molecule with internal coordinates as a spatial graph $\mathcal{M} = (V, E, A, D)$. Here, $A \in \mathbb{R}^{N \times N_a}$ is the set of angles (triplet angles) formed by sequentially connected triplets of vertices/atoms, and $D \in \mathbb{R}^{N \times N_d}$ is the set of dihedral angles formed by 2 sets of 3 sequentially connected

vertices/atoms with 2 common vertices/atoms between the 2 sets. We place all the triplet angles for triplets $(v_x, v_2, v_y) \in \Gamma_3$ in the row in A corresponding to v_2 and the column based on the ordering by IndexA. Similarly, we place all the dihedral angles for 4-tuples $(v_1, v_x, v_y, v_z) \in \Gamma_4$ in the row in D corresponding to v_1 and the column based on the ordering by IndexD.

When fixing the order of the atoms, as in a Z-matrix, we need $N - 1$ bond lengths, $N - 2$ angles and $N - 3$ dihedral angles to represent the internal coordinates of the molecule. However, with our spatial graph representation, we do not want to impose an order of the atoms, as such we set N_a and N_d to be determined by the dataset. This allows us to capture all angles for all orderings of atoms in valid molecules. It should be noted that N_a and N_d will not explode in size in the domain of molecules, as we represent molecules as simple graphs with a limited amount of connections. For example, using the QM9 dataset, we have $N_a = 10$ and $N_d = 76$.

1.3.1 Converting from Spatial Graph to Z-matrix

When converting from a spatial graph, $\mathcal{M} = (V, E, A, D)$ with N nodes, to a Z-matrix, we assume the graph to be connected, i.e. it does not contain any subgraphs that are disconnected. For the conversion, we define the following additional functions:

$$\begin{aligned} \text{IndexV} &: V \mapsto [1, \ldots, N] \\ \text{Bond} &: E \times V \times V \mapsto \mathbb{R} \\ \text{Next} &: \emptyset \mapsto [1, \ldots, N] \times [1, \ldots, N] \end{aligned}$$

where IndexV returns the index of the input atom in V, and Bond returns the bond length between the 2 input atoms based on their atom types and the type of bond connecting the atoms. In practice, Bond is defined as a static lookup table based on the findings of Pyykkö and Atsumi, 2009. Next returns the index of unvisited atom and the discovery time of the parent atom, it is connected to, based on a depth-first traversal of the graph. We will use $\perp$ for the discovery time of the parent atom for the seed atom and whenever we do not need said discovery time in our algorithm. The depth-first traversal that is used internally in Next is based on our set of edges E that is considered an

internal part of the Next function; therefore, no input is given when calling the function. We assume the seed atom of the internal depth-first traversal to be connected to 2 other atoms in sequence such that the seed is an endpoint of a triplet of atoms.

Using these functions the conversion from a spatial graph, $\mathcal{M}$, to a Z-matrix, Z, is performed using Algorithm 1.

Algorithm 1 Spatial Graph to Z-matrix

Input: $\mathcal{M} = (V, E, A, D)$
Output: $Z \in \mathbb{R}^{N \times 7}$

1: **for** $i = 1$ to N **do**
2: **if** $i = 1$ **then**
3: $\mathcal{M}, Z \leftarrow\!\shortmid \textsc{case1}(\mathcal{M}, Z)$ ▷ First case of the Z-matrix
4: **else if** $i = 2$ **then**
5: $\mathcal{M}, Z \leftarrow\!\shortmid \textsc{case2}(\mathcal{M}, Z)$ ▷ Second case of the Z-matrix
6: **else if** $i = 3$ **then**
7: $\mathcal{M}, Z \leftarrow\!\shortmid \textsc{case3}(\mathcal{M}, Z)$ ▷ Third case of the Z-matrix
8: **else**
9: $\mathcal{M}, Z \leftarrow\!\shortmid \textsc{case4}(\mathcal{M}, Z, i)$ ▷ Fourth case of the Z-matrix
10: **end if**
11: **end for**

In Algorithm 1, we build the Z-matrix row by row, by iterating over the number of atoms in the molecule and adding one row to the Z-matrix for each iteration. The 4 cases of the Z-matrix described in Section 1.2.1 are identified by the `if` statements checking the iteration number i. When the first 3 rows are added in the bodies of the `if` statements, the remaining rows are added using logic for the fourth case of the Z-matrix. Pseudocode for each case of the Z-matrix is presented in Algorithm 2, 3, 4, and 5.

Algorithm 2 Spatial Graph to Z-matrix - Case 1

procedure CASE1($\mathcal{M}, Z$)
    $v, \perp = \text{Next}()$
    $Z_1 = (V_v, 0, 0, 0, 0, 0, 0)$
    return $\mathcal{M}, Z$
end procedure

In Algorithm 2, the first atom is inserted into the Z-matrix. As the first atom does not have any references to other atoms, the other entries in this row of the Z-matrix are set to 0.

Algorithm 3 Spatial Graph to Z-matrix - Case 2

procedure CASE2($\mathcal{M}, Z$)
    $v1 \leftarrow\!\shortmid \text{IndexV}(Z_{1,1})$
    $v2, \perp = \text{Next}()$
    $b \leftarrow\!\shortmid \text{Bond}((V_{v1}, V_{v2}), V_{v1}, V_{v2})$
    $Z_2 = (V_{v2}, 1, b, 0, 0, 0, 0)$
    return $\mathcal{M}, Z$
end procedure

In Algorithm 3, the second row in the Z-matrix is inserted. In this case, the bond length between the first inserted atom and a connected atom is retrieved. Next, the second atom, reference atom, and bond length are inserted into the Z-matrix.

Algorithm 4 Spatial Graph to Z-matrix - Case 3

procedure CASE3($\mathcal{M}, Z$)
    $v1, v2 \leftarrow\!\shortmid \text{IndexV}(Z_{1,1}), \text{IndexV}(Z_{2,1})$
    $v3, \perp \leftarrow\!\shortmid \text{Next}()$
    $b, a \leftarrow\!\shortmid \text{Bond}((V_{v2}, V_{v3}), V_{v2}, V_{v3}), \text{IndexA}(V_{v1}, V_{v2}, V_{v3})$
    $Z_3 = (V_{v3}, 2, b, 1, A_{v2,a}, 0, 0)$
    return $\mathcal{M}, Z$
end procedure

In Algorithm 4, the third row in the Z-matrix is inserted. Here, we first get the indices for the 3 connected atoms based on the previous 2 entries in the Z-matrix and the Next function. Using these 3 atoms, we compute the bond length and the angle index. Lastly, a new row in the Z-matrix is inserted using the 3 atoms, bond length, and the angle.

Algorithm 5 Spatial Graph to Z-matrix - Case 4

1: **procedure** CASE4($\mathcal{M}, Z, i$)
2: $v4, j \leftarrow \text{Next}()$
3: **if** $j = 1$ **then**
4: $z1, z2 \leftarrow 3, 2$
5: **else if** $j = 2$ **then**
6: $z1, z2 \leftarrow 3, 1$
7: **else**
8: $z1, z2 \leftarrow Z_{j,4}, Z_{j,2}$
9: **end if**
10: $v1, v2, v3 \leftarrow \text{IndexV}(Z_{z1,1}), \text{IndexV}(Z_{z2,1}), \text{IndexV}(Z_{j,1})$
11: $b \leftarrow \text{Bond}((V_{v3}, V_{v4}), V_{v3}, V_{v4})$
12: $a, d \leftarrow \text{IndexA}(V_{v2}, V_{v3}, V_{v4}), \text{IndexD}(V_{v1}, V_{v2}, V_{v3}, V_{v4})$
13: $Z_i \leftarrow (V_{v4}, j, b, z2, A_{v3,a}, z1, D_{v1,d})$
14: **return** $\mathcal{M}, Z$
15: **end procedure**

Algorithm 5 is used to insert the remaining rows of the Z-matrix. First, an unvisited atom and the discovery time of its parent node is retrieved using the Next function. Using the discovery time of the parent node, the atoms forming a triplet angle and dihedral angle with the unvisited node can be found in the Z-matrix. However, two special cases exist when the discovery time is either 1 or 2, as these entries do not contain both reference atom 1 and 2, which we need for the triplet angle and dihedral angle. The bond length is then calculated, and the indices for the triplet angle and dihedral angle is retrieved. With this information, a new row is inserted in the Z-matrix.

Chapter 2

Diffusion Models

Diffusion models are used as generative machine learning models and can be applied to the task of de novo drug design where the models are used to generate molecules (Vignac et al., 2023; Guan, Qian, et al., 2023; Guan, Zhou, et al., 2023). The intuition behind diffusion models is a process in two parts, a forward process and a reverse process. For the forward process a data sample x is gradually diffused, meaning noise is gradually applied to x. The diffusion is performed over T timesteps, denoting x at step t as x_t, where x_0 is the original un-noised data sample and x_T is complete white noise. When gradually applying noise, the amount of noise added between each step is determined using a variance schedule, often called a noise schedule. For the reverse process, the diffusion model learns to remove the applied noise from a noisy data sample x_t utilising a parameterised neural network. An illustration of the forward process and the reverse process on molecules can be seen in Figure 2.1. During the forward process, the atom types are changed, the type of bonds are changed, and bonds may either be removed or added, as seen in the top row of Figure 2.1. The reverse process tries to undo the changes in the molecule, which most often does not result in the original molecule, as seen in the bottom row of Figure 2.1.

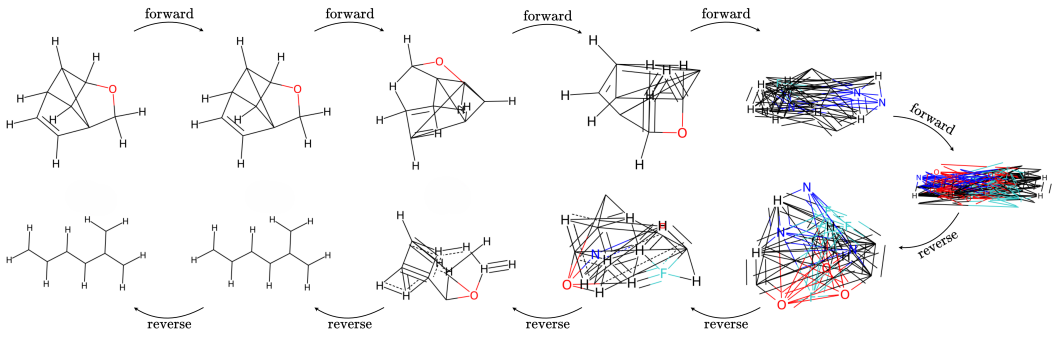

Figure 2.1: Illustration of the purpose of the forward process and the reverse process in a diffusion model.

Diffusion models can be used for generating new data samples, $\hat{x}_0$, when the complete white noise samples, x_T, follow a prior probability distribution that can be sampled from. Generation is then performed by first sampling from the prior distribution and then performing the reverse process for T steps to obtain the new cleaned sample, $\hat{x}_0$. The training objective for a diffusion model is having the parameterised neural network optimised for denoising the data sample, x_t, in the reverse process.

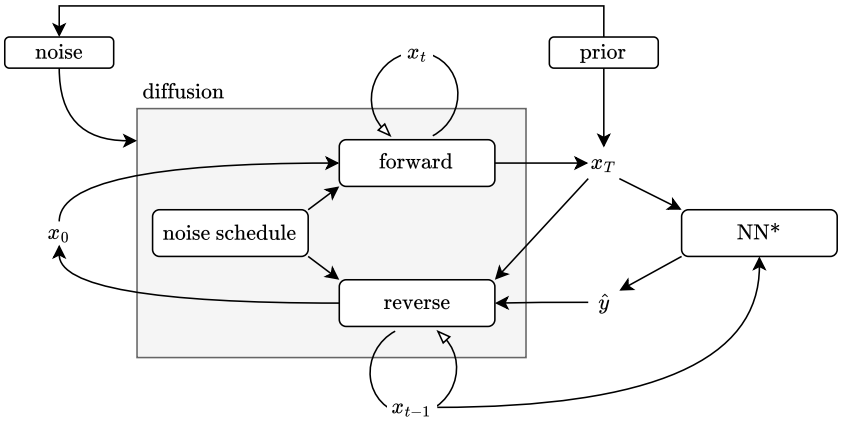

Figure 2.2: Overview of the components in a diffusion model.

Figure 2.2 shows the interaction between the noise schedule, the forward process, the reverse process and the parameterised neural network. The forward component can be captured by the function $f(x_t) \mapsto x_{t+1}$. The function f is utilised iteratively, starting with the initial sample, x_0, and adds one step of noise at a time, depicted by the loop, always working with the current sample x_t and stops when x_T is obtained. The reverse component can be captured by the function $r(x_t, \hat{y}) \mapsto x_{t-1}$. The function r is also utilised iteratively, starting with the complete noisy sample, x_T, and a prediction, $\hat{y}$, to clean the sample one step at a time, depicted by the loop. It always works on the current sample x_t and a prediction $\hat{y}$, based on the current sample, and produces a sample from the previous timestep x_{t-1}, which becomes the new current sample for the next iteration. The reverse process halts when x_0 is obtained. In the function r, $\hat{y}$ is the prediction from the neural network, and it can be a prediction of either x_0, x_{t-1} or the added noise from the forward process. The choice of the prediction depends on the formulation of the reverse process. The neural network is marked with *, as it can be interchanged with any component that

produces $\hat{y}$ in the same format as x_0 and x_t. The figure also shows that the complete noisy sample, x_T, follows a prior distribution that is also imposed as a target/limit distribution on the noise. The noise distribution is imposed on the forward and reverse processes in the diffusion model.

In this project, we focus on using diffusion models on molecules for de novo drug design, where molecules are represented as graphs. A molecule exists in 3D-space and consists of atoms that are connected to each other by different types of bonds. Before introducing how diffusion is applied to a molecule in 3D-space, we first introduce how diffusion is applied on a molecule without spatial information. We represent a molecule without spatial information consisting of N atoms as a simple graph, $\mathcal{M} = (V, E)$, where V and E are the same as introduced in Section 1.3. The processes in the diffusion model work directly on the molecule, $\mathcal{M}$, by applying each process on each element, V and E, separately. With this approach, the goal is to generate molecules that are valid, unique and novel.

In the following, we cover the different aspects and components of diffusion models using continuous and categorical distributions. Section 2.1 covers the noise schedule that is used in the forward process in Section 2.2. The reverse process is then covered in Section 2.3 followed by the loss function for the parameterised neural network in Section 2.4. Lastly, it is covered how to train a diffusion model and how the trained neural network can be used for generation of new molecules in Section 2.5 and Section 2.6, respectively.

For the diffusion processes, we let $\mathbf{v} \in \mathbb{R}^{N \times K_v}$ and $\mathbf{e} \in \mathbb{R}^{N \times N \times K_e}$ be the one-hot encodings of the atom types, V, and bond types, E. For the encodings, we can use the argmax function to get the resulting category. For brevity, we denote encoded molecules as $M = (\mathbf{v}, \mathbf{e})$. We allow the diffusion processes to change the atom types, bond types, and number of edges. Hence, we do not allow the processes to add or delete atoms. To handle the deletion of a bond, we introduce an extra category for the bonds that indicates no bond between two atoms.

2.1 Noise Schedule

The noise schedule controls the amount of noise that is added to an input when transitioning from timestep $t-1$ to t. The noise schedule is independent from the rest of the diffusion model and can be considered as a function $f(t)$ that determines the step size of the noise application. The output of $f(t)$ is denoted as β_t. β_t is a scalar value that describes how much a data sample is changed/noised for each timestep, e.g. when using a Gaussian distribution for adding noise in the diffusion model, β_t is the variance. The noise schedule is used both in the forward process and reverse process of the diffusion model. If the forward process can be computed in closed form, it utilises $\bar{\alpha}_t$ given from α_t and β_t as follows (Ho, Jain, and Abbeel, 2020):

$$\alpha_t = 1 - \beta_t \qquad\qquad \bar{\alpha}_t = \prod_{s=1}^{t} \alpha_s \tag{2.1}$$

where $\bar{\alpha}_t$ represents the noise accumulation on the input sample in an inverse proportionality fashion, i.e. a small $\bar{\alpha}_t$ means a large amount of noise is added to the input sample. Both β_t, α_t and $\bar{\alpha}_t$ are often utilised in the reverse process when denoising the sample based on the output of a parameterised neural network. Several variants and definitions of $f(t)$ have been proposed for diffusion models, e.g. the constant, linear, cosine and sigmoid schedules. Here, we cover some of the more used definitions, namely the linear and cosine.

2.1.1 Linear

In the linear noise schedule the difference between $f(t)$ and $f(t-1)$ is constant. For the linear noise schedule, a start value β_{min} and an end value β_{max} is specified. The noise schedule is then defined as a linear interpolation from $\beta_1 = \beta_{\text{min}}$ to $\beta_T = \beta_{\text{max}}$ using the number of timesteps T (Nichol and Dhariwal, 2021).

2.1.2 Cosine

In the cosine noise schedule, the difference between $f(t)$ and $f(t-1)$ varies depending on t. The cosine noise schedule is specified as follows:

$$f(t) = \cos\left(\frac{t/T + s}{1+s} \cdot \frac{\pi}{2}\right)^2 \qquad \tilde{\alpha}_t = \frac{f(t)}{f(0)} \qquad \beta_t = 1 - \frac{\tilde{\alpha}_t}{\tilde{\alpha}_{t-1}} \tag{2.2}$$

where s is a variable used to prevent β_t from being too small in the early timesteps. In practice, Nichol and Dhariwal, 2021 propose to use $s = 0.008$ and clipping the β_t values to be no larger than 0.999. Hence, computing β_t, α_t and $\bar{\alpha}_t$ is a three step process when using the cosine schedule. Firstly, β_t values are computed following (2.2). Secondly, the β_t values are clipped to a max value, e.g. 0.999. Lastly, the α_t and $\bar{\alpha}_t$ values are computed following (2.1).

2.1.3 Comparison of Noise Schedules

Figure 2.3 shows a comparison of the cosine noise schedule and the linear noise schedule. In the figure, it can be seen that the linear noise schedule has a steeper slope, which means that noise quickly accumulates compared to the cosine noise schedule. The cosine noise schedule has a more even noise accumulation that gives the parameterised neural network more timesteps where the molecule is not complete noise. Additionally, using the linear noise schedule results in close to complete noise when $t/T > 0.7$ as $\bar{\alpha}_t \approx 0$, whereas using the cosine noise schedule at timestep $t/T = 0.7$ we only have $\bar{\alpha}_t \approx 0.2$.

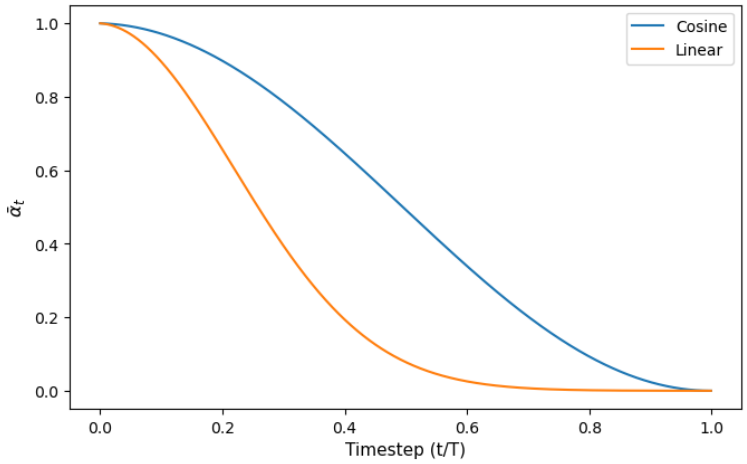

Figure 2.3: Comparison of $\bar{\alpha}_t$ for the cosine noise schedule and the linear noise schedule.

2.2 Forward Process

The forward process depends on the type of noise distribution that is used for the atom types and bond types. In the forward process, the β_t and $\bar{\alpha}_t$ values, as described in Section 2.1, are utilised. In this section, we cover how both continuous and categorical noise distributions can be used in the forward process of a diffusion model.

2.2.1 Continuous Distribution

For the continuous distribution, we use a Gaussian distribution for adding noise in the forward process. Following Ho, Jain, and Abbeel, 2020, we can define one step of the forward process as:

$$q(M_t|M_{t-1}) = \prod_{\mathbf{x}_{t-1} \in S} \mathcal{N}(\mathbf{x}_t; \sqrt{1-\beta_t}\mathbf{x}_{t-1}, \beta_t \mathbf{I}) \tag{2.3}$$

where $S = \{\mathbf{v}_{t-1}, \mathbf{e}_{t-1}\}$, $\mathbf{I}$ is the identity matrix, and $\mathbf{v}$ and $\mathbf{e}$ contain the one-hot encodings at timestep $t = 0$, but for timestep $t > 0$ they are continuous vectors that can be decoded into categories for the atom and bond types using the argmax function. The complete forward process going from M_0 to M_T is

defined as:

$$q(M_{1:T}|M_0) = \prod_{t=1}^{T} q(M_t|M_{t-1}) \tag{2.4}$$

Together, (2.3) and (2.4) form a Markov chain that in this instance of the forward process adds Gaussian noise to the input data (Ho, Jain, and Abbeel, 2020). The forward process allows for sampling of M_t in closed form (Ho, Jain, and Abbeel, 2020):

$$q(M_t|M_0) = \prod_{\mathbf{x}_0 \in S} \mathcal{N}(\mathbf{x}_t; \sqrt{\bar{\alpha}_t}\mathbf{x}_0, (1-\bar{\alpha}_t)\mathbf{I}) \tag{2.5}$$

where $S = \{\mathbf{v}_0, \mathbf{e}_0\}$. Using the reparameterisation trick on (2.5) allows us to directly sample M_t for any timestep t from M_0 as follows (Ho, Jain, and Abbeel, 2020):

$$\begin{gathered} M_t(M_0, \boldsymbol{\epsilon}_v, \boldsymbol{\epsilon}_e) = (h(\mathbf{v}_0, \boldsymbol{\epsilon}_v), h(\mathbf{e}_0, \boldsymbol{\epsilon}_e)) \\ \text{where } h(\mathbf{x}_0, \boldsymbol{\epsilon}_x) = \sqrt{\bar{\alpha}_t}\mathbf{x}_0 + \sqrt{1-\bar{\alpha}_t}\boldsymbol{\epsilon}_x, \\ \text{and } \boldsymbol{\epsilon}_v, \boldsymbol{\epsilon}_e \sim \mathcal{N}(\mathbf{0}, \mathbf{I}) \end{gathered} \tag{2.6}$$

Note that in (2.3) and (2.5) the covariances of the Gaussian distributions are fixed based on the noise schedule.

2.2.2 Categorical Distribution

For the forward process using a categorical distribution we follow the process outlined by Guan, Qian, et al., 2023. Hence, we can define one step of the forward process as:

$$q(M_t|M_{t-1}) = \prod_{\mathbf{x}_{t-1}, K_x \in S} \mathcal{C}(\mathbf{x}_t|(1-\beta_t)\mathbf{x}_{t-1} + \beta_t/K_x) \tag{2.7}$$

where $S = \{(\mathbf{v}_{t-1}, K_v), (\mathbf{e}_{t-1}, K_e)\}$ and $\mathcal{C}$ is a categorical distribution defined by the probabilities on the right-hand side of $|$ with K_x categories. The complete forward process going from M_0 to M_T is defined exactly as (2.4) and, together with (2.7), it forms a Markov chain adding categorical noise to the input molecule. Following Guan, Qian, et al., 2023, we can sample M_t in closed form:

$$q(M_t|M_0) = \prod_{\mathbf{x}_0, K_x \in S} \mathcal{C}(\mathbf{x}_t|\bar{\alpha}_t\mathbf{x}_0 + (1-\bar{\alpha}_t)/K_x) \tag{2.8}$$

where $S = \{(\mathbf{v}_0, K_v), (\mathbf{e}_0, K_e)\}$. Using the reparameterisation trick on (2.8) allows us to directly sample M_t for any timestep t from M_0 as (Guan, Qian, et al., 2023):

$$\begin{gathered} M_t = (h(\mathbf{v}_0, K_v, \mathbf{g}_v), h(\mathbf{e}_0, K_e, \mathbf{g}_e)) \\ \text{where } h(\mathbf{x}_0, K_x, \mathbf{g}_x) \sim \texttt{argmax}(\mathbf{g}_x + (\bar{\alpha}_t \mathbf{x}_0 + (1 - \bar{\alpha}_t)/K_x)), \\ \text{and } \mathbf{g}_v, \mathbf{g}_e \sim \text{Gumbel}(\mathbf{0}, \mathbf{1}) \end{gathered} \tag{2.9}$$

where we sample from the Gumbel distribution as:

$$\text{Gumbel}(\mathbf{0}, \mathbf{1}) = -\log(-\log(\text{Uniform}(\mathbf{0}, \mathbf{1}) + 10^{-30}) + 10^{-30})$$

2.2.3 Comparison of Noise Distributions

When working with molecules represented as graphs, which are inherently discrete, the choice of distribution for the noise has a great effect on how the features, atom types and bond types, are changed. This is a result of how the features are represented using different noise distributions. For continuous noise, the features can be represented using one-hot encoding, which can be noised using continuous noise. The feature from the noised one-hot encoding is obtained using argmax, where the index corresponds to a feature category, i.e. an atom type or a bond type. The effect of the choice of noise distribution is illustrated in Figure 2.4, where the continuous noise seldom makes changes to any features in the beginning before radically increasing the amount of features changed. Comparably, the categorical noise has a more smooth increase in the amount of features changes during the forward process.

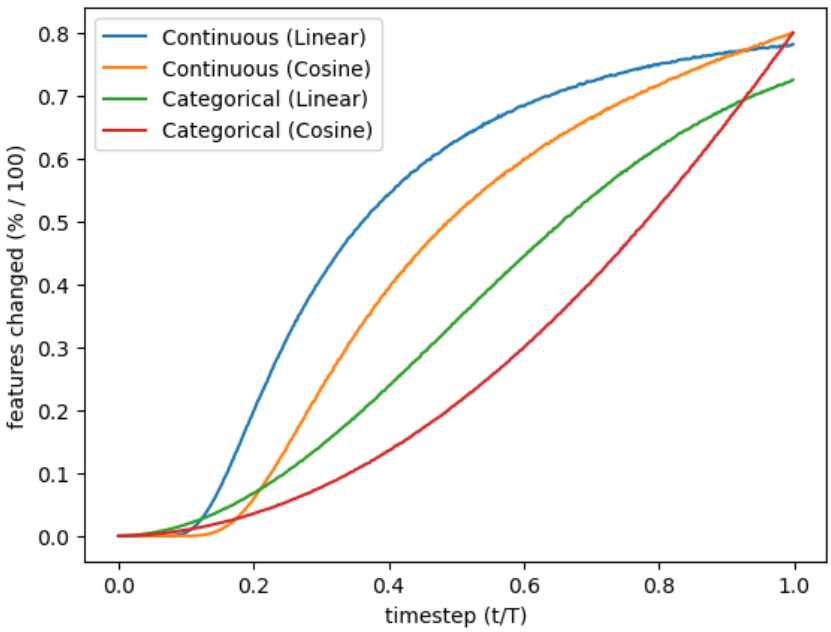

Figure 2.4: Number of features changed during the forward process compared to the total number of features.

Figure 2.4 also depicts the difference between the choice of noise schedule, as explained in Section 2.1.3. Using a linear schedule results in more features changes in the beginning of the forward process and less in the end of the process. Comparably, as expected, using the cosine schedule results in smoother curves in the beginning and end of the process but more changes in the middle of the process. The difference, seen in Figure 2.4, is also clearly visible in practice. As illustrated by Figure 2.5 and Figure 2.6, both distributions do not change the molecules in the very early stages of the forward process, $t/T = 0.02$. However, the radical increase in amount of features changed with the continuous noise is clearly visible in the $t/T = 0.2$, 0.4 and 0.7 stages. In these stages, the molecule is a lot more noisy when using the continuous noise compared with the use of categorical noise.

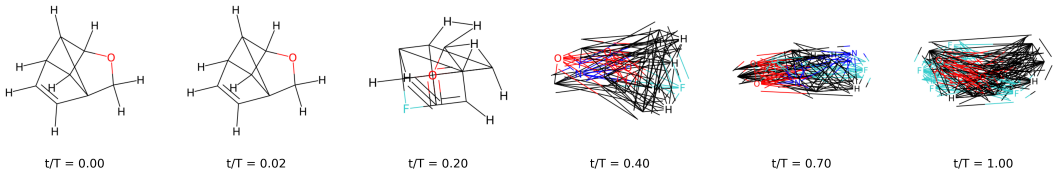

Figure 2.5: Illustration of the forward process using a continuous noise distribution with a cosine noise schedule.

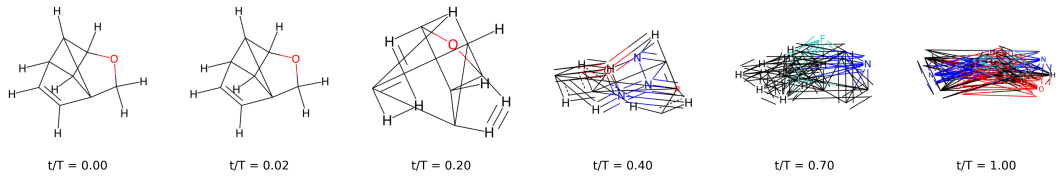

Figure 2.6: Illustration of the forward process using a categorical noise distribution with a cosine noise schedule.

In summary, when using a continuous noise in the forward process, the noise in the molecules accumulates more radically. This can make it harder for the denoising neural network, used in the reverse process, to learn to properly clean the molecules.

2.3 Reverse Process

The reverse process depends on the type of prior distribution that is used for the atom types and bond types. In the reverse process, the β_t, α_t and $\bar{\alpha}_t$ values, as described in Section 2.1, are utilised. In this section, we cover how both continuous and categorical distributions can be used as a prior for the reverse process of a diffusion model. However, independent from the type of prior distribution, the reverse process utilises a neural network parameterised by θ. The parameterised neural network can either be used to predict the noise used in the forward process, $\hat{\boldsymbol{\epsilon}} = \boldsymbol{\epsilon}_\theta(M_t, t)$, or it can be used to predict the clean molecule, $\hat{M}_0 = M_\theta(M_t, t)$, with $\boldsymbol{\epsilon}_\theta$ and M_θ being parameterised neural networks predicting the noise and the clean molecule, respectively. For convenience, we superscript the predictions with v and e to indicate the predictions related to the atom and bond types from the parameterised neural networks, e.g. $\boldsymbol{\epsilon}^v_\theta$ and M^v_θ for the predictions related to atom types.

2.3.1 Continuous Distribution

For the continuous distribution, we use a Gaussian distribution as a prior, which means that formally one step in the reverse process can be described as

follows (Ho, Jain, and Abbeel, 2020):

$$
\begin{aligned}
p(M_{t-1}|M_t) &= \prod_{\mathbf{x}_t \in S} \mathcal{N}(\mathbf{x}_{t-1}; \boldsymbol{\mu}_t(\mathbf{x}_t, t), \beta_t \mathbf{I}) \\
\text{where } \boldsymbol{\mu}_t(\mathbf{x}_t, t) &= \frac{1}{\sqrt{\alpha_t}} \left(\mathbf{x} - \frac{\beta_t}{\sqrt{1-\bar{\alpha}_t}} \boldsymbol{\epsilon} \right)
\end{aligned}
\tag{2.10}
$$

where $S = \{\mathbf{v}_t, \mathbf{e}_t\}$, $\mathbf{v}$ and $\mathbf{e}$ contain continuous values that can be decoded into the atom and bond types using the argmax function, $\boldsymbol{\epsilon}$ is the added noise from (2.6), and $\mathbf{I}$ is the identity matrix. The complete reverse process going from M_T to M_0 is defined as (Ho, Jain, and Abbeel, 2020):

$$
p(M_{0:T}) = p(M_T) \prod_{t=1}^{T} p(M_{t-1}|M_t) \tag{2.11}
$$

where $p(M_T) = \mathcal{N}(\mathbf{0}, \mathbf{I})$. Together, (2.10) and (2.11) form a Markov chain that through the neural network learns Gaussian transitions from M_t to M_{t-1} (Ho, Jain, and Abbeel, 2020), and they show the formal definition of the reverse process. During generation, the added noise $\boldsymbol{\epsilon}$ is unknown; therefore, we approximate the posterior using a parameterised neural network. We let the neural network predict the noise, $\boldsymbol{\epsilon}$ in (2.6), as $\hat{\boldsymbol{\epsilon}} = \boldsymbol{\epsilon}_\theta(M_t, t) = (\boldsymbol{\epsilon}_v, \boldsymbol{\epsilon}_e)$, and thus parameterise $\boldsymbol{\mu}_t$ in (2.10) as follows (Ho, Jain, and Abbeel, 2020):

$$
\begin{gathered}
\boldsymbol{\mu}_t(\mathbf{x}_t, t) = \frac{1}{\sqrt{\alpha_t}} \left(\mathbf{x}_t - \frac{\beta_t}{\sqrt{1-\bar{\alpha}_t}} \boldsymbol{\epsilon}_\theta^x(M_t, t) \right) \\
\text{for } \mathbf{x}_t \in \{\mathbf{v}_t, \mathbf{e}_t\}
\end{gathered}
\tag{2.12}
$$

This gives us the following parameterisation of (2.10) (Ho, Jain, and Abbeel, 2020):

$$
\begin{gathered}
M_{t-1} = (h(\mathbf{v}_t, \mathbf{z}_v), h(\mathbf{e}_t, \mathbf{z}_e)) \\
\text{where } h(\mathbf{x}_t, \mathbf{z}_x) = \boldsymbol{\mu}_t(\mathbf{x}_t, t) + \sqrt{\beta_t} \mathbf{z}_x \\
\text{and } \mathbf{z}_v, \mathbf{z}_e \sim \mathcal{N}(\mathbf{0}, \mathbf{I})
\end{gathered}
\tag{2.13}
$$

2.3.2 Categorical Distribution

With the categorical prior, we follow the process outlined by Guan, Qian, et al., 2023. Hence, we define one step of the reverse process as follows:

$$
\begin{aligned}
p(M_{t-1}|M_t, M_0) &= \prod_{\mathbf{x}_t, \mathbf{x}_0, K_x \in S} \mathcal{C}(\mathbf{x}_{t-1}|\, \tilde{\mathbf{c}}_t(\mathbf{x}_t, \mathbf{x}_0, K_x)) \\
\text{where } \tilde{\mathbf{c}}_t(\mathbf{x}_t, \mathbf{x}_0, K_x) &= \mathbf{c}^\star / \sum_{k=1}^{K_x} c_k^\star, \\
\text{and } \mathbf{c}^\star(\mathbf{x}_t, \mathbf{x}_0, K_x) &= [\alpha_t \mathbf{x}_t + (1-\alpha_t)/K_x] \odot [\bar{\alpha}_{t-1}\mathbf{x}_0 + (1-\bar{\alpha}_{t-1})/K_x]
\end{aligned}
\tag{2.14}
$$

where $S = \{(\mathbf{v}_t, \mathbf{v}_0, K_v), (\mathbf{e}_t, \mathbf{e}_0, K_e)\}$, $\odot$ is element-wise multiplication, $\mathcal{C}$ is a categorical distribution with K_x categories, and $\tilde{\mathbf{c}}_t$ contains probabilities for either the atom types or the bond types. The definition of the complete reverse process, i.e. going from M_T to M_0, is defined exactly as (2.11), where $p(M_T) = \mathcal{C}(\mathbf{v}_T|\, \mathbf{1}/K_v) \cdot \mathcal{C}(\mathbf{e}_T|\, \mathbf{1}/K_e)$. When generating new molecules, M_0 is not known, as such we approximate the posterior using a parameterised neural network. We let the neural network predict M_0 as $\hat{M}_0 = (\hat{\mathbf{v}}_0, \hat{\mathbf{e}}_0) = M_\theta(M_t, t)$ and feed $\hat{M}_0$ through (2.14) in place of M_0. With this parameterisation of the neural network we can sample M_{t-1} as (Guan, Qian, et al., 2023):

$$
\begin{aligned}
M_{t-1} &= (h(\mathbf{v}_t, \hat{\mathbf{v}}_0, \mathbf{g}_v, K_v), h(\mathbf{e}_t, \hat{\mathbf{e}}_0, \mathbf{g}_e, K_e)) \\
\text{where } h(\mathbf{x}_t, \hat{\mathbf{x}}_0, \mathbf{g}_x, K_x) &\sim \texttt{argmax}(\mathbf{g}_x + \tilde{\mathbf{c}}_t(\mathbf{x}_t, \hat{\mathbf{x}}_0, K_x)), \\
\text{and } \mathbf{g}_v, \mathbf{g}_e &\sim \text{Gumbel}(\mathbf{0}, \mathbf{1})
\end{aligned}
\tag{2.15}
$$

where we sample from the Gumbel distribution as

$$
\text{Gumbel}(\mathbf{0}, \mathbf{1}) = -\log(-\log(\text{Uniform}(\mathbf{0}, \mathbf{1}) + 10^{-30}) + 10^{-30})
$$

2.4 Loss Function

Different formulations of the loss function have been proposed (Ho, Jain, and Abbeel, 2020; Guan, Qian, et al., 2023). The applied formulation depends on the task at hand, the distributions used in the diffusion model and the parameterisation of the neural network. In our case, we work on a dataset of molecules represented as simple graphs, $\mathcal{M} = (V, E)$. As the neural network

works on noisy molecules at a timestep, t, we construct the noisy molecules during training using randomly sampled timesteps. The task is then to reconstruct the molecules for the timestep, $t-1$, using the prediction from the neural network. The loss is calculated based on the noisy molecules at timestep, t, and the prediction from the neural network.

When parameterising the neural network to predict the noise applied to the sample and working with a continuous distribution for the diffusion model, we use the simple loss formulation proposed by Ho, Jain, and Abbeel, 2020. Ho, Jain, and Abbeel, 2020 proposed to use an unweighted mean squared error as a simple loss function, given as:

$$\begin{aligned} L &= \mathbb{E}_{M_t,t,\boldsymbol{\epsilon}}\left[\|\boldsymbol{\epsilon} - \boldsymbol{\epsilon}_\theta(M_t,t)\|^2\right] \\ &= \sum_{\boldsymbol{\epsilon}_x,\boldsymbol{\epsilon}_\theta^x(M_t,t)\,\in\, S} \mathbb{E}_{M_t,t,\boldsymbol{\epsilon}_x}\left[\|\boldsymbol{\epsilon}_x - \boldsymbol{\epsilon}_\theta^x(M_t,t)\|^2\right] \end{aligned} \tag{2.16}$$

where $S = \{(\boldsymbol{\epsilon}_v, \boldsymbol{\epsilon}_\theta^v(M_t,t)), (\boldsymbol{\epsilon}_e, \boldsymbol{\epsilon}_\theta^e(M_t,t))\}$, M_t is calculated using (2.6) for atoms and bonds separately, and $\boldsymbol{\epsilon}$ is the noise used to calculate M_t. Similarly, a formulation for an unweighted mean squared error when predicting the clean molecule can be used as a simple loss. However, Ho, Jain, and Abbeel, 2020 found that this formulation did not work very well.

When parameterising the neural network to predict the clean sample and working with a categorical distribution for the diffusion process, Guan, Qian, et al., 2023 proposed to use the KL-divergence as a loss:

$$L = D_{\mathrm{KL}}(q(M_{t-1}|M_0) \parallel p(M_{t-1}|M_t))$$

which can be computed directly for the categorical distribution as the sum of the loss for atom and bond types (Guan, Qian, et al., 2023):

$$\begin{aligned} L_c &= \sum_{\mathbf{x}_t,\mathbf{x}_0,\hat{\mathbf{x}}_0,K_x\,\in\, S} L_x(\mathbf{x}_t,\mathbf{x}_0,\hat{\mathbf{x}}_0,K_x) \\ \text{where } L_x(\mathbf{x}_t,\mathbf{x}_0,\hat{\mathbf{x}}_0,K_x) &= \sum_{k=1}^{K_x} \tilde{\mathbf{c}}_t(\mathbf{x}_t,\mathbf{x}_0,K_x)_k \log \frac{\tilde{\mathbf{c}}_t(\mathbf{x}_t,\mathbf{x}_0,K_x)_k}{\tilde{\mathbf{c}}_t(\mathbf{x}_t,\hat{\mathbf{x}}_0,K_x)_k} \end{aligned} \tag{2.17}$$

where $S = \{(\mathbf{v}_t,\mathbf{v}_0,\hat{\mathbf{v}}_0,K_v), (\mathbf{e}_t,\mathbf{e}_0,\hat{\mathbf{e}}_0,K_e)\}$.

2.5 Training

The training of the parameterised neural network depends on the type of noise distribution that is used for the molecules. In this section, we cover how both continuous and categorical distributions can be used during training.

2.5.1 Continuous Distribution

When training the parameterised neural network with standard Gaussian noise, we use the simple loss in (2.16) and follow the algorithm depicted in Algorithm 6.

Algorithm 6 Training - Continuous

repeat
  $M_0 \sim q(M_0)$
  $t \sim \text{Choice}(\{1, \dots, T\})$
  $\boldsymbol{\epsilon}_v, \boldsymbol{\epsilon}_e \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$
  $M_t \leftarrow\!\shortmid M_t(M_0, \boldsymbol{\epsilon}_v, \boldsymbol{\epsilon}_e)$
  $\hat{\boldsymbol{\epsilon}} \leftarrow\!\shortmid \boldsymbol{\epsilon}_\theta(M_t, t)$
  Take gradient descent step on
    $\nabla_\theta \left(\|\boldsymbol{\epsilon}_v - \hat{\boldsymbol{\epsilon}}^v\|^2 + \|\boldsymbol{\epsilon}_e - \hat{\boldsymbol{\epsilon}}^e\|^2 \right)$
until converged

In Algorithm 6, we first get a molecule, M_0, from the dataset, $q(M_0)$, and then sample a timestep t randomly and the noise, $\boldsymbol{\epsilon}_v$ and $\boldsymbol{\epsilon}_e$, from a standard Gaussian distribution. Next, we add noise to the molecule, M_0, using $M_t(M_0, \boldsymbol{\epsilon}_v, \boldsymbol{\epsilon}_e)$ following (2.6) before using the parameterised neural network, $\boldsymbol{\epsilon}_\theta$, to predict the noise used in the forward process, $\hat{\boldsymbol{\epsilon}}$. Lastly, we take a gradient descend step using the calculated loss following (2.16). We repeat these steps until convergence or for a given number of iterations over the dataset.

2.5.2 Categorical Distribution

When training the parameterised neural network with categorical noise, we use the loss in (2.17) and follow the algorithm depicted in Algorithm 7.

Algorithm 7 Training - Categorical

repeat
  $M_0 \sim q(M_0)$
  $t \sim \text{Choice}(\{1, \ldots, T\})$
  $\mathbf{g}_v, \mathbf{g}_e \sim \text{Gumbel}(\mathbf{0}, \mathbf{1})$
  $M_t \leftarrow\!\shortmid (h(\mathbf{v}_0, K_v, \mathbf{g}_v), h(\mathbf{e}_0, K_e, \mathbf{g}_e))$
    where $h(\mathbf{x}_0, K_x, \mathbf{g}_x) \sim \texttt{argmax}(\mathbf{g}_x + (\bar{\alpha}_t \mathbf{x}_0 + (1 - \bar{\alpha}_t)/K_x))$
  $\hat{M}_0 \leftarrow\!\shortmid M_\theta(M_t, t)$
  Take gradient descent step on $\nabla_\theta(L_v + L_e)$
until converged

In Algorithm 7, we first get a molecule, M_0, from the dataset, $q(M_0)$, before sampling the timestep, t, randomly and the noise, $\mathbf{g}_v$ and $\mathbf{g}_e$, from a Gumbel distribution. Next, we noise the molecule, M_0, to sample M_t following (2.9). Lastly, we take a gradient descent step on the loss from (2.17) based on the prediction of the clean molecule, $\hat{M}_0$, passed through (2.14), replacing M_0 with the prediction. Similarly to Algorithm 6, we repeat these steps.

2.6 Generation

Generating new molecules depends on the type of prior distribution used for the atom types and bond types. In this section, we cover how a parameterised neural network can be used for generating new molecules with a continuous and categorical prior. The general idea is to iteratively apply the reverse process for T timesteps utilising the neural network as an approximator.

2.6.1 Continuous Distribution

When generating new molecules with a standard Gaussian distribution as a prior, we utilise the neural network as an approximator for the noise in the molecule, and feed the noise predictions of the neural network into (2.13). This is done iteratively for T steps to remove noise from the molecule, as depicted in Algorithm 8.

Algorithm 8 Generation - Continuous

$M_T \sim \mathcal{N}(\mathbf{v}_T; \mathbf{0}, \mathbf{I}) \cdot \mathcal{N}(\mathbf{e}_T; \mathbf{0}, \mathbf{I})$
for $t = T$ to 1 **do**
    $\mathbf{z}_v, \mathbf{z}_e \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$ if $t > 1$ else $\mathbf{z}_v, \mathbf{z}_e = \mathbf{0}$
    $M_{t-1} \leftarrow\!\shortmid (h(\mathbf{v}_t, \mathbf{z}_v), h(\mathbf{e}_t, \mathbf{z}_e))$
        where $h(\mathbf{x}_t, \mathbf{z}_x) = \boldsymbol{\mu}_t(\mathbf{x}_t, t) + \sqrt{\beta_t}\mathbf{z}_x$
end for

In Algorithm 8, we first sample noise from a standard Gaussian distribution to get the completely noisy molecule M_t. The reverse process is then iteratively applied for T timesteps, where $\mathbf{z}_v, \mathbf{z}_e$ is sampled from a standard Gaussian distribution and M_{t-1} is calculated using (2.13). However, following Ho, Jain, and Abbeel, 2020, in the last timestep, $t = 1$, no noise is sampled and instead $\mathbf{z}_v, \mathbf{z}_e = \mathbf{0}$.

2.6.2 Categorical Distribution

When generating new molecules with a categorical distribution as a prior, we utilise the neural network as an approximator for the clean molecules, and feed the predictions of the neural network into (2.15). This is done iteratively to remove noise from the molecule until a clean and new molecule is generated, as depicted in Algorithm 9.

Algorithm 9 Generation - Categorical

$M_T \sim \mathcal{C}(\mathbf{v}_T|\, \mathbf{1}/K_v) \cdot \mathcal{C}(\mathbf{e}_T|\, \mathbf{1}/K_e)$
for $t = T$ to 1 **do**
    $\mathbf{g}_v, \mathbf{g}_e \sim \text{Gumbel}(\mathbf{0}, \mathbf{1})$
    $M_{t-1} \leftarrow\!\shortmid (h(\mathbf{v}_t, \hat{\mathbf{v}}_0, \mathbf{g}_v, K_v), h(\mathbf{e}_t, \hat{\mathbf{e}}_0, \mathbf{g}_e, K_e))$
        where $h(\mathbf{x}_t, \hat{\mathbf{x}}_0, \mathbf{g}_x, K_x) \sim \texttt{argmax}(\mathbf{g}_x + \tilde{\mathbf{c}}_t(\mathbf{x}_t, \hat{\mathbf{x}}_0, K_x))$
end for

In Algorithm 9, we first randomly sample a completely noisy molecule and convert the molecule to one-hot encoding. We then iteratively apply the reverse process for T timesteps following (2.15). Similarly to Algorithm 8, when $t = 1$ we consider a special case of the reverse process. Here, the implication is

that we set $[\bar{\alpha}_{t-1}\mathbf{x}_0 + (1-\bar{\alpha}_{t-1})/K_x] = \mathbf{x}_0$ in the definition of $\mathbf{c}^\star(\mathbf{x}_t, \mathbf{x}_0, K_x)$ in (2.14), i.e. the right term of $\mathbf{c}^\star$ is equal to the prediction from the neural network.

2.7 Combining Continuous and Categorical Distributions

Our graph representation of a molecule in 3D-space, $\mathcal{M} = (V, E, A, D)$, introduced in Section 1.3, contains both inherently discrete components and inherently continuous components. Whereas, the atom types and bond types are inherently discrete, the triplet angles and dihedral angles are inherently continuous. This makes it logical to use different distributions for the different components of the graph in the diffusion processes.

When using diffusion models, as described in this chapter, this does not pose a problem. As it can be seen in both (2.3) and (2.7), the distribution for our simple graphs is a product of 2 independent distributions, 1 for each of the components in the graphs. This makes it easy to use different distributions for the components to create a new combined distribution for the graphs. For example, using categorical distributions for atom types and bond types, and continuous distributions for triplet angles and dihedral angles, we can define one step of the forward process as:

$$\begin{aligned} q(M_t|M_{t-1}) = &\prod_{\mathbf{x}_{t-1}, K_x \in S_d} \mathcal{C}(\mathbf{x}_t|(1-\beta_t)\mathbf{x}_{t-1} + \beta_t/K_x) \\ &\cdot \prod_{\mathbf{x}_{t-1} \in S_c} \mathcal{N}(\mathbf{x}_t; \sqrt{1-\beta_t}\mathbf{x}_{t-1}, \beta_t \mathbf{I}) \end{aligned} \tag{2.18}$$

where $S_d = \{(\mathbf{v}_{t-1}, K_v), (\mathbf{e}_{t-1}, K_e)\}$ and $S_c = \{\mathbf{a}_{t-1}, \mathbf{d}_{t-1}\}$. This is an inherent property of how diffusion models work, as it uses products and sums to combine components, which is also apparent in the definition of the loss function. Adding to the above example, we can define the loss for such a model as:

$$\begin{aligned} L = &\sum_{\mathbf{x}_t, \mathbf{x}_0, \hat{\mathbf{x}}_0, K_x \in S_d} L_x(\mathbf{x}_t, \mathbf{x}_0, \hat{\mathbf{x}}_0, K_x) \\ &+ \sum_{\boldsymbol{\epsilon}_x, \boldsymbol{\epsilon}_\theta^x(M_t,t) \in S_c} \mathbb{E}_{M_t, t, \boldsymbol{\epsilon}_x} \left[\|\boldsymbol{\epsilon}_x - \boldsymbol{\epsilon}_\theta^x(M_t, t)\|^2 \right] \end{aligned} \tag{2.19}$$

where $S_d = \{(\mathbf{v}_t, \mathbf{v}_0, \hat{\mathbf{v}}_0, K_v), (\mathbf{e}_t, \mathbf{e}_0, \hat{\mathbf{e}}_0, K_e)\}$ contains the atom and bond components and $S_c = \{(\boldsymbol{\epsilon}_a, \boldsymbol{\epsilon}_\theta^a(M_t, t)), (\boldsymbol{\epsilon}_d, \boldsymbol{\epsilon}_\theta^d(M_t, t))\}$ contains the triplet and dihedral angle components. Note that in this case the model predicts the added noise for the triplet angles and dihedral angles but predicts the clean atom types and bond types, $\boldsymbol{\epsilon}_\theta(M_t, t) = (\hat{\mathbf{v}}_0, \hat{\mathbf{e}}_0, \hat{\boldsymbol{\epsilon}}_a, \hat{\boldsymbol{\epsilon}}_d)$.

When using categorical diffusion on the angles A and D, we define the categories based on a approximation of the angles by grouping the angles into intervals. The number of groups for triplet angles is K_a and for dihedral angles it is K_d. Using this logic, we let $\mathbf{a} \in \mathbb{R}^{N \times N_a \times K_a}$ and $\mathbf{d} \in \mathbb{R}^{N \times N_d \times K_d}$ be the one-hot encodings of the angles and dihedral angles, respectively. With this approach to categorical diffusion of the angles, we use the mean of the intervals as the angles during sampling. For example, if the reversed/cleaned molecule contains category 1 for an angle and category 1 represents the interval $[0, 5)$, we use 2.5 as the angle. In summary, we can use categorical diffusion for the angles, similarly to how we apply it on atom types and bond types.

2.8 Code Implementation

The implementation of our diffusion models can be found on the GitHub repository at `https://github.com/MouritsJJ/diffgraph`. Our implementation has the following file structure:

```
/
├── README.md
├── experiments
├── Graph_Framework
└── notebooks
```

The `notebooks` folder contains the python notebook, `metrics.ipynb`, we use for running evaluation on the generated molecules. It also contains a python file, `schedule_comp.py`, that generates a graph comparing our implementation of the linear and cosine noise schedules.

```
notebooks
├── metrics.ipynb
└── schedule_comp.py
```

The `experiments` folder contains all the configuration `yaml` files, we used for running the experiments described in Chapter 3. There is a folder matching each configuration of our models following the naming convention `dist1_dist2_hydrogen_schedule`. For example, the folder `cat_con_noH_cosine` contains the `yaml` files for the model with categorical atom and bond types, continuous angles, implicit hydrogen atoms and the cosine noise schedule. For configurations without spatial information we do not specify `dist2`.

```
experiments
├── cat_H_cosine
├── cat_H_linear
├── cat_noH_cosine
├── cat_noH_linear
├── con_H_cosine
├── con_H_linear
├── con_noH_cosine
├── con_noH_linear
├── cat_cat_H_cosine
├── cat_cat_noH_cosine
├── cat_con_H_cosine
└── cat_con_noH_cosine
```

The `Graph_Framework` folder contains all the folders and files with code for the diffusion processes, neural networks, conversion algorithms, and evaluation metrics. Throughout the code we use the following packages: PyTorch[1], PyTorch Geometric[2], NumPy[3], RDKit[4], and py3Dmol[5].

PyTorch is used for everything concerning the neural network architecture including the forward pass of the model and updating model parameters through backpropagation.

PyTorch Geometric is used for accessing the dataset, pre-processing of the dataset, and batching of graphs.

NumPy is used in our conversion algorithms when working with vectors and matrices.

[1] `https://pytorch.org/`
[2] `https://pyg.org/`
[3] `https://numpy.org/`
[4] `https://www.rdkit.org/`
[5] `https://pypi.org/project/py3Dmol/`

RDKit is used to build molecules from our graph representations during evaluation in our experiments.
py3Dmol is used in combination with RDKit to visualise molecules in 3D.
In the root of `Graph_Framework`, we have the python files for the training loop (`main.py`), generation loop (`generate.py`), evaluation metrics (`metrics.py`) and pre-processing of the dataset (`process_dataset.py`).

```
Graph_Framework
├── main.py
├── generate.py
├── metrics.py
└── process_dataset.py
```

The folders inside `Graph_Framework` serve different purposes. The `configs` folder contains template `yaml` files for configuration of the training and generation runs. These can be used as explanatory examples of how to setup training of a model and generation of new samples with models using this diffusion framework we have implemented. `datasets` contains all the datasets we use for the experiments. In this case, all the different representations of the QM9 dataset are defined by a single file, `qm9data.py`. The `utils` folder contains utility files for ease of use of the framework. `util.py` provides functions for the framework regarding models, optimisers, logging, datasets, and checkpoints. `graph_utils.py` provides functions for batching of graphs, encoding the no-edge category, mirroring of an adjacency matrix, computing angle masks, and conversion algorithms. `mol_utils.py` provides functions for converting from our graph representations to RDKit molecules.

```
Graph_Framework
├── configs
│   ├── template_train.yml
│   └── template_generate.yml
├── datasets
│   └── qm9data.py
└── utils
    ├── util.py
    ├── graph_utils.py
    └── mol_utils.py
```

The `models` folder in `Graph_Framework` contains a folder for each of the models that can be used in the framework and an example folder `modelname`. All folders must contain three files, `model.py`, `train.py`, and `sample.py`. To avoid repetition, we only show the files for `modelname`. `model.py` is the file containing the machine learning model as a PyTorch `Module`. `train.py` is the file used during training of the model and must contain the function `loss_fn` that uses a dataset sample, the diffusion processes and the model to calculate the loss used in backpropagation. Moreover, the file must contain the function `val_fn` that runs validation after a user specified number of epochs. `sample.py` is the file used during generation of molecules and must contain the function `sample_batch` that is used to sample a random batch, and the function `sample_reverse` that reverses the sampled batch one step. Lastly, the file must contain the function `sample_mols` that converts the reversed graphs into RDKit molecules.

```
Graph_Framework
└── models
    ├── modelname
    │   ├── model.py
    │   ├── train.py
    │   └── sample.py
    ├── nncon
    ├── nncat
    ├── nncombrad
    └── nncombcat
```

The `diffusion` folder contains code for the categorical and continuous diffusion processes described in this chapter. `con_diffusion.py` implements the continuous version and `cat_diffusion.py` implements the categorical version. As the choice of noise schedule is independent from the diffusion processes, we implement the different schedules inside `noise_schedules.py`.

```
Graph_Framework
└── diffusion
    ├── con_diffusion.py
    ├── cat_diffusion.py
    └── noise_schedules.py
```

Every line of code in the implementation has been written by us. However, in some places we have taken inspiration from published literature. Regarding the neural network architecture, we take inspiration from Vignac et al., 2023 for the general architecture and use functionality proposed in other literature, as detailed in Section 3.1.2 and 3.2.2. For the diffusion processes, we base our implementation on the equations presented in this chapter. Additionally, for the loss in the categorical diffusion process we take inspiration from an implementation made by our external supervisor, Alessandro Tibo.

Chapter 3

Experiments

In this chapter, we report on the experiments that we conduct to evaluate 2 aspects of our diffusion models. First, how a continuous diffusion model compares to a categorical diffusion model when working with graphs, which are inherently discrete, and how different noise schedules affects these models. The second aspect covers the generation of 3D conformations using our spatial graph representation. In both experiments, we train the models for 1000 epochs with a validation run every 5 epochs. During a validation run, 1000 molecules are generated, and the validity, uniqueness, and novely metrics in Section 1.1 are reported. We use the metrics from the validation runs to determine the best model for the generation process. In the generation process, we generate 10,000 molecules 10 times and calculate the mean and standard deviation over the evaluation metrics. In the following, we cover various details of the experiments to ensure the reproducibility of the presented results.

3.1 Testing Distributions and Noise Schedules

In this experiment, we use the simple graph representation without angles, $\mathcal{M} = (V, E)$. We focus on comparing the continuous and categorical versions of the diffusion model and how these are affected by different noise schedules. Additionally, we conduct the experiment with implicit and explicit hydrogen atoms following the approach by Vignac et al., 2023 and Le et al., 2023. To evaluate the generated molecules and compare our models, we use the validity, uniqueness and novelty metrics presented in Section 1.1, as we focus on how these diffusion models work in the domain of de novo drug design. When comparing our models with models proposed by others, we only use the V, U_v, and N_u metrics from Section 1.1, as these are reported for the other models.

3.1.1 Setup

For the parameters in the noise schedules, we use the same parameters for both the continuous and the categorical version of the diffusion models. For the linear noise schedule, we use $\beta_{min} = 10^{-4}$ and $\beta_{max} = 0.02$. For the cosine noise schedule, we use the proposed values of $s = 0.008$, $\beta_{min} = 0.0$ and $\beta_{max} = 0.999$ (Nichol and Dhariwal, 2021). When working with the categorical version of the diffusion model, we convert the encodings to a log soft one-hot encoding following the implementation by Hoogeboom, Nielsen, et al., 2021. This encoding replaces the 0's in the one-hot encoding with 10^{-30} and applies the log function to the encoding. We use the encoding when applying Gumbel noise in (2.9) and (2.15).

The dataset used in the experiment is the QM9 dataset (Wu et al., 2017) which is accessed through PyTorch Geometric (PyG). QM9 consists of close to 130,000 small molecules that only use the heavy atoms carbon, nitrogen, oxygen, and flour with each molecule having up to 9 heavy atoms. For the QM9 dataset we use the first 100k molecules for the training data. To convert a molecule from the QM9 dataset into a graph $\mathcal{M} = (V, E)$, we use the atom types, bond types, and edge list, which are all included for each molecule. Using the `to_dense_adj` function from PyG, we can convert the bond types and edge list into a categorical adjacency matrix. In the adjacency matrix, each entry is either 1 if the atoms are not connected or the category for the bond type if the atoms are connected. Here, we use category 0 as padding, such that the neural network can differentiate between the category for no bond between existing atoms, 1, and the category for bonds for non-existing atoms, 0.

3.1.2 Denoising Neural Network

For this experiment, we use the neural network shown in Figure 3.1 and Figure 3.2. For a forward pass through the neural network, we first embed the graph using MLPs, one for each component of the representation. Next, we apply 6 graph transformer layers, before decoding the features using different MLPs. This is shown in the left part of Figure 3.1. Each graph transformer layer first consists of a self-attention block and MLPs that utilise dropout and

layer normalisation, as shown to the right in Figure 3.1.

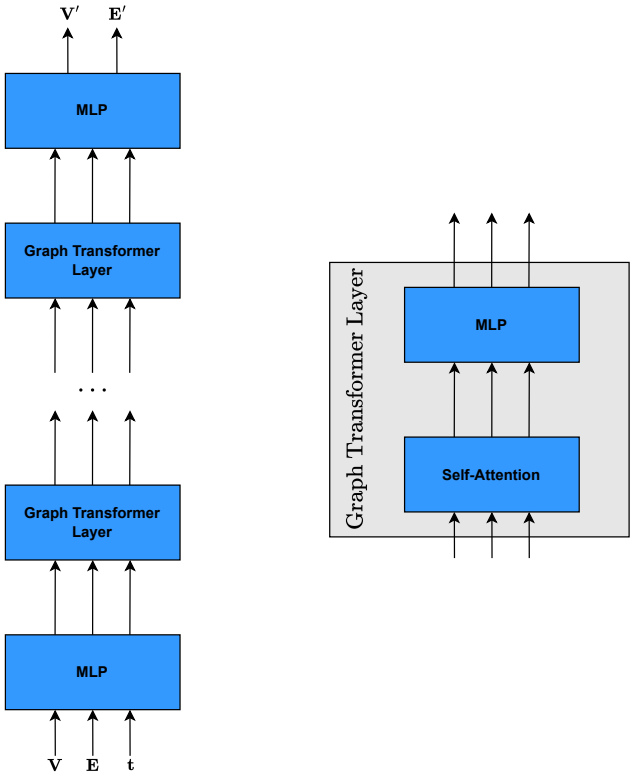

Figure 3.1: Left: Neural network architecture with repeating graph transformer layers. Right: Structure of a graph transformer layer.

The encoding and decoding MLPs consists of several linear layers with ReLU activation functions in between. For the categorical diffusion model, we first apply a learnable embedding for the atom types and bond types and a fixed positional embedding (Vaswani et al., 2023) of the timesteps $\mathbf{t}$ as part of the encoding MLPs. Additionally, we softmax encode the output of the decoding MLPs, when using categorical diffusion.

The self-attention block computes attention scores based on the atom types, $\mathbf{V}$, and the bond types, $\mathbf{E}$, as shown in the left part of Figure 3.2. The attention scores are used for updating every component of the graph except the timestep embedding, which is updated in a separate block. As the attention scores have the same dimensions as our bonds, $\mathbf{E}$, we use summation to reduce the dimensions to fit the dimensions of the atom types, $\mathbf{V}$. The updated atom and bond types are obtained by flattening the heads of the corresponding scores and incorporating the timesteps $\mathbf{t}$ into the result. In the self-attention block, we

utilise FiLM as proposed by Perez et al., 2017, and defined as $\text{FiLM}(\mathbf{Y}_1, \mathbf{Y}_2) = \mathbf{Y}_1\mathbf{W}_1 + (\mathbf{Y}_1\mathbf{W}_2) \odot \mathbf{Y}_2 + \mathbf{Y}_2$ for 2 sets of learnable weights $\mathbf{W}_1$ and $\mathbf{W}_2$.

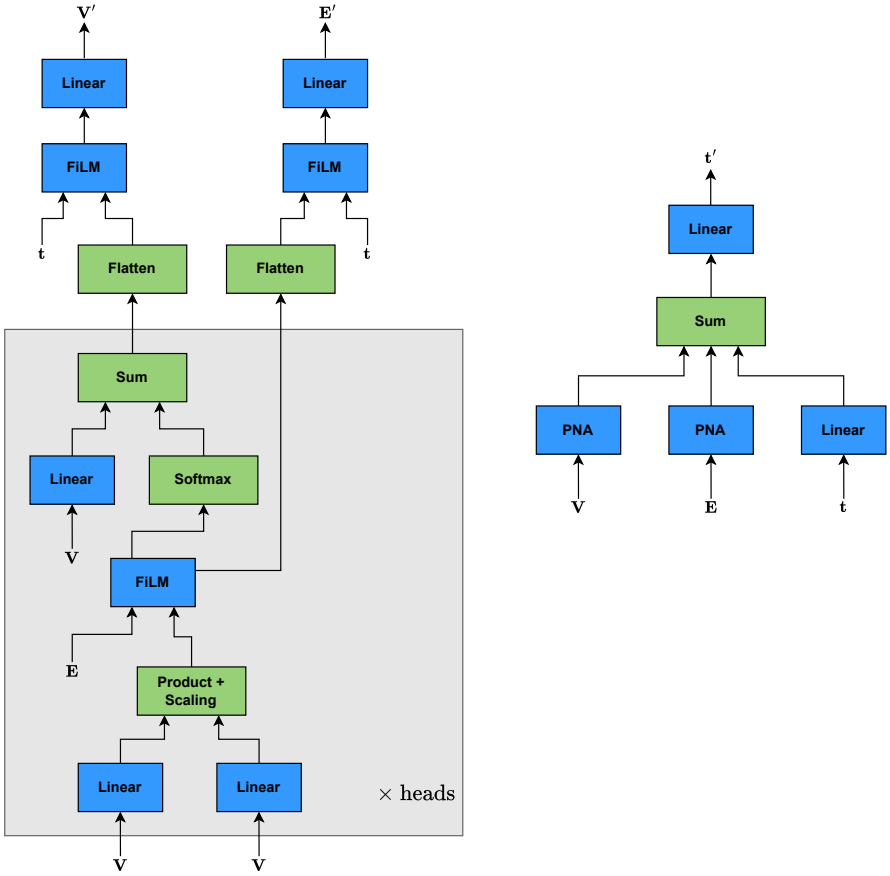

Figure 3.2: Overview of the self-attention block of our graph transformer layer.

The timestep embedding is updated as shown in the right part of Figure 3.2. Here, we only work with the timesteps, $\mathbf{t}$, as a global feature but other features can be added without the need for additional changes to the updating block. In the block, we let the atom types, $\mathbf{V}$, and bond types, $\mathbf{E}$, influence the update. We utilise PNA blocks as proposed by Corso et al., 2020 to let atom and bond types influence the timesteps. PNA is defined as the concatenation of 4 aggregations of the input, $\text{PNA}(\mathbf{Y}) = \text{cat}(\max(\mathbf{Y}), \min(\mathbf{Y}), \text{mean}(\mathbf{Y}), \text{std}(\mathbf{Y}))\mathbf{W}$ for learnable weights $\mathbf{W}$.

In this experiment, for the continuous diffusion models we let the neural network predict the added noise in the graph: $(\mathbf{V}', \mathbf{E}') = (\hat{\boldsymbol{\epsilon}}_v, \hat{\boldsymbol{\epsilon}}_e)$. For the categorical diffusion models, we let the neural network predict the clean graph: $(\mathbf{V}', \mathbf{E}') = (\hat{\mathbf{v}}_0, \hat{\mathbf{e}}_0)$.

3.1.3 Results

The validity and uniqueness scores of our diffusion models for molecule generation without spatial information can be seen in Table 3.1. From the scores it is clear that high validity and uniqueness is obtainable with our models, but there is not a single model that outperforms the others across all the scores. Training on molecules with implicit hydrogen atoms seems to improve the validity and the number of valid and unique molecules, U_s. However, it also affects the number of unique molecules compared with the valid molecules, U_v, with a slight drop in the scores. In short, using implicit hydrogen for training seem to be the slightly better approach for this dataset with our models. An added benefit of using implicit hydrogen is that the training and generating time is reduced threefold, even for QM9 which only contains very small molecules.

Distribution	**Schedule**	$\boldsymbol{V}$	$\boldsymbol{U_s}$	$\boldsymbol{U_v}$
Categorical	Cosine	88.75 ± 1.01	87.35 ± 1.01	98.43 ± 0.15
	Linear	89.59 ± 0.65	88.57 ± 0.62	**98.86** ± 0.13
Continuous	Cosine	77.83 ± 2.23	76.25 ± 2.08	97.98 ± 0.19
	Linear	74.40 ± 3.48	73.44 ± 3.36	**98.72** ± 0.14
Categorical (No H)	Cosine	**97.24** ± 0.19	90.64 ± 0.24	93.21 ± 0.20
	Linear	**97.22** ± 0.15	89.76 ± 0.19	92.33 ± 0.27
Continuous (No H)	Cosine	95.81 ± 0.16	91.61 ± 0.11	95.62 ± 0.14
	Linear	95.47 ± 0.17	**92.26** ± 0.30	96.63 ± 0.22

Table 3.1: Validity and uniqueness scores of the different diffusion models used for molecule generation without spatial information. The best results are marked in bold.

Looking at the choice of noise schedule, it can be seen from Table 3.1 that the use of either the linear or cosine noise schedule is similarly effective. The choice of noise schedule does not provide any benefit for the training or generation time either.

In Table 3.2 the novelty scores for our different models can be seen. Similar to Table 3.1, the choice of noise schedule does not seem to improve any of the novelty scores for our models.

Distribution	Schedule	N_s	N_v	N_u
Categorical	Cosine	49.27 ± 1.09	55.54 ± 1.79	56.42 ± 1.82
	Linear	**53.39** ± 1.44	59.60 ± 1.99	60.29 ± 2.00
Continuous	Cosine	51.73 ± 0.80	66.50 ± 1.49	67.87 ± 1.44
	Linear	**52.98** ± 1.64	**71.26** ± 1.35	**72.19** ± 1.32
Categorical (No H)	Cosine	39.39 ± 0.34	40.51 ± 0.37	43.46 ± 0.35
	Linear	39.35 ± 0.58	40.47 ± 0.60	43.84 ± 0.59
Continuous (No H)	Cosine	**52.31** ± 0.40	54.59 ± 0.46	57.10 ± 0.45
	Linear	**53.03** ± 0.50	55.55 ± 0.53	57.48 ± 0.52

Table 3.2: Novelty scores of the different diffusion models used for molecule generation without spatial information. The best results are marked in bold.

A comparison of our models with state of the art models can be seen in Table 3.3. Here, we compare our models with DiGress (Vignac et al., 2023) and EQGAT (Le et al., 2023). In this comparison, we fix the noise schedule to the cosine variant and use the models trained with explicit hydrogen. The superscript h in the N_u^h indicates that the novelty is calculated with explicit hydrogen atoms, and the absence of said h indicates that implicit hydrogen atoms are used. The results show that EQGAT is superior when it comes to validity and uniqueness, V and U_v, but this is also expected, as they use post processing in their evaluation pipeline that improves their validity and uniqueness scores. Looking at the novelty scores, both the categorical and continuous version of our model largely outperforms DiGress and EQGAT.

Model	V	U_v	N_u^h	N_u
Categorical	88.75 ± 1.01	98.43 ± 0.15	75.28 ± 0.41	56.42 ± 1.82
Continuous	77.83 ± 2.23	97.98 ± 0.19	**85.93** ± 0.41	**67.87** ± 1.44
DiGress	89.8	97.8		33.4
EQGAT	**98.96**	**100.00**	64.03	

Table 3.3: Comparison of our models and other proposed models using the validity, uniqueness, and novelty scores also reported by the other proposed models. Metrics from other proposed models are presented without validation directly from the literature. Missing values indicate that the authors do not report these metrics. The best results are marked in bold.

Table 3.4 shows the percentage of valid, unique and novel molecules out of all the generated molecules for the different models. Here, both our models performs slightly better than EQGAT when using explicit hydrogen. Compared to DiGress that uses implicit hydrogen, both of our models perform better. Both EQGAT and DiGress generate 10,000 molecules multiple times to report the mean.

Model	$\boldsymbol{N_s^h}$	$\boldsymbol{N_s}$
Categorical	**65.76** ± 0.67	49.27 ± 1.09
Continuous	65.52 ± 1.88	**51.73** ± 0.80
DiGress		29.33
EQGAT	62.71	

Table 3.4: Comparison of our models and other proposed models using the average percentage of valid, unique and novel molecules generated. Metrics from other proposed models are presented without validation directly from the literature. Missing values indicate that the authors do not report these metrics. The best results are marked in bold.

The drop in novelty from using explicit hydrogen atoms to implicit hydrogen atoms, seen in both Table 3.3 and Table 3.4, is a result of some of the generated molecules with explicit hydrogen atoms being charged molecules. As QM9 does not include charged molecules, all of the generated charged molecules will be considered novel. Removing the hydrogen atoms and making them implicit removes the charge from the molecules and results in fewer molecules being novel. An illustration of the scenario can be seen in Figure 3.3.

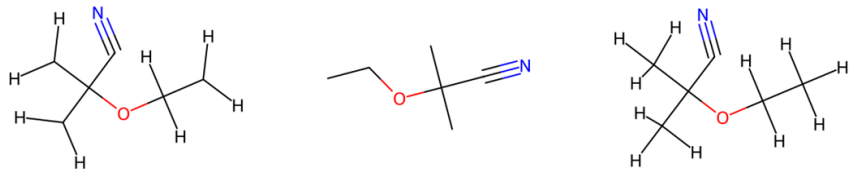

Figure 3.3: Left: generated molecule. Middle: molecule without hydrogen. Right: molecule from dataset.

Comparing the molecules in the left and right of Figure 3.3, it is clear that the generated molecule to the left is missing 3 hydrogen atoms compared with the molecule from the dataset to the right. Two of the hydrogen atoms are missing in the left part of the generated molecule and one hydrogen atom is

missing in the right part. However, removing the hydrogen atoms from both the generated molecule and the molecule from the dataset results in the same molecule, seen in the middle of Figure 3.3.

3.2 Testing Molecule Generation in 3D-space

In this experiment, we evaluate the 3D conformations of the generated molecules using our spatial graph representation, $\mathcal{M} = (V, E, A, D)$. We test both a continuous and categorical version of the angles, A and D, but fix the atom and bond types, V and E, to use categorical diffusion. Additionally, we fix the noise schedule to the cosine schedule. The fixing of the distribution and noise schedule is based on the results from the previous experiment where there is only a small difference between the combinations of distribution and noise schedule. Taking this into account, we refer to the models based on the distribution used for the spatial information, i.e. when reporting the results, categorical refers to the diffusion model using categorical diffusion for the angles.

For evaluation of the 3D conformations of the generated molecules, we use the energy measures and RMSD score from Section 1.1. When calculating the RMSD, we compare the 3D conformation of the generated molecule with the 3D conformation of the energy minimised version of the same generated molecule. We use RDKit with the MMFF94 setting to compute the energy minimised conformations and the energy of the 3D conformations.

3.2.1 Setup

As described in Section 3.1.1, we use $s = 0.008$, $\beta_{min} = 0.0$ and $\beta_{max} = 0.999$ for the cosine noise schedule, and we employ a log soft one-hot encoding when using categorical diffusion. Additionally, we also use QM9 for this experiment. In addition to the atom types, bond types and edge list mentioned in Section 3.1.1, the molecules in QM9 includes Cartesian coordinates in 3D-space. To convert each molecule into our spatial graph representation, $\mathcal{M} = (V, E, A, D)$, we perform the same transformations for V and E, as described in Section 3.1.1, and use the calculations described in Section 1.2.2 to

get A and D from the Cartesian coordinates and the edge list.

3.2.2 Denoising Neural Network

For this experiment, we extend the neural network shown in Figure 3.1 with the angles A and D from our spatial graph representation. This extension is shown in Figure 3.4.

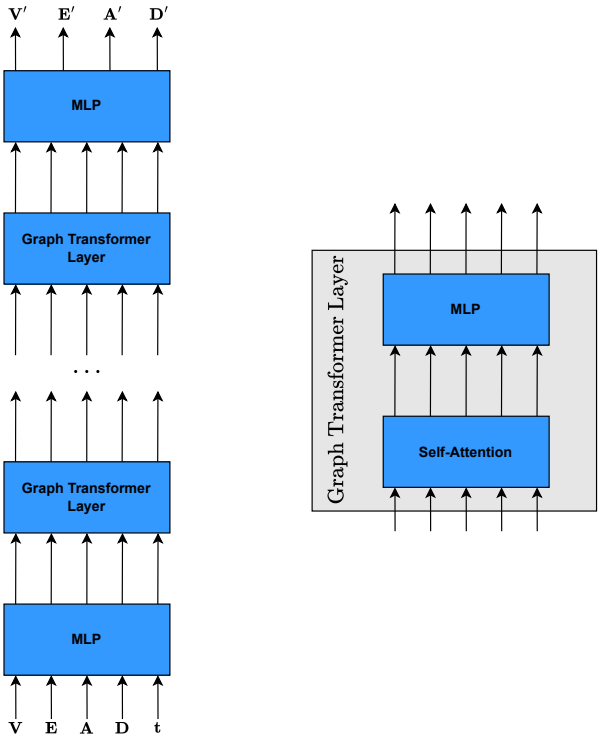

Figure 3.4: Left: Neural network architecture with repeating graph transformer layers. Right: Structure of a graph transformer layer. Both extended with spatial information.

In the neural network in Figure 3.4, we employ MLPs for encoding and decoding the input and output at the beginning and end of the network, and we utilise 6 graph transformer layers in the middle. Similarly, when working with categorical diffusion, we use learnable embeddings for $\mathbf{V}$, $\mathbf{E}$, $\mathbf{A}$, and $\mathbf{D}$ and fixed positional embedding to the timesteps, $\mathbf{t}$, for the encoding MLPs. Likewise, we use softmax at the end of the decoding MLPs.

The extended self-attention block can be seen in Figure 3.5. As shown, the block for updating the timestep embedding is unchanged compared to the block presented in Figure 3.2. Similarly, the updating of the atom and bond types, $\mathbf{V}$ and $\mathbf{E}$, are unchanged and not influenced by the extension with angles. When updating the angles, $\mathbf{A}$ and $\mathbf{D}$, in the self-attention block, we utilise the attention scores from the atom types and bond types. We employ a FiLM module between the current attention scores and the angles, before reducing the dimensions, using the mean, to fit the dimensions of $\mathbf{A}$ and $\mathbf{D}$. Lastly, we flatten the heads of the scores and incorporate the timesteps $\mathbf{t}$ to finish the update of the angles.

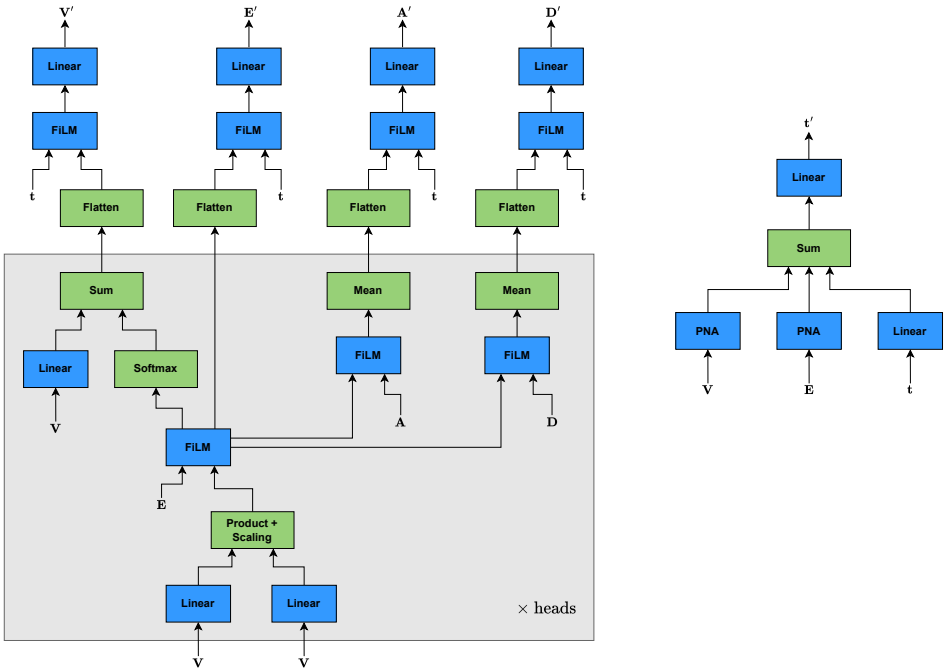

Figure 3.5: Overview of the self-attention block of our graph transformer layer extended with angles.

When using continuous diffusion for the angles, we let the neural network predict the added noise to the angles: $(\mathbf{V}', \mathbf{E}', \mathbf{A}', \mathbf{D}') = (\hat{\mathbf{v}}_0, \hat{\mathbf{e}}_0, \hat{\boldsymbol{\epsilon}}_a, \hat{\boldsymbol{\epsilon}}_d)$. When using categorical diffusion for the angles, we group the angles into intervals of 5 degrees to get the categories and let the neural network predict the clean angles: $(\mathbf{V}', \mathbf{E}', \mathbf{A}', \mathbf{D}') = (\hat{\mathbf{v}}_0, \hat{\mathbf{e}}_0, \hat{\mathbf{a}}_0, \hat{\mathbf{d}}_0)$. The grouping into intervals of 5 degrees results in $K_a = 37$ categories for the triplet angles and $K_d = 74$ categories for the dihedral angles. During generation, we will use the mean of the intervals as the angle.

3.2.3 Results

The validity and uniqueness in Table 3.5 have seen small changes compared to the results in Table 3.1. Additionally, comparing with the novelty scores in Table 3.2, we see a notable improvement. Importantly, there is no notable drop in the measures when adding the spatial information to the graph representation.

Model	$\boldsymbol{V}$	$\boldsymbol{U_v}$	$\boldsymbol{N_u^h}$	$\boldsymbol{N_u}$
Categorical	84.68 ± 0.69	**99.33** ± 0.06	92.21 ± 0.29	**90.23** ± 0.30
Continuous	92.92 ± 0.55	98.42 ± 0.16	**97.42** ± 0.14	**89.83** ± 0.54
Categorical (No H)	**97.99** ± 0.12	97.27 ± 0.13		86.64 ± 0.27
Continuous (No H)	97.52 ± 0.14	97.58 ± 0.19		86.44 ± 0.33

Table 3.5: Validity, uniqueness and novelty scores for our models trained with spatial information. The missing novelty entries is due to the models being trained without hydrogen atoms. Therefore, we cannot compute the novelty with hydrogen atoms. The best results are marked in bold.

In Table 3.6, we present the energy metrics and the RMSD for our models. It can clearly be seen that we do not in general generate 3D conformations that are minimal energy conformations, and we generate 3D conformations with energies several order of magnitudes larger than the dataset. However, when the 3D conformations are energy minimised, we match the dataset more closely. Looking at the RMSD for the generated molecules, only small changes are needed for the generated 3D conformations to be transformed into the energy minimised conformations. Here, we consider small changes equivalent to an RMSD score below 2 angstroms following Ding et al., 2016.

Model	**E**	$\mathbf{E}_{min}$	$\mathbf{E}_{strain}$	**RMSD**
Dataset	33.02 ± 36.68	22.66 ± 34.41	0.61 ± 0.82	0.24 ± 0.23
Categorical	125880 ± 5771	28.66 ± 1.66	7141 ± 322	1.54 ± 0.00
Continuous	61241 ± 5440	22.86 ± 0.75	3407 ± 310	1.31 ± 0.01
Categorical (No H)	79446 ± 9342	13.37 ± 0.43	9175 ± 1164	1.14 ± 0.00
Continuous (No H)	67649 ± 6879	14.13 ± 0.26	7620 ± 783	1.11 ± 0.00

Table 3.6: Energy metrics (kcal/mol) and RMSD (Å) for our models working on graphs with spatial information.

The RMSD also indicates that the models are able to generate good 3D conformations, but considering the energy as well, it is not the general case that the 3D conformations are great. Figure 3.6 shows some of the sensible generated 3D conformations for each of the models, where we see no clashes of atoms, no stretched bonds and low energies. From left to right in Figure 3.6, the energy of the molecules are 137.60, 35.93, 43.46, and 35.56. Similarly, the RMSD score for the molecules are 0.85, 0.36, 0.89, and 0.46.

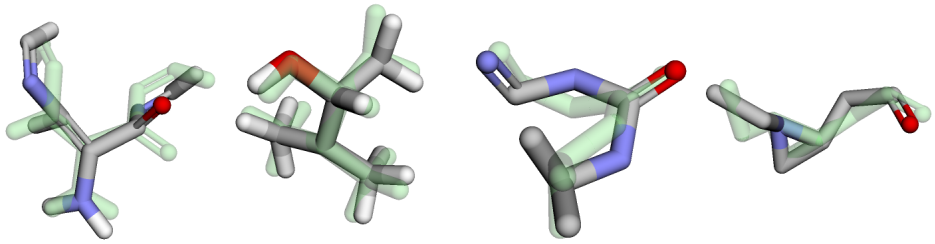

Figure 3.6: Showing examples from the generated molecules with sensible 3D conformations with the energy minimised versions being shown in green. From left to right the generated molecules are from the model: Categorical, Continuous, Categorical (No H), Continuous (No H).

Figure 3.7 shows examples of generated molecules where small errors in the 3D conformations result in very high energies. From left to right in Figure 3.7, the energy of the molecules in millions are 0.12, 3.33, 22.26, and 1.64. Similarly, the RMSD score for the molecules are 1.26, 0.99, 1.57, and 1.44.

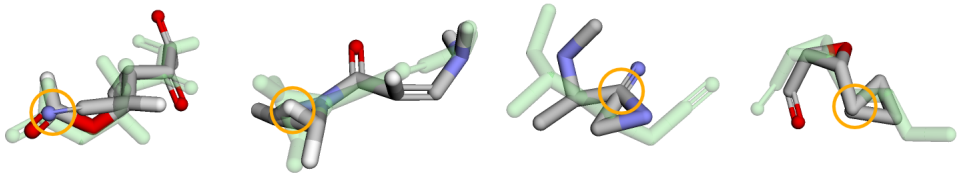

Figure 3.7: Showing examples from the generated molecules where the models generated sensible 3D conformation with only 1-2 small errors resulting in very high energies. The energy minimised versions are shown in green. The small errors are marked with orange circles. From left to right the generated molecules are from the model: Categorical, Continuous, Categorical (No H), Continuous (No H).

Other examples of small errors greatly increasing the energies can be seen in Figure 3.8. These examples show how some molecules have stretched bonds between two atoms while the remaining molecule have a sensible 3D conformation with more sensible bond lengths. This structure is the most commonly found error in the generated molecules. From left to right in Figure 3.8, the energy of the molecules in millions are 0.44, 1.45, 0.34, and 2.08. Similarly, the RMSD score for the molecules are 1.10, 2.10, 1.45, and 1.81.

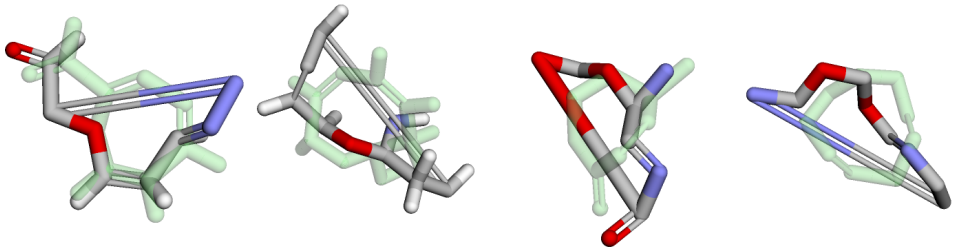

Figure 3.8: Showing examples from the generated molecules where the bond lengths are stretched, with the energy minimised versions being shown in green. From left to right the generated molecules are from the model: Categorical, Continuous, Categorical (No H), Continuous (No H).

Chapter 4

Discussion

In this chapter, we cover some interesting points from our experiments based on the presented results. Additionally, we discuss whether our results show any basis for choosing one distribution over another in the diffusion process. Lastly, we elaborate on several problems revolving around our choice of representation for the spatial information in a molecule and the algorithms we employ with this representation.

4.1 Experiment Results

Usually for a classification task in machine learning, the loss function reflects part of, if not the complete, goal, such that the neural network is in some way knowledgeable about the metrics it will later be evaluated and tested on. However in our case, the goal is to generate valid, unique, and novel molecules, but this goal is not directly reflected in the loss functions in Section 2.4. The loss functions are instead tailored to train the neural network to be a good approximator of the distribution over the dataset. With this in mind, it is assumed that training on a dataset with valid and unique molecules, the denoising neural network will learn to be a good approximator for generating valid and unique molecules. The novelty is then obtained by the randomness of the generation process as explained in Section 2.6. Looking at the results in Section 3.1.3, it can be seen that high scores in the 3 metrics are achievable even with this assumption and not using the metrics in the loss functions directly.

Between the validity, uniqueness and novelty from the first experiment, Table 3.1 and Table 3.2, and the second experiment, Table 3.5, we see an improvement, especially concerning the uniqueness and novelty. This may seem

a bit counter-intuitive, as these metrics only consider the atoms and their connectivity. However, we believe the addition of the spatial information allows the deniosing neural network to better learn valid molecular structures without being tied to generating molecules from the dataset.

Similar to the validity, uniqueness, and novelty scores, the energy and RMSD metrics are not part of the loss functions, when training the diffusion models for the graphs with spatial information. Again, it is assumed that the model will learn to perform well on these metrics as it learns the distribution over the training data. While we saw this for the validity, uniqueness, and novelty metrics, it does not seem to work as well for the energy and RMSD metrics using our spatial graph representation. It could be that integrating the metrics directly in the loss will improve the results, otherwise another representation for the spatial information might be needed.

4.2 Continuous vs. Categorical

In Section 3.1.3, we see the results of using a categorical and continuous distribution in our diffusion model working with molecules as simple graphs. Comparing the results from the two distributions, it seems, based on the validity measure, that using a categorical distribution for discrete features such as atom and bond types is the better choice. However, for our goal of generating valid, unique and novel molecules, the choice between the use of a categorical or continuous distribution is not important. This is clear as using either distribution we generate roughly the same valid, unique, and novel molecules.

An area where the use of a categorical distribution instead of a continuous distribution for the atom and bond types has its benefits, is convergence time. During our experiment presented in Section 3.1, the categorical diffusion models converged after approximately 200-300 epochs, where the continuous diffusion models usually needed approximately 800-900 epochs. This is a notable difference that favours the categorical diffusion models when training time is an important aspect of the task at hand.

In Section 3.2.3, we see the results of generating molecules in 3D-space using our spatial representation, $\mathcal{M} = (V, E, A, D)$ of a molecule. In this experi-

ment, we fixed the distribution of V and E to be categorical, but we tested how using categorical or continuous diffusion on A and D affected the results. Comparing the use of the two distributions for the angles, it seems that the use of continuous diffusion for the angles is slightly better than categorical. At least based on the non-minimised energy and the RMSD scores from Table 3.6. This is probably due to angles being inherently continuous values, and, therefore, the denoising neural network might be able to learn a more precise approximation of the good angles for the 3D conformations.

The difference between the use of continuous and categorical diffusion for the angles might be due to our grouping of angles into intervals of 5 degrees. This grouping looses some precision of the angles for the categorical diffusion, but it is uncertain whether grouping into smaller or larger intervals will improve the generated 3D conformations. The grouping into intervals of 5 degrees was chosen to keep the number of categories relatively small during training. Additionally, training and convergence time is an area where the use of continuous diffusion for the angles is better. Training times for continuous without hydrogen atoms is 3 times faster than categorical without hydrogen atoms, while training times for continuous with hydrogen atoms is 10 times faster than categorical with hydrogen atoms. Moreover, we see convergence for the use of continuous angles at around 200-300 epochs while convergence for categorical angles happens at around 800-900 epochs. Note, for this experiment the atom and bond types are fixed to categorical diffusion and only the distributions for the angles are varied.

Based on the results from our experiments in Section 3.1.3 and Section 3.2.3, it seems that using a distribution for the diffusion process that follows the inherent distribution of an element yields better results. For example, using categorical diffusion for discrete values and continuous diffusion for continuous values. It also seems that doing this results in faster convergence, as described above. The first experiment saw faster convergence with categorical diffusion for the atom and bond types, while the second experiment saw faster convergence with continuous diffusion for the angles and the fixed categorical diffusion for the atom and bond types.

4.3 Spatial Graph Representation

When choosing a representation for molecules in 3D-space, the spatial information can be captured either with external or internal coordinates. For external coordinates, the Cartesian representation is often used, and for internal coordinates, the Z-matrix is often used. In this project, we propose another representation of the internal coordinates, the spatial graph representation. In the Z-matrix only one ordering of the atoms is considered, and the same molecule can be represented by multiple different Z-matrices. The same is true for Cartesian coordinates, where any E(3) transformation on the Cartesian coordinates is still the same molecule, just transformed in 3D-space. Therefore, one needs to make sure that the model is equivariant to E(3) transformations (Guan, Qian, et al., 2023; Hoogeboom, Satorras, et al., 2022). To avoid the problem of needing an equivariant model and only considering one ordering of the atoms, as in a Z-matrix, we propose the spatial graph representation, $\mathcal{M} = (V, E, A, D)$, where for each molecule we consider multiple different triplet angles and dihedral angles. However, our spatial graph representation introduces a new problem by imposing a dependency between E and A, and E and D. These dependencies are contained in the functions IndexA and IndexD. The functions are used to index angles, triplets and dihedral, in a row in either A or D such that a molecule can be constructed in 3D-space from our spatial graph representation. Naturally, a triplet angle consists of 3 atoms, and a dihedral angle consists of 4 atoms, but angles in A and D are retrieved by a single reference atom and the index from either IndexA or IndexD. This means that the functions IndexA and IndexD have to specify an order of the angles, which we do based on E, creating the aforementioned dependencies. To avoid the dependencies, the naive approach would be to use a matrix of size $\mathbb{R}^{N\times N\times N}$ for A and $\mathbb{R}^{N\times N\times N\times N}$ for D. However, the size of such matrices will explode for large molecules, and they would be extremely sparse matrices, as an atom is generally only connected to 1-4 other atoms in a molecule. A different approach would be to investigate if any alternative graph representations exists, where the angles are not represented in rows for a single reference atom. A non-conventional graph representation with angles could be to consider triplets of atoms as the smallest unit instead of atoms. With this representation, we remove the dependencies and do not have an exploding representation as for the naive approach above. However, due to time constraints

of the project, we were unable to experiment with such a representation and will leave that for future works.

When converting from our spatial graph representation to Cartesian coordinates, we first convert our representation to a Z-matrix, and the Z-matrix is converted to Cartesian coordinates using the NeRF method. The Z-matrix only specifies each atom once, each atom has its own row, in relation to 1 other atom with a bond length. This becomes a problem when generating new molecules, as all atom orderings of the generated spatial graphs does not result in a unique 3D conformation of a molecule in a Z-matrix. This is due to the inherent randomness of the generation process using diffusion and the allowed degree of ordering freedom our spatial graph representation has for the angles. In our case, the ordering is based on the Next function, which means changing the ordering inside the Next function would result in different 3D conformations with different energies for the generated molecules. The ordering is especially important when an atom is connected to more than 1 other atom, as when inserting an atom in the Z-matrix, only one of its bond lengths will be represented. As such, the bond lengths of the other bonds can easily be violated by either shrinking or stretching the bonds. This problem is the one we found occurring the most and is illustrated in Figure 3.8. A naive way to counteract this problem would be to change our conversion algorithm to compute the Cartesian coordinates directly from our spatial graphs. However, in such an algorithm, finding a set of Cartesian coordinates that adheres to all the bonds, triplet angles and dihedral angles quickly explodes computationally as the number of atoms in the molecules increases. Additionally, we would like the model to generate 3D conformations that are sensible instead of using a computationally expensive algorithm to check for orderings that result in sensible 3D conformations. Moreover, with such a computationally expensive algorithm, one could use a random starting point instead of the output of a machine learning model, making the model obsolete.

4.4 Neural Network Architecture

Inside the self-attention block of the neural network seen in Figure 3.5, it can be seen that the angles do not influence the atom types, bond types or the timestep embeddings in a forward pass. Normally, every component would influence the computed attention scores in a transformer. However, when conducting experiments with an architecture that used the angles for computing the attention scores and updating the timestep embeddings, every metric during evaluation dropped tremendously. Therefore, we chose to exclude the angles from the attention scores and updating of the timestep embeddings.

The drop in the evaluation metrics, when using the angles in the attention scores, is most likely a result of how we represent triplet angles and dihedral angles. In our spatial graph representation of a molecule in 3D-space, $\mathcal{M} = (V, E, A, D)$, the triplet angles, A, and dihedral angles, D, are represented in rows based on a reference atom. However, in reality A and D are not solely based on a single reference atom as angles are formed by either triples or 4-tuples. To counteract this problem, we would need another representation for the angles, as discussed in Section 4.3.

When we batch the graphs in our diffusion models, we use the largest graph, based on the number of nodes, to determine the size of the matrices. We then pad the matrices for the smaller graphs to fit the dimensions of the largest graph, essentially representing non-existing nodes in the smaller graphs. As such we need to mask these non-existing nodes in the smaller graphs during the diffusion processes and inside the neural network. This is also the case for non-existing edges in our adjacency matrix, E, and the angles in our A and D matrices. As we allow the diffusion process to add and delete edges in E, and we base our representation of the angles, A and D, on our edges, E, we run into masking problems inside our self-attention block found in Figure 3.5. Inside the self-attention block the edges, E, are embedded; therefore, we cannot calculate the number of angles for each reference node and mask the angles accordingly. This makes it impossible for the denoising neural network to know which angles are relevant and which are not. To allow for correct masking of the angles inside the neural network, while keeping the current spatial graph representation, we would need to change the architecture of the neural network to handle the edges separately from the angles.

Chapter 5

Conclusion

In this project, we have shown how different factors affect a diffusion model within the domain of generating molecules using graphs. We have shown how a continuous diffusion model compares to a categorical version, when working with de novo drug design, and how these models are affected by the use of different noise schedules. Using categorical diffusion for de novo drug design following the approach by Hoogeboom, Nielsen, et al., 2021 outperforms the use of continuous diffusion, when representing molecules as simple graphs, by producing similar results in a shorter amount of time. The use of different noise schedules clearly affects the diffusion models, but for this domain the use of one schedule instead of another does not ensure improved results across all reported metrics, i.e. validity, uniqueness and novelty.

In addition to the comparison of a continuous and categorical approach to diffusion, we also present a novel approach to generating 3D conformations of molecules. We introduce a spatial graph representation, $\mathcal{M} = (V, E, A, D)$, that extends the common graph with triplet angles, A, and dihedral angles, D, to capture the molecular internal coordinates. With the use of both continuous and categorical diffusion, we can utilise this spatial graph representation, alongside presented conversion algorithms, to generate new 3D conformations of molecules. During experimentation, we found that the use of continuous diffusion for the angles, A and D, greatly improves training and generation time, but only slightly improves on the presented metrics. Our approach shows great promise for molecule generation in 3D-space, but introduces a few problems inherent to the logic of the representation. To counteract these problems, we would like, for future works, to experiment with different representations for the molecular internal coordinates that could solve these problems either partly or entirely.

Bibliography

Corso, Gabriele et al. (2020). *Principal Neighbourhood Aggregation for Graph Nets*. arXiv: 2004.05718 [cs.LG].

Ding, Yun et al. (Oct. 2016). "Assessing the similarity of ligand binding conformations with the Contact Mode Score". In: *Comput. Biol. Chem.* 64, pp. 403–413.

Guan, Jiaqi, Wesley Wei Qian, et al. (2023). "3D Equivariant Diffusion for Target-Aware Molecule Generation and Affinity Prediction". In: *The Eleventh International Conference on Learning Representations*. URL: https://openreview.net/forum?id=kJqXEPXMsE0.

Guan, Jiaqi, Xiangxin Zhou, et al. (July 2023). "DecompDiff: Diffusion Models with Decomposed Priors for Structure-Based Drug Design". In: *Proceedings of the 40th International Conference on Machine Learning*. Ed. by Andreas Krause et al. Vol. 202. Proceedings of Machine Learning Research. PMLR, pp. 11827–11846. URL: https://proceedings.mlr.press/v202/guan23a.html.

Ho, Jonathan, Ajay Jain, and Pieter Abbeel (2020). *Denoising Diffusion Probabilistic Models*. arXiv: 2006.11239 [cs.LG].

Hoogeboom, Emiel, Didrik Nielsen, et al. (2021). *Argmax Flows and Multinomial Diffusion: Learning Categorical Distributions*. arXiv: 2102.05379 [stat.ML].

Hoogeboom, Emiel, Victor Garcia Satorras, et al. (2022). *Equivariant Diffusion for Molecule Generation in 3D*. arXiv: 2203.17003 [cs.LG].

Kufareva, Irina and Ruben Abagyan (2012). "Methods of protein structure comparison". In: *Methods Mol Biol* 857, pp. 231–257.

Le, Tuan et al. (2023). *Navigating the Design Space of Equivariant Diffusion-Based Generative Models for De Novo 3D Molecule Generation*. arXiv: 2309.17296 [cs.LG].

Leach, Andrew R. (2001). *Molecular Modelling: Principles and Applications*. 2. ed. Essex: Pearson. ISBN: 0582382106.

Nichol, Alex and Prafulla Dhariwal (2021). "Improved Denoising Diffusion Probabilistic Models". In: DOI: 10.48550/arXiv.2102.09672. eprint: arXiv:2102.09672. URL: https://doi.org/10.48550/arXiv.2102.09672.

Parsons, Jerod et al. (July 2005). "Practical Conversion from Torsion Space to Cartesian Space for In Silico Protein Synthesis". In: *Journal of Computational Chemistry* 26, pp. 1063–8. DOI: 10.1002/jcc.20237.

Perez, Ethan et al. (2017). *FiLM: Visual Reasoning with a General Conditioning Layer*. arXiv: 1709.07871 [cs.CV].

Pyykkö, Pekka and Michiko Atsumi (2009). "Molecular Double-Bond Covalent Radii for Elements Li–E112". In: *Chemistry – A European Journal* 15.46, pp. 12770–12779. DOI: https://doi.org/10.1002/chem.200901472. eprint: https://chemistry-europe.onlinelibrary.wiley.com/doi/pdf/10.1002/chem.200901472. URL: https://chemistry-europe.onlinelibrary.wiley.com/doi/abs/10.1002/chem.200901472.

Roskilde Universitet (2009). *Constructing Z-Matrices*. URL: http://thiele.ruc.dk/~spanget/help/g09/c_zmat.htm.

Tong, Jiahui and Suwen Zhao (2021). "Large-Scale Analysis of Bioactive Ligand Conformational Strain Energy by Ab Initio Calculation". In: *Journal of Chemical Information and Modeling* 61.3. PMID: 33630603, pp. 1180–1192. DOI: 10.1021/acs.jcim.0c01197. eprint: https://doi.org/10.1021/acs.jcim.0c01197. URL: https://doi.org/10.1021/acs.jcim.0c01197.

Vaswani, Ashish et al. (2023). *Attention Is All You Need*. arXiv: 1706.03762 [cs.CL].

Vignac, Clement et al. (2023). *DiGress: Discrete Denoising diffusion for graph generation*. arXiv: 2209.14734 [cs.LG].

Wu, Zhenqin et al. (2017). "MoleculeNet: A Benchmark for Molecular Machine Learning". In: *CoRR* abs/1703.00564. arXiv: 1703.00564. URL: http://arxiv.org/abs/1703.00564.