Conditional distribution modelling of mono-metal oxides using diffusion models

Thomas Brun Lau Christensen (DVF755)
Rasmus Pallisgaard (DWP992)

Supervised by Raghavendra Selvin and Frederik Lizak Johansen

June 2023

Contents

1 Introduction

The problem of determining the structure of a metallic nanostructure plays a core part in the development of new appliances and materials[27]. In practice, however, the easiest method of measuring the structure is through X-ray diffraction measurements processed into a pair distribution function (PDF). This measurement is not bijective as each molecular structure maps to a unique PDF, while multiple molecules can map to the same PDF. The problem of finding the true molecular structure of a material given only its PDF then becomes generating a set of candidates and manually selecting the most probable one based on expert knowledge. This project will deal with the molecular structure generating problem of mono-metal oxides, working with simulated structures and PDFs.

We attempt this problem from the angle of generative modelling, attempting to model the distribution of mono-metal oxides conditioned on the PDF. The bijective nature of the reverse of the process of generating the PDF fits well for generative modelling, allowing for multiple predictions in a continuous space. We approach this problem with the specific class of generative machine learning models of diffusion models[10], which have seen success for high quality photo synthesis[23].

The model approximating the true distribution of molecular structures needs to produce high quality samples, in the sense that small adjustments cause huge differences in molecular properties, and it should have mode coverage as we wish to examine all possible candidates when determining the final predicted structure. This makes diffusion models a suitable choice in theory since they are known to possess these two properties, although at the cost of sampling speed as outlined in the generative learning trilemma[29].

This project entails looking at different ways of modelling molecular structures for diffusion models, encompassing both discrete and continuous properties of the structures. We experiment with modelling these structures from the different modelling choices all with the aim of generating molecular structures with a high degree of fidelity.

2 Background

2.1 Mono-metal oxides and pair distribution functions

A mono-metal oxide atom structure is a clustering of atoms usually arranged in a crystal comprising of a single type of metallic atom and oxygen atoms.

Very often these structures are too small to observe and other means of describing mono-metallic structures are required for experimentation. A Pair Distribution Function (PDF) is such method of observing an atomic structure. PDFs (see figure 1) can be thought of as a histogram of the interatomic distances that reside in the structure. A PDF is obtained by Fourier transforming scattering data, meaning the PDF measures the frequency of distances between atoms within the structure. Some concessions are made when computing this transformation, such as not using the entire Q-space from 0 to inf. This specifically causes the PDF to be negative in some places. An important note is that when these measures of interatomic distances are done experimentally, atoms are of varied sizes, vibrate in different frequencies and at different distances, and have electron clouds of varied sizes. A consequence of this is that the PDF, when measured experimentally, is not simply a series of peaks but can have combinations of low, high, wide, and narrow peaks throughout.

Experimentally these PDFs are computed via X-ray diffraction on a sample of the crystal structure. This diffraction produces a plot of intensity peaks in what we call Q-space, and when we compute the PDFs through a Fourier transform, we convert this to real space. These PDFs can furthermore be simulated through different libraries, such as `diffpy`[14]. One notable thing about this PDF computation is that it is not bidirectional - When computing artificially one atomic structure corresponds to a specific PDF, but one PDF can correspond to multiple atomic structures, thus making this computation one way. When plotting we typically note the PDF function as $G(r)$.

The PDF is meant to be interpreted such that for a certain distance along the x-axis, the structure of the graph can tell you a lot about the properties and features of the structure. According to [27]

When comparing two PDFs for similarity we use the R_{wp} measure, which by is computed by:

$$R_{wp} = \sqrt{\frac{\sum_r (G_p(r) - G_t(r))^2}{\sum_r (G_t(r))^2}} \tag{1}$$

where $G_t(r)$ is the target PDF and $G_p(r)$ is the predicted PDF.

2.2 Graph neural networks and equivariance

Graph neural networks are architectures suited for data expressing graph structures. Graph networks are invariant to the same transformations that graphs are, i.e, graph networks are permutation-invariant.

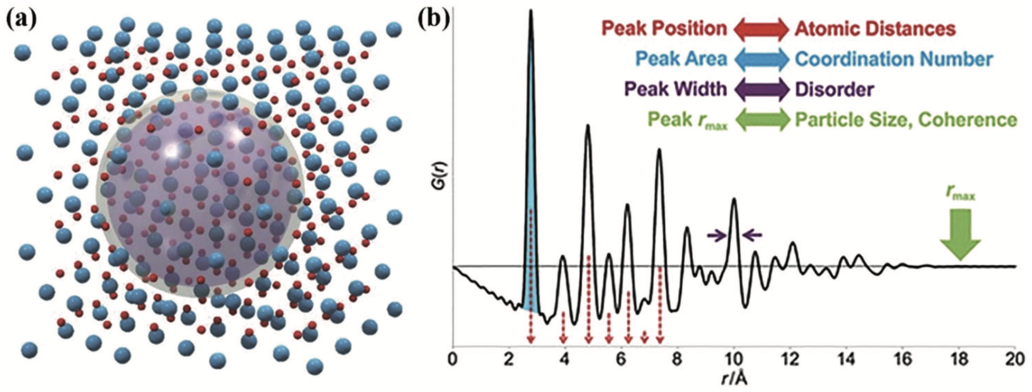


Figure 1: Figure borrowed from [28], page 3. An example of an atomic structure and its corresponding PDF. The peaks indicate distances of atomic pairs and the area under the peak describes the intensity. The width indicates the disorder such as atomic vibration and size difference between the pair of atoms. The peak at the end of the PDF denotes the maximum distance between two atoms and therefore roughly describe the general size of the structure.

A common type of GNN is a graph convolution layer, which we will describe based on notation from [24] whose notation took inspiration from [9]. For a graph $G = (V, E)$ with nodes $v_i \in V$ and edges $e_{ij} \in E$ a graph convolutional layer is defined as

$$\begin{aligned}\mathbf{m}_{ij} &= \phi_e(\mathbf{h}_i^l, \mathbf{h}_j^l, a_{ij}) \\ \mathbf{m}_i &= \sum_{j \in \mathcal{N}(i)} \mathbf{m}_{ij} \\ \mathbf{h}_i^{l+1} &= \phi_h(\mathbf{h}_i, \mathbf{m}_i)\end{aligned}$$

where $\mathbf{h}_i^l \in \mathbb{R}^{\text{nf}}$ is the nf-dimensional embedding of node i at layer l, a_{ij} are edge features, $\mathcal{N}(i)$ is the indices of v_i's neighbors, ϕ_e is an edge operation, and ϕ_h is a node operation. In practice ϕ_e and ϕ_h are modelled with neural networks.

Formally, a function f is equivariant to a transformation g if $T_g(f(x)) = f(S_g(x))$, where T_g, S_g are linear representations associated with the transformation g. A function can be equivariant to a group of transformations $G = \{g_1..g_N\}$ if it is equivariant to all $g \in G$. In essence the function is equivariant to a transformation if the function and transformation commute.

When we model data living in euclidean spaces with underlying graph struc-

tures, the model should be equivariant to 3 transformations. It should be equivariant to translations ($\phi(\mathbf{x}) = \mathbf{X} + \mathbf{c}$), rotations ($\phi(\mathbf{X}) = \mathbf{RX}$ where R is a rotational matrix), and permutations ($\phi(\mathbf{X})$ is a permutation on row-indexes of $\mathbf{X}$), where $\mathbf{X}$ is the data position matrix with row i corresponding to the positional values $\mathbf{x}_i^T$

[24] shows a network architecture that is equivariant to these 3 transformations. For given data $\{(\mathbf{x}_1, \mathbf{h}_1)...(\mathbf{x}_N, \mathbf{h}_N)\}$ where $\mathbf{x}_i \in \mathbb{R}^d$ is the datapoints position in d-dimensional space and $\mathbf{h}_i \in \mathbb{R}^{\text{nf}}$ is a corresponding f-dimensional feature vector. The equivariant graph convolutional layer is defined as follows

$$\mathbf{m}_{ij} = \phi_e(\mathbf{h}_i^l, \mathbf{h}_j^l, ||\mathbf{x}_i^l - \mathbf{x}_j^l||^2, a_{ij}) \tag{2}$$

$$\mathbf{x}_i^{l+1} = \mathbf{x}_i^l + C \sum_{j \neq i} (\mathbf{x}_i^l - \mathbf{x}_j^l) \phi_x(\mathbf{m}_{ij}) \tag{3}$$

$$\mathbf{m}_i = \sum_{j \neq i} \mathbf{m}_{ij} \tag{4}$$

$$\mathbf{h}_i^{l+1} = \phi_h(\mathbf{h}_i, \mathbf{m}_i) \tag{5}$$

Where C a normalization constant set to $1/N$. Since the edges of the graph structure are not explicit, the equivariant graph convolutional layer infers the edges. This is done by using information about the positions of any pair of nodes in the form of the distance in the edge update ϕ_e, as well adding the attention network $\phi_x : \mathbb{R}^{\text{nf}} \to \mathbb{R}$ to determine how much the position update of node i should depend on all other nodes in the graph.

2.3 Denoising diffusion modelling

Unless other sources are explicitly referred, the following is based heavily on [17]. Denoising diffusion modelling is a generative modeling approach that seeks to learn a true data distribution $p(x)$ of some data x for the purpose of generating samples $x \sim p(x)$. In this paper we concern ourselves with learning true distributions through maximizing the likelihood of the observed data, often via maximizing $\log p(x)$.

2.3.1 Evidence Lower Bound

For diffusion models we learn this true distribution by learning an associated latent or unseen random variable z, often of smaller size or simpler structure than the true data distribution. If we model the latent variables z and the true data x as a joint distribution $p(x, z)$, and then either marginalizing out

the latent variable z:

$$p(x) = \int p(x, z)\, dz \tag{6}$$

or use the chain rule of probability:

$$p(x) = \frac{p(x, z)}{p(z|x)} \tag{7}$$

Using both is intractable as we either need to integrate over possibly overly complex models $p(x, z)$ or we need to know the ground truth posterior encoder $p(z|x)$. We can however derive called the evidence lower bound, or ELBO for short:

$$\log p(x) = \mathbb{E}_{q_\theta(z|x)}\left[\log \frac{p(x, z)}{q_\theta(z|x)}\right] + D_{\mathrm{KL}}\left(q_\theta(z|x) \parallel p(z|x)\right) \tag{8}$$

$$\geq \mathbb{E}_{q_\theta(z|x)}\left[\log \frac{p(x, z)}{q_\theta(z|x)}\right] \tag{9}$$

Where $q_\theta(z|x)$ is an approximate to the true posterior $p(z|x)$ with parameters θ which we then seek to optimize. We refer to [17] for the full derivation of equation 8, and it is easy to see the derivation of inequality 9 as the KL-divergence is non-negative. We denote the RHS of inequality 9 the ELBO, and we use the ELBO as an optimization objective because computing the KL-divergence of $q_\theta(z|x)$ and $p(z|x)$ is intractable, as we still do not know the true posterior $p(z|x)$. Furthermore, we know $p(x)$ is computed by marginalizing out all the latent variables from the joint distribution $p(x, z)$, and thus the likelihood term is a constant regardless of our choice of parameters θ. Regardless of choice of θ, the sum of the ELBO and KL divergence terms sum to the same constant, and maximizing ELBO results in a minimization of the KL divergence term. Thus, the ELBO can be used as a direct proxy for the maximization of the likelihood, since the more we optimize ELBO, the closer $q_\theta(z|x)$ gets to $p(z|x)$.

2.3.2 Diffusion Models

Denoising diffusion models seek to maximize the likelihood of the observed data $p(x)$. It models some data $x = z_0$ by repeatedly computing an intermediate latent distribution $q(z_t|z_{t-1})$ T times such that z_T comes from a predefined $p(z_T)$. In practice we pick T large enough such that we can safely assume that $q(z_T|z_0)$ matches $p(z_T)$. In a diffusion model this distribution is not learned but predefined. We instead learn a diffusion process $p_\theta(z_t|z_{t+1})$ that we use to iteratively compute closer and closer recreations

of the original data sample. For both types of distributions, we assume that they are Markovian. At sampling time, we can then sample from $p(z_T)$ and use $p_\theta(z_t|z_{t+1})$ sample $x = z_0$. For a diffusion model we furthermore let the latent space be of the same dimensionality as the data space.

In terms of notation, we let $z_t = x_t$. We then say that at time $t = 0$, x_0 is the original data, and x_t for $t \in [1, T]$ is any of the T latent variables indexed by x_t. We can now represent the joint distribution and the posterior as:

$$p(x_{0:T}) = p(x_0, x_{1:T}) = p(x_T) \prod_{t=1}^{T} p_\theta(x_{t-1}|x_t) \tag{10}$$

$$q(x_{1:T}|x_0) = \prod_{t=1}^{T} q(x_t|x_{t-1}) \tag{11}$$

Using these we can now derive an ELBO for our diffusion model:

$$\begin{aligned} \log p(x) \geq \mathbb{E}_{q(x_1|x_0)} \left[\log p_\theta(x_0|x_1)\right] - D_{\mathrm{KL}} \left(q(x_T|x_0) \parallel p(x_T)\right) \\ - \sum_{t=2}^{T-1} \mathbb{E}_{q(x_t|x_0)} \left[D_{\mathrm{KL}} \left(q(x_{t-1}|x_t, x_0) \parallel p_\theta(x_{t-1}|x_t)\right)\right] \end{aligned} \tag{12}$$

We refer to [17] for this derivation. These three terms are called the *reconstruction term*, the *prior matching term*, and the *denoising matching term* respectively. The *reconstruction term* maximizes the log probability of the original data sample given the first latent variable. This can be optimized using Monte Carlo estimation. The *prior matching term minimizes* the KL divergence between the final latent distribution and the preset prior $p(x_T)$. Under our assumptions this KL divergence is zero and does not need to be considered. The *denoising matching term* minimizes the KL divergence between our learned denoising distribution $p_\theta(x_{t-1}|x_t)$ and a tractable, ground truth denoising distribution $q(x_{t-1}|x_t, x_0)$ (also called the forward process). This can be a ground truth since the distribution is conditioned on the original data sample x_0. Since we do not optimize over the prior matching term, the denoising matching term dominates the complexity due to the difficulty of computing the KL divergence of arbitrarily complex distributions. The next two sections cover ways to handle this optimization difficulty.

2.4 Gaussian diffusion modelling

A way of making this intractable KL Divergence computation computable is first to let the pre-defined encoder distribution be gaussian centered around

the output of the previous timestep, i.e.

$$q(x_t|x_{t-1}) = \mathcal{N}(x_t; \sqrt{\alpha_t}x_{t-1}, (1-\alpha_t)\mathbf{I}) \tag{13}$$

where α_t can be either fixed or learned. It is standard to define $\sigma_t^2 = \beta_t = 1 - \alpha_t$. In this project we have chosen to fix these using specific schedules (more on this in section 2.6). Furthermore we let the latent distribution $p(x_T)$ be a standard normal:

$$p(x_T) = \mathcal{N}(x_T; 0, \mathbf{I}) \tag{14}$$

By restricting these distributions we can derive tractable optimisations. By Bayes rule:

$$q(x_{t-1}|x_t) = q(x_{t-1}|x_t, x_0) = \frac{q(x_t|x_{t-1}, x_0)q(x_{t-1}|x_0}{q(x_t|x_0)} \tag{15}$$

By using the reparameterisation trick, a sample from $q(x_t|x_{t-1})$ can be rewritten as

$$x_t = \sqrt{\alpha_t}x_{t-1} + \sqrt{1-\alpha_t}\epsilon, \quad \epsilon \sim \mathcal{N}(\epsilon; 0, \mathbf{I}) \tag{16}$$

using the reparametrisation trick again on $x_{t-1}, x_{t-2}, ..$ we can recursively find a sample $x_t \sim q(x_t|x_0)$ as:

$$\begin{aligned} x_t &= \sqrt{\alpha_t}x_{t-1} + \sqrt{1-\alpha_t}\epsilon && (17)\\ &= \sqrt{\prod_{i=1}^{t}\alpha_i}x_0 + \sqrt{1-\prod_{i=1}^{t}\alpha_i}\epsilon_0 && (18)\\ &= \sqrt{\bar{\alpha}_t}x_0 + \sqrt{1-\bar{\alpha}_t}\epsilon_0 && (19)\\ &\sim \mathcal{N}(x_t; \sqrt{\bar{\alpha}_t}x_0, (a-\bar{\alpha}_t)\mathbf{I}) && (20) \end{aligned}$$

Where $\bar{\alpha}_t = \prod_{i=1}^{t}\alpha_i$. By equation 15 we have:

$$q(x_{t-1}|x_t, x_0) \propto \mathcal{N}\left(x_{t-1}; \mu_q(x_t, x_0), \Sigma_q(t)\right) \tag{21}$$

Where

$$\mu_q(x_t, x_0) = \frac{\sqrt{a_t}(1-\bar{\alpha}_{t-1}x_t) + \sqrt{\bar{\alpha}_{t-1}}(1-\alpha_t)x_0}{1-\bar{\alpha}_t} \tag{22}$$

$$\Sigma_q(t) = \frac{(1-\alpha_t)(1-\bar{\alpha}_{t-1})}{1-\bar{\alpha}_t} \tag{23}$$

Again, referring to [17] for the full derivation. Note that Σ_t has no learnable parameters as we choose a fixed α_t. Since we wish to learn a distribution

$p_\theta(x_{t-1}|x_t)$ that matches $q(x_{t-1}|x_1, x_0)$, we also let $p_\theta(x_{t-1}|x_t)$ be gaussian. By [17] we can then compute

$$\arg\min_\theta D_{\mathrm{KL}}\left(q(x_{t-1}|x_t, x_0) \parallel p_\theta(x_{t-1}|x_t)\right) \tag{24}$$

$$= \arg\min_\theta \frac{1}{2\sigma_q^2(t)}\left[\|\mu_q - \mu_\theta\|_2^2\right] \tag{25}$$

Where $\sigma_q^2(t) = \frac{(1-\alpha_t)(1-\bar{\alpha}_{t-1})}{1-\bar{\alpha}_t}$, $\Sigma_q(t) = \sigma_q^2(t)\mathbf{I}$ and $\mu_\theta = \mu_\theta(x_t, x_0), \mu_q = \mu_q(x_t, t)$. Since Σ_t is fixed we can learn the distribution simply by learning a model $\hat{x}_\theta(x_t, t)$ that learns to predict x_0 given an example x_t at time point t. Using this we can compute:

$$\mu_\theta(x_t, x_0) = \frac{\sqrt{a_t}(1-\bar{\alpha}_{t-1}x_t) + \sqrt{\bar{\alpha}_{t-1}}(1-\alpha_t)\hat{x}_\theta(x_t, t)}{1-\bar{\alpha}_t} \tag{26}$$

by [10], if we isolate x_0 from equation 17-20 and insert it into equation 26 we get

$$= \arg\min_\theta \frac{1}{2\sigma_q^2(t)}\left[\left\|\frac{1}{\sqrt{\bar{\alpha}_t}}\left(x_t(x_0, \epsilon) - \frac{\beta_t}{\sqrt{1-\bar{\alpha}_t}}\epsilon\right) - \mu_\theta(x_t(x_0, \epsilon), t)\right\|_2^2\right] \tag{27}$$

As shown in [10], since μ_θ has to predict $\frac{1}{\sqrt{\bar{\alpha}_t}}\left(x_t - \frac{\beta_t}{\sqrt{1-\bar{\alpha}_t}}\epsilon\right)$ given x_t, and x_t is available as input already, we can rewrite this objective as

$$\mu_\theta(x_t, t) = \frac{1}{\sqrt{\bar{\alpha}_t}}\left(x_t - \frac{\beta_t}{\sqrt{1-\bar{\alpha}_t}}\epsilon_\theta(x_t, t)\right) \tag{28}$$

And instead seek to predict the noise ϵ that was previously added to get from x_{t-1} to x_t. With this plus the finding of [10] that dropping the constant from the denoising matching term computed in equation 27 yields:

$$\arg\min_\theta \left[\left\|\epsilon - \epsilon_\theta(\sqrt{\bar{\alpha}_t}x_0 + \sqrt{1-\bar{\alpha}_t}\epsilon, t)\right\|_2^2\right] \tag{29}$$

Which is computed for every $t = 1$ to T. This can be done through Monte Carlo estimation and yields:

$$\mathbb{E}_{t,x_0,\epsilon}\left[\left\|\epsilon - \epsilon_\theta(\sqrt{\bar{\alpha}_t}x_0 + \sqrt{1-\bar{\alpha}_t}\epsilon, t)\right\|_2^2\right] \tag{30}$$

where x_0 is sampled from data, $\epsilon \sim \mathcal{N}(\epsilon; 0, \mathbf{I})$, and t is sampled uniformly.

2.5 Discrete diffusion

We use the method of discrete state spaces for diffusion models presented in [1]. Let each $x_t, 0 \leq t \leq T$ be a one hot encoding vector of a discrete state. We can then represent the transition probability matrix by $[\mathbf{Q}_t]_{ij} = q(x_t = j|x_{t-1} = i)$. Then we can write $q(x_t|x_{t-1})$ as the categorical distribution

$$q(x_t|x_{t-1}) = \text{Cat}\left(x_t; x_{t-1}\mathbf{Q}_t\right) \tag{31}$$

with the probability of changing from a certain state x_{t-1} to x_t has probabilities $p = x_{t-1}\mathbf{Q}_t$ where p is a vector describing the probability of transitioning to each state. With this we define

$$q(x_t|x_0) = \text{Cat}\left(x_t; x_0\overline{\mathbf{Q}}_t\right), \quad \overline{\mathbf{Q}}_t = \mathbf{Q}_1\mathbf{Q}_2...\mathbf{Q}_t \tag{32}$$

$$q(x_{t-1}|x_t, x_0) = \frac{q(x_t|x_{t-1}, x_0)q(x_{t-1}|x_0)}{q(x_t|x_0)} = \text{Cat}(x_{t-1}; \frac{x_t\mathbf{Q}_t^\top \odot x_0\overline{\mathbf{Q}}_{t-1}}{x_0\overline{\mathbf{Q}}_t x_t^\top}) \tag{33}$$

By the Markov assumption on q, we have that $q(x_t|x_{t-1}, x_0) = q(x_t|x_{t-1})$. Constructing $p_\theta(x_{t-1}|x_t)$ in an equivalent manner, the KL-divergence from the denoising matching term can be computed by summing over all values of the variables x_t, x_{t-1}.

The transition probability matrix can take many forms, one of which we highlight from [1], constructs a transition matrix

$$\mathbf{Q}_t = (1 - \beta_t)\mathbf{I} + \beta_t/K, \tag{34}$$

for which the stationary distribution of this is uniform.

[26] Extends this discrete diffusion theory to construct their DiGress model - a discrete graph diffusion model. For a graph $G = (V, E)$, we use $x_i \in \{0, 1\}^a$ to represent a one hot feature vector for the vertex $i \in V, |V| = n$, with these being organised in a matrix $\mathbf{X} \in \{0, 1\}^{n\times a}$. The one hot feature of an edge $e_{ij} \in \{0, 1\}^b$ for each edge $(i, j) \in E$ can similarly be stored in a tensor $\mathbf{E} \in \{0, 1\}^{n\times n\times b}$. Furthermore $G_t = (\mathbf{X}_t, \mathbf{E}_t)$ is the graph produced at timestep t by adding noise. We define the transition matrices

$$[\mathbf{Q}_t^{\mathbf{X}}]_{ij} = q(x_t = j|x_{t-1} = i), \quad [\mathbf{Q}_t^{\mathbf{E}}]_{ij} = q(e_t = j|e_{t-1} = i) \tag{35}$$

With this we can define the forward process as

$$q(G_t|G_{t-1}) = \left(\mathbf{X}_{t-1}\mathbf{Q}_t^{\mathbf{X}}, \mathbf{E}_{t-1}\mathbf{Q}_t^{\mathbf{E}}\right) \tag{36}$$

$$q(G_t|G_0) = \left(\mathbf{X}_0\overline{\mathbf{Q}}_t^{\mathbf{X}}, \mathbf{E}_0\overline{\mathbf{Q}}_t^{\mathbf{E}}\right) \tag{37}$$

where $\overline{\mathbf{Q}}_t^{\mathbf{X}} = \mathbf{Q}_1^{\mathbf{X}}\mathbf{Q}_2^{\mathbf{X}}...\mathbf{Q}_t^{\mathbf{X}}$, $\overline{\mathbf{Q}}_t^{\mathbf{E}} = \mathbf{Q}_1^{\mathbf{E}}\mathbf{Q}_2^{\mathbf{E}}...\mathbf{Q}_t^{\mathbf{E}}$. We thus use the discrete diffusion method explained previously on the vertices and edges separately to compute the new vertex and edge features and then update the graph structure.

The DiGress architecture seeks to learn a prediction network ϕ_θ parametrized by θ that seeks to predict the original graph G_0 given G_t. This network outputs transition probabilities $\hat{p}_G = (\hat{p}^X, \hat{p}^E)$, and we compute the loss via the cross-entropy loss between the predicted transition probabilities and the true labels x_i, e_{ij}:

$$\mathcal{L}(\hat{p}, G) = \sum_{1\leq i\leq n} \text{cross-entropy}(x_i, \hat{p}_i^X) + \lambda \sum_{1\leq i,j\leq n} \text{cross-entropy}(e_{ij}, \hat{p}_{ij}^E) \quad (38)$$

where $\lambda \in \mathbb{R}^+$ weights the importance of vertices vs edges. Once the network is trained, we can estimate $p_\theta(G_{t-1}|G_t)$ via:

$$p_\theta(G_{t-1}|G_t) = \prod_{1\leq i\leq n} p_\theta(x_{i,t-1}|G_t) \prod_{1\leq i,j\leq n} p_\theta(e_{ij,t-1}|G_t) \quad (39)$$

$$p_\theta(x_{i,t-1}|G_t) = \sum_{x\in\mathcal{X}} p_\theta(x_{i,t-1}|x_i = x, G_t)\hat{p}_i^X(x) \quad (40)$$

$$p_\theta(x_{i,t-1}|x_i = x, G_t) = \begin{cases} q(x_{i,t-1}|x_i = x, x_{i,t}) & \text{if } q(x_{i,t}|x_i = x) > 0 \\ 0 & \text{otherwise} \end{cases} \quad (41)$$

With equation 41 having been chosen heuristically, and where $x_{i,t}$ is the ith vertex feature vector at timestep t. Definitions for e_{ij} are similar to those in equation 40 and 41. Although we will not cover this in detail, [26] shows that DiGress is permutation equivariant, which is a property that is particularly important for graph data, since many permutations of vertex and edge matrices can represent the same matrix.

2.6 Noise schedules

How we schedule α_t and β_t is called noise scheduling. [10] used a method of linear noise scheduling where we set the variance to linearly increase from $\beta_1 = 10^{-4}$ to $\beta_T = 0.02$. [18] found that the linear scheduling results in the latter 20% of the forward process not contributing to sample size due to $\bar{\alpha}_t$ becoming exceedingly small. They proposed the cosine noise schedule:

$$\bar{\alpha}_t = \frac{f(t)}{f(0)},\ f(t) = \cos\left(\frac{t/T + s}{1+s} \cdot \frac{\pi}{2}\right)^2,\ \beta_t = 1 - \frac{\bar{\alpha}_t}{\bar{\alpha}_{t-1}} \quad (42)$$

Which leads to a scheduling of $\bar{\alpha}_t$ that is linear in the middle and changes little at the extremes. In practice they recommend we clip this schedule at between 0.001 and 0.999.

2.7 Conditional diffusion modelling

The diffusion modelling described above only models the distribution $p(x_0)$, while we aim to model a conditional distribution $p(x_0|y)$ in this project. To model a conditional distribution, we condition the transition steps on the conditioning vector y as well

$$p(x_{0:T}|y) = p(x_0, x_{1:T}|y) = p(x_T) \prod_{t=1}^{T} p_\phi(x_{t-1}|x_t, y)$$

In practice, this is done by concatenating y to the input of the denoising network. Guidance is introduced to explicitly control the amount of weight the conditioning plays in training the diffusion model. Guidance comes in 2 varieties, classifier guidance and classifier-free guidance. In classifier guidance we first observe that modelling a conditional distribution allows us to rewrite the gradients of the log likelihood as

$$\begin{aligned}
\nabla_x \log p(x|y) &= \nabla_x \log \left(\frac{p(x)p(y|x)}{p(y)} \right) && (43)\\
&= \nabla_x \log p(x) + \nabla_x \log p(y|x) - \nabla_x \log p(y) && (44)\\
&= \nabla_x \log p(x) + \nabla_x \log p(y|x) && (45)
\end{aligned}$$

The term $\nabla_x \log p(x)$ is the standard likelihood while $\nabla_x \log p(y|x)$ is the adversarial gradient. Training using classifier guidance can thus be done by training an unconditional model as usual with the addition of y concatenated to the input of the network, as well as adding a classifier model $p(y|x)$.

We can omit the need of training another model in parallel with our denoising network by using classifier-free guidance. Here we instead rewrite the gradients of the log-likelihood as follows.

$$\begin{aligned}
\nabla_x \log p(x|y) &= \nabla_x \log p(x) + \gamma \left(\nabla_x \log p(x|y) - \nabla_x \log p(x|y) \right) && (46)\\
&= (1-\gamma) \nabla_x \log p(x) + \gamma \nabla_x \log p(x|y) && (47)
\end{aligned}$$

equation 46 use the derivation from equation 45. This way we still guide the training process by explicitly setting the term γ to represent how much we

want to weigh the conditional information, without having to train a separate classifier model. With $\gamma = 0$ indicating we want to learn the unconditional distribution, and $\gamma = 1$ indicating we only want to model the conditional distribution. With $\gamma > 1$ we indicate we explicitly only want to generate samples with a conditioning, penalizing any samples that could not have been generated without.

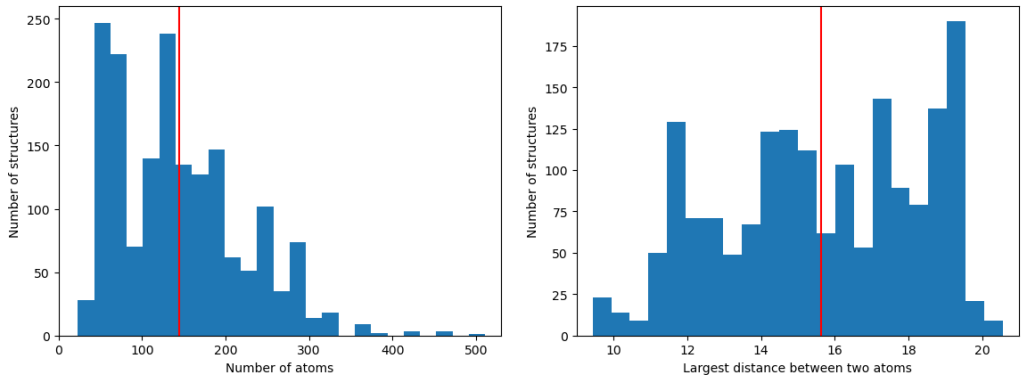
3 Data

Our dataset comprises of a series of mono-metal oxides - atomic structures encoded in the HDF5 file format. Each file contains a structure encoded as a graph. For each atom in the structure, the following node attributes are included: atomic number, atomic radius, atomic density, electron affinity, x-coordinate, y-coordinate, and z-coordinate. The edges in the graph are encoded as a sparse adjacency matrix, and for each edge we are given the distance of the edge as an edge attribute. Furthermore, for each structure we are also given an r vector of the distances in Ångstrøm and a PDF vector containing the PDF values for each distance in the r vector. These two vectors can be used to plot a PDF. All these structures are known in chemistry. However, all PDFs have been artificially generated.

The dataset contains 1728 structures, and the number of atoms ranges from 23 atoms to 511 with an average atom count of 143, a standard deviation of 76.42, and a mode of 135 atoms. The size of the structures ranges from 9.49 to 20.57. The average size of the structures is 15.64 Ånsgtrøm, with a standard deviation of 2.72, meaning that while the number of atoms have large deviations, the sizes of the structures do not deviate much.

In figure 2 we see two histograms of the number of atoms in each structure and the largest distance between two atoms in each structure respectively, with the latter also noted as the true structure size. Looking at the distribution of the number of atoms in a structure we see a wide range of values, although most of the structures are between 50 and 200 atoms in size. Generally, we see a large spread in structure size, which will be relevant later on during experimentation. For the size of the structures, we see that they are much more evenly distributed, with almost all lying within the span of 10-20 Ångstrøm. Interestingly this shows that for these structures the number of atoms and the general size of the structure does not necessarily correlate.

The dataset has a mixture of discrete and continuous data, and for our modelling purpose we can disregard some of the information available, as



(a) Histogram of the number of atoms in each structure. The mean number of atoms in each structure is 76.42.

(b) Histogram of the largest distance between two atoms in each structure. The mean interatomic distance is 15.64.

Figure 2: Histograms of statistics on the dataset. Features histograms of the number of atoms in each structure and the largest distance between two atoms, which we also denote as the structure size, as mentioned in section 2.1. The red lines indicate the mean value in the dataset.

these features are not relevant or related to the problem. Our main goal is to construct a model that produces graphs using both continuous data (coordinates) and discrete data (atomic numbers), and thus our work deals with how one can construct diffusion models that deal with this type of data.

4 Distance matrix modelling

4.1 Method

As mentioned previously, the general problem is that of generating molecules that would produce a given PDF. Since the PDF can coarsely be thought of as a histogram of interatomic distances in the molecule, we experiment with modelling the molecules by their distance matrix. Formally, for a molecule given by the coordinates of all its atoms $\mathbf{X} = \left[\mathbf{x}_1^\top ... \mathbf{x}_N^\top\right]^\top, \mathbf{x}_i \in \mathbb{R}^3, i = 1...N$ we define its distance matrix as $\mathbf{D} = [d_{ij}], d_{ij} = ||\mathbf{x}_i - \mathbf{x}_j||^2$. Our method then consists of generating molecules by their distance matrix via a Gaussian diffusion process by considering $\mathbf{D} = x_0$.

Notice here that this formulation does not entail any features about the individual nodes (e.g. their atom type), but only information about their

relative distances. This is to simplify the problem to get a simpler model to work and to learn about the difficulties of the problem before working on the full problem in all its complexity.

The following are 4 properties of any metric, and thus by extension euclidean distance, which needs to hold for the entries in our distance matrix. We denote the metric d with objects x, y, z

$$
\begin{aligned}
&1\text{: } d(x,y) = d(y,x) \text{ (symmetry)} \\
&2\text{: } d(x,y) = 0 \implies x = y \\
&3\text{: } d(x,y) \geq 0 \text{ (Positivity)} \\
&4\text{: } d(x,z) \leq d(x,y) + d(y,z) \text{ (Triangle inequality)}
\end{aligned}
$$

Our Gaussian diffusion process with prior $p(x_T) = \mathcal{N}(x_T; 0, \mathbf{I})$ cannot guarantee that the distance values in any $x_t, t \in 0...T$ satisfies any of these properties, necessitating some further simplifications. We start by considering only the strict upper triangular matrix, i.e., the upper triangular with 0-entries at the diagonal. This can easily be converted from the full distance matrix $\mathbf{D}$ to the strict upper triangular $\mathbf{U}$ and convert it back by $\mathbf{U} + \mathbf{U}^\top$.

This simplification reduces the input space by $N(N-1)/2$, and now allows x_t to preserve property 1 by only modelling each distance once. We partly preserve property 2 by never modelling the values in the diagonal and always letting them be 0, but this only ensures that if $x = y$ then $d(x, y) = 0$, but there can in theory still exist pairs $x \neq y$ where $d(x, y) = 0$.

For the purposes of visualization and simulation of PDF's, we construct a pipeline from the distance matrix representation to a point cloud representation by first linearly re-scaling all distance values into a predefined but experiment specific, and positive range, then point coordinates of the nodes are constructed using multidimensional scaling (MDS)[3], then lastly the whole point cloud is rotated to lie on its maximal variance axis using PCA (Principal Component Analysis).

We use the MDS formulated in [16] which frames it as an optimization problem to find the best fit. This alleviates the need for precise distance metrics for the MDS to find a solution. This formulation also does not strictly require the given 'distances' to adhere to the metric properties described above, making it a good fit for our problem. We will discuss the consequences of this later.

The denoising function is modelled by a neural network utilizing self-attention. Loosely inspired by transformer architectures described in [25]. We use a series of self-attention layers. Before the self-attention layers the strict upper

triangular matrix is extracted and flattened such that the input is now a sequence of edge embeddings. The full architecture can be found in A.1.

The choice the self-attention mechanisms are done for 2 main reasons. First, it ensures the permutation equivariance in our whole network since the attention mechanism is inherently permutation equivariant. Second, the self-attention mechanism makes it easier to model molecules of varying sizes. In a sense we are training on sequences of edge embeddings utilizing neural networks originally designed for training on sequences of text embeddings. The network structure is also conditioned on the time step value passed through a sinusoidal embedding layer and then used in each self-attention layer.

4.2 Experiments

Our approach to experimenting with this modeling choice is to start simple and add complexity after each successful step. To that end, the first experiment is to see if we can overfit a single sample.

In this setup we single out one of the smallest and therefore simplest molecular structures in our dataset, train only on that sample. We expect this to simplify the learning problem to such a degree that our model can overfit and be able to generate only this single sample. The success criteria for this experiment is to be able to sample the specific sample again from the model after it has completed training. For visualization of this sample, we first compute the minimum and maximum distance of the original noise-free sample and use that for the scale during the visualization pipeline.

We choose $T = 1000$ as done in [10] and $\gamma = 0.9$ as explored in [11]. We implement and train the network using PyTorch[19] and PyTorch Lightning[8]. For the multidimensional scaling we use Scikit-Learn's implementation[20]. The complete network architecture and training hyperparameters can be found in appendix A.1.

We evaluate the model by examining the generated samples. For illustration of the reverse diffusion process, x_{1000}, x_{500}, and x_0 (the reconstruction), will be plotted alongside the original sample. Since we can examine the level of structure though the histograms of the positions and relative distances in the molecule, histograms throughout both the forward and the backwards process are plotted. We expect a good sample to have a high correlation between its relative distance histogram and its simulated PDF. For the diffusion process the noise is fixed for all $t = 0...T$, while it is not for the reverse process. To reduce the amount of computation we only compute the positions of the atom and its histograms at $t = 0, 5, 10...T$ for the forward process and

$t = T, T-5, T-10...0$ for the backwards process. The results can be seen in figure 3.

We see from the results in figure 3 that the model does not learn to reconstruct the structure of the original sample very well. We see in figures 3e and 3f that the pipeline that transforms the distance matrix to some set of coordinates is not stable with regards to the diffusion process, frequent jumps in rotation and positions gives these large periodic variations in the forward process. This example was picked, but other examined examples show the same behavior.

5 Coordinate modelling

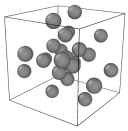
5.1 Method

Given how the distance model didn't perform very well and underlying issues with the model not adhering to the assumptions of euclidean space, we conduct experiments with the molecular structures as point clouds, wherein each atom is represented by a set of coordinates in Cartesian space with an associated node feature vector. This follows the formulation described in section 2.2 wherein a single structure from our dataset consists of N pairs of coordinates and feature vectors $\{(\mathbf{x}_1, \mathbf{h}_1)...(\mathbf{x}_N, \mathbf{h}_N)\}$ where N is specific to the sample. To simplify this approach even further we do not consider the node feature vectors in our dataset, letting oxygen and metallic atoms be represented the same.

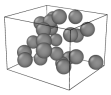
For this approach we choose the standard diffusion model with prior $p(x_T) = \mathcal{N}(x_T; 0, \mathbf{I})$ where any $x_t, t \in 0...T$ are the positions of all atoms in the molecule. We immediately see that the equivariances explained in section 2.2 is needed since we are working with positional data that has a strong underlying graph structure.

Compared to the distance matrix modelling method, this approach does not directly model the properties we are interested in generating with respect to computing a simulated PDF. This formulation allows us to model the positions directly without the need for compromising on required metric properties, as well allowing us to visualize the noising and denoising process more naturally.

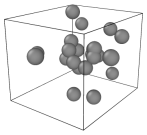
Our setup of the data mirrors [12] exactly except that we omit the node features, and we use the same network architecture as described in section 2.2.



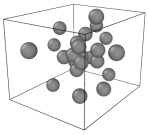
(a) Original



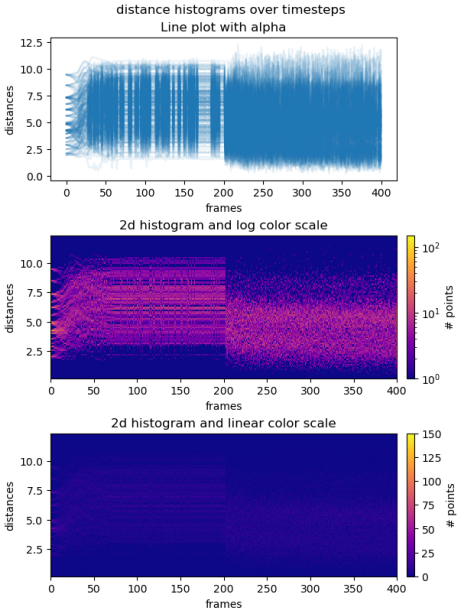
(b) $t = 1000$



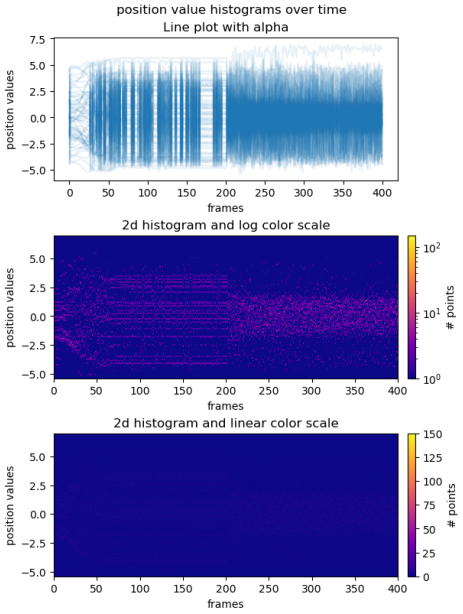
(c) $t = 500$



(d) $t = 0$



(e) Histograms of distance values over forward and backwards diffusion process



(f) Histograms of positional values over forward and backwards diffusion process

Figure 3: Evaluations of a sample from a model trained on the distance matrix of a single sample. 3a shows the original sample. 3b, 3c, and 3d shows the sample at different times in the reverse diffusion process. 3e and 3f shows the histograms of the relative distances and the positional values respectively throughout both the diffusion process and reverse diffusion process. Note here that the term frame represents 5 time steps and the two processes are considered as a single process for visualization purposes.

There are however some notable differences between our implementation of their method and the method presented in [12]. These differences are in the implementation of the equivariant graph convolutional layer described in section 2.2. First, in equation 3 and 4 the sums are normalized by constants $C = 1/N$ and 1 respectively (i.e. no normalization), while we opt to use the normalization constant of $C = 1/100$. Second, the value of the edge inference network ϕ_x in equation 3 is done by re-scaling the coordinate updates for each sample using the hyperbolic tangent and a pre-defined range. Both changes are taken from the implementation of [12], even though these additions are not mentioned in the paper itself. Note that our experiment using this method on a larger dataset does not have this final change implemented as we could not run the experiment in time.

Formally, the whole equivariant graph convolution layer is now defined as

$$\mathbf{m}_{ij} = \phi_e\left(\mathbf{h}_i^l, \mathbf{h}_j^l, ||\mathbf{x}_i^l - \mathbf{x}_j^l||^2, a_{ij}\right) \tag{48}$$

$$\mathbf{x}_i^{l+1} = \mathbf{x}_i^l + \frac{1}{100}\sum_{j\neq i}\left[(\mathbf{x}_i^l - \mathbf{x}_j^l)\tanh\left(\phi_x(\mathbf{m}_{ij})\right)r\right] \tag{49}$$

$$\mathbf{m}_i = \frac{1}{100}\sum_{j\neq i}\mathbf{m}_{ij} \tag{50}$$

$$\mathbf{h}_i^{l+1} = \phi_h\left(\mathbf{h}_i, \mathbf{m}_i\right) \tag{51}$$

Where r is a pre-defined range of which one layer can maximally reposition each node with respect to each neighbor. We set $r = 30$.

Finally, the original implementation requires node features as part of the networks input, to circumvent this we initialize a dummy value for each $\mathbf{h}_i$ such that $\mathbf{h}_i = [1], i \in 1...N$

As also described in [12] the network we use, which consists of a series of equivariant graph convolutional layers, is used to predict the noise at time step t by

$$\left[\hat{\epsilon}_t^{(x)}, \hat{\epsilon}_t^{(h)}\right] = \phi\left(z_t^{(x)}, \left[z_t^{(h)}, t/T, c\right]\right) - \left[z_t^{(x)}, \mathbf{0}\right] \tag{52}$$

where $\left[\hat{\epsilon}_t^{(x)}, \hat{\epsilon}_t^{(h)}\right]$ is the concatenation of the predicted noise on positional coordinates and node features respectively, although the noise on node features is never used. ϕ is a neural network consisting of a series of equivariant graph convolutional layers, and $\left[z_t^{(h)}, t/T, c\right]$ is the concatenation of the node feature embeddings, the time normalized into the range $[0, 1]$, and a conditioning vector c, which is the PDF values. The way we condition our network

is thus through concatenating the conditioning information to the node features. Also notice that the original coordinates are removed from the output of the network only updates the node positions w.r.t. the noise.

5.2 Experiments

5.2.1 Overfitting on single sample

As in section 4.2 we train an unconditional model to overfit on a single sample, to see if we can train a model on a simplified version of the problem. The same sample from section 4.2 was chosen here as well.

To make learning the reverse diffusion easier we choose $T = 10000$, we keep $\gamma = 0.9$ and use the same libraries as in section 4.2. The full network and training hyperparameters can be found in appendix A.2. Again, we evaluate it by examining a single sample in the same manner as in section 4.2, at time steps $t = 10000, 5000, 0$. The results can be seen in figure 4.

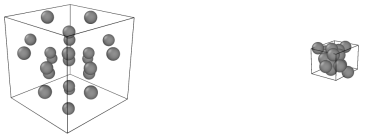
From figure 4 we see that the model is not able to reconstruct sufficient structure in the structure. Since the size in term of position norms in the molecule itself is unbounded we see that it changes over the diffusion process. This phenomenon is explored further in appendix B. This example was handpicked, but all other examined samples show the same trends.

5.2.2 Overfitting on cubic structure

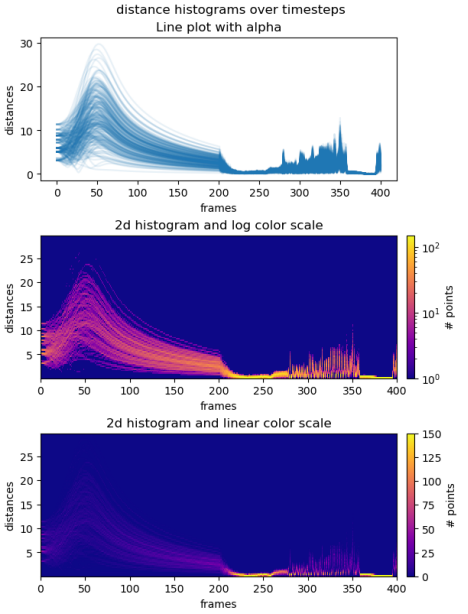
Having found in 5.2.1 that overfitting on a single sample proved difficult, to simplify the problem even further we experiment with overfiting a model on an artificially constructed cubic structure that is not physically feasible in the real world but is significantly simpler. The cube is constructed to have 8 corners, zero-centered, and each side has a length of 1. We note that such a cube is perfectly regular and so many distances are the same, making it hard for the equivariant graph convolutional layers to differentiate between them. To remedy this, we introduce a small amount of noise sampled from $\mathcal{N}(0, 0.001)$ on all the nodes of the cube each time it is sampled.

All network and training hyperparameters are the same as in section 5.2.1 and can also be found in appendix A.3. A curated generated sample from the trained model can be seen in figure 5

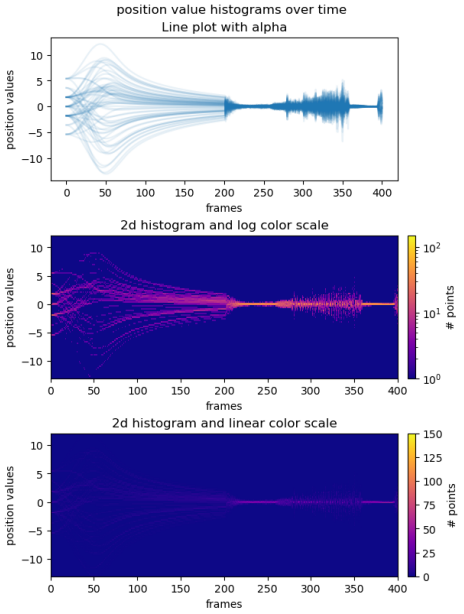
From figure 5 we see that some structure is learned, although it is not perfect. We see, in a sense, that it learns to reconstruct the cube, although it does not get all the distances right, it does reconstruct some of the regularities.



(a) Original (b) $t = 10000$ (c) $t = 5000$ (d) $t = 0$



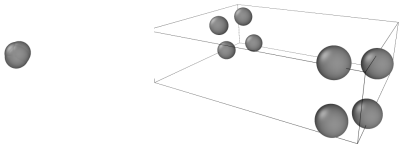
(e) Histograms of distance values over forward and backwards diffusion process



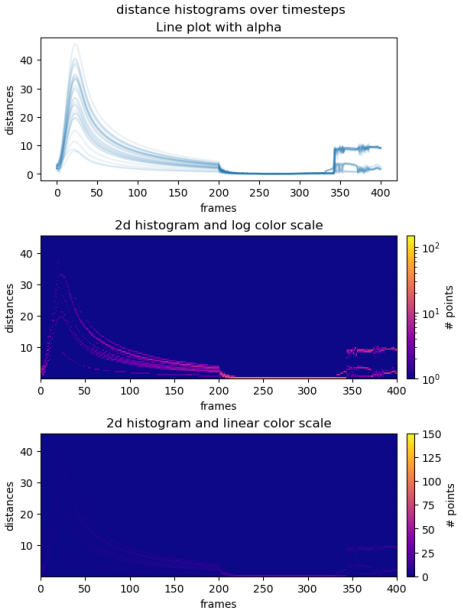
(f) Histograms of positional values over forward and backwards diffusion process

Figure 4: Evaluations of a sample from a model trained on the point positions of atoms. 4a shows the original sample. 4b, 4c, and 4d shows the sample at different times in the reverse diffusion process. 4e and 4f shows the histograms of the relative distances and the positional values respectively throughout both the diffusion process and reverse diffusion process. Note here that the term frame represents 50 time steps and the two processes are considered as a single process for visualization purposes.

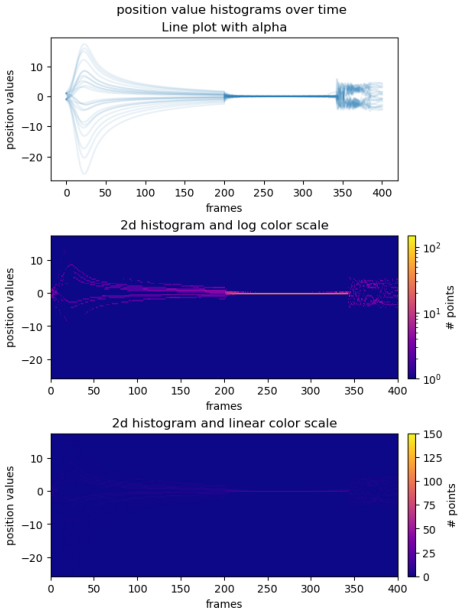
(a) Original (b) $t = 10000$



(c) $t = 5000$ (d) $t = 0$



(e) Histograms of distance values over forward and backwards diffusion process



(f) Histograms of positional values over forward and backwards diffusion process

Figure 5: Evaluations of a sample from a model trained on the point positions of atoms. 5a shows the original sample. 5b, 5c, and 5d shows the sample at different times in the reverse diffusion process. 5e and 5f shows the histograms of the relative distances and the positional values respectively throughout both the diffusion process and reverse diffusion process. Note here that the term frame represents 50 time steps and the two processes are considered as a single process for visualization purposes.

We also see that the model does not reconstruct the sample until around $t = 7500$ from which it creates the structure instantly.

5.2.3 Training on larger dataset

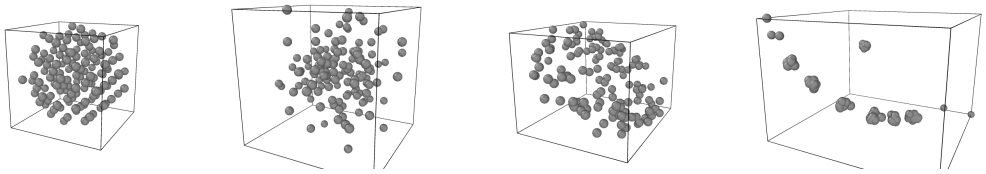
Along with experimenting with unconditioned generation using small samples, we experiment with a larger subset of the dataset. This subset consists of all 432 structures with 135 atoms, which lets us have the largest possible dataset while only having structures with the same number of atoms, allowing us to avoid padding tokens as part of our input. Each structure is also accompanied by a PDF. This dataset was split into an 80% training dataset and a 20% validation dataset. The training and sampling is done with $T = 1000$, and $\gamma = 0.9$, and the model is conditioned on the PDFs. The full network and training hyperparameters can be found in appendix A.4. An example of the results of this experiment can be seen in figure 6. Although this example was selected manually out of a batch of 64 for illustrative purposes, the rest of the batch exhibit the same behavior.

The selected sample shows the trends observed in all the generated samples we examined. Herein we see that the sampling process converges to NaN values, but not before showing some structure, although it is not the one, we would expect. For each sample in the validation set we also sampled 16 different structures and compared their computed PDFs to the true PDF using R_{wp}. Across different samples we aggregated means and standard deviations, which can be seen in figure 7. For reference, according to [7] a solid R_{wp} is between 0.01 and 0.05.

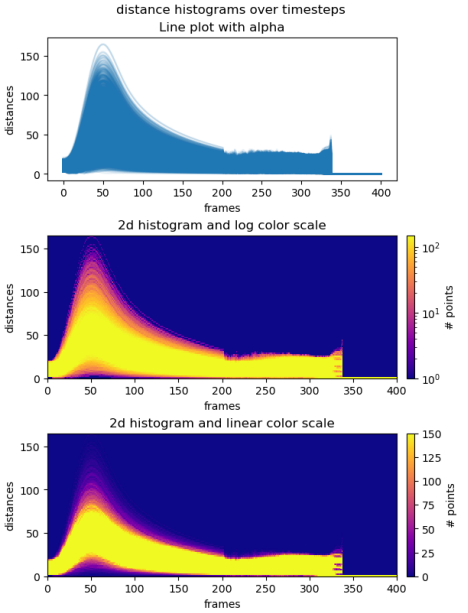
6 Discussion

6.1 Reflection of methods

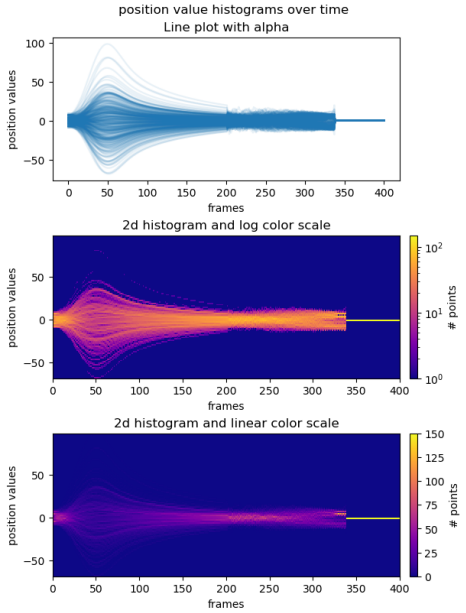
Through the different approaches proposals on how to model the problem of generating mono-metallic structures with diffusion models we see that the problem allows for multiple interpretations each with their own advantages. The distance matrix modelling approach offered a representation close to the properties we are interested in modelling, that being interatomic distances, but utilizes methods not fit independent Gaussians as noise functions. This method relies on reconstructing a set of atom positions from a predicted distance matrix, which itself is not guaranteed to adhere to the metric properties. This makes the predicted atom positions the minimum of an optimization problem to find the best fit, which can shift and vary the results in



(a) Original (b) $t = 1000$ (c) $t = 500$ (d) $t = 315$

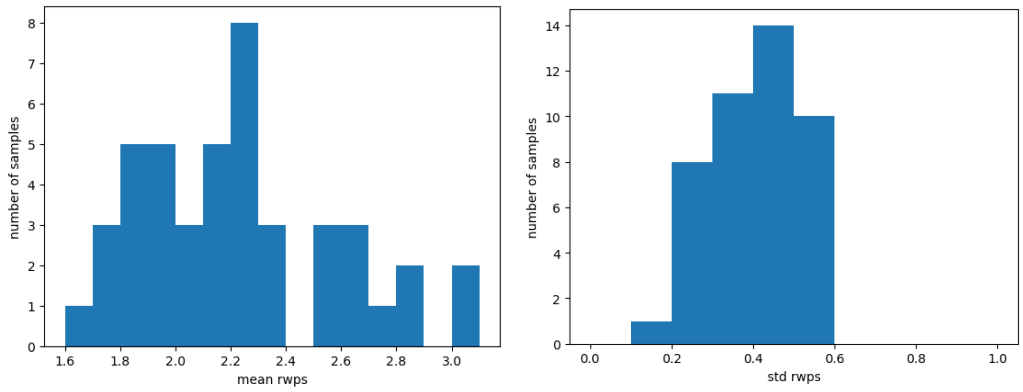


(e) Histograms of distance values over forward and backwards diffusion process



(f) Histograms of positional values over forward and backwards diffusion process

Figure 6: Evaluations of a sample from a model trained on the point positions of atoms. 6a shows the original sample. 6b, 6c, and 6d shows the sample at different times in the reverse diffusion process. 6e and 6f shows the histograms of the relative distances and the positional values respectively throughout both the diffusion process and reverse diffusion process. Note here that the term frame represents 5 time steps and the two processes are considered as a single process for visualization purposes. Note also that the 0-values we observe after around $t = 315$ are NaN values replaced by 0s.



(a) Histogram of the number of atoms in each structure.

(b) Histogram of the largest distance between two atoms in each structure.

Figure 7: For each sample in the 135-atom structure validation set, we sampled 10 different structures and estimated their PDFs. Then we computed the R_{wp} values. This figure shows histograms of the mean and standard deviation of these R_{wp} measures computed over the samples from each validation example. Note that an R_{wp} between 0.01 and 0.05 very good fit.

an unwanted manner.

This restriction is inherent to the choice of representing the positions via their relative distances, something we do not address directly but instead we rely on the network to learn the denoising process well enough. A direction to develop this method is to construct a noise function that operates on the distance matrix, but still upholds the metric properties, such that the noise itself is applied to the positions.

Another approach to modelling the molecule through its distance matrix is to assume the molecule a discrete graph with the distances being discretized. This method is explored deeper in appendix C, and while it allows for a natural formulation of the node features in the system, it still falls prone to the restrictions mentioned above. Despite these limitations, we chose to initially complete this experiment as we wanted to gain experience with modelling the problem and be sure if this was an issue.

The coordinate modelling does not need to adhere to these restrictions, as it provides a more natural modelling approach to the data's structure. We explored this method in greater depth with more experiments than with the distance model, and this is owed to the results of the distance model. We will discuss this further in section 6.2.

Our methods were only explored preliminary in our experiments, as such we only present preliminary and very qualitative results. Had our preliminary results been more promising, we would have evaluated on a larger scale, calculating the Fréchet inception distance (FID) for samples generated by our models and the feasibility of the molecular structures.

6.2 Reflection of results

The method viewing the atomic structures as a matrix of interatomic distances was thought to be an initial modelling method to gain knowledge of the task at hand and explore the data and the difficulties of modelling it going ahead. In figure 3 we saw that our distance matrix model failed to learn to reconstruct our single example. We believe the poor results are in large part to two reasons: This method was too simple to model such a complex data type, and our model inherently broke with the assumptions of Euclidean space. We expected the figure to show similar patterns on the denoising side as on the diffusion side. We see structured patterns of distances as noise is added in the first part of the plot, but during denoising we see that the distances and positions become very noisy. This noise was very apparent and signaled a complete failure to learn, along with showing that adding and subtracting noise in such a way shuffled positions and distances. This experiment showed clear signs that this method was not the right one, and as such we decided to pursue it directly with the coordinate model to see what results we could yield.

In figure 4 we covered the results of our experiments on overfitting on a single sample using the equivariant coordinate method. Compared to the results from our distance matrix experiments, we see the difference between adding noise to the positions compared to the distances. As we added noise to an example, we saw the distances increase in size before converging towards noise. This effect is discussed in appendix B. We do, however, see that the model failed to converge towards any structure and instead collapsed into a point. After some discussions we wanted to experiment with even smaller structures to understand if it was an inherent problem of our model, leading to the experiment on a cubic structure. In section 6.4 we further discuss reasons for this vertex collapse and lack of convergence on a single sample.

Figure 5 show the results of our experiments on trying to fit this to a cube with some noise added to each edge. We see that the sampling does converge to some structure in an interesting way. The atoms lie in a singularity up until around $t = 7500$ diffusion steps, at which it begins to form a stable structure.

This was an interesting structure, as 4 atoms each formed a structured face of a cube, even though they did not connect. This shows that the equivariant diffusion model can learn some of these 3-dimensional features.

In 6 we conducted a larger experiment on all structures with 135 atoms. The figure shows how training on a larger set of structures increased sampling ability and consistency. Just before the NaN value explosion, the denoising process did produce atomic structures that had some structure to them, as seen in figure 6d. Note also that this model was trained on conditional data, meaning that this sampling worked while we conditioned the structures. In section 6.4 we discuss how the model's ability to sample structured monometals may be a result of the lack of simplifications we have made.

Figure 7 gives us insight into how well this model performs at sampling structures with appropriate PDFs. The main take away here is that our model could not sample structures from our distribution of atomic structures in any way. This result should come as no surprise given how we in figure 6 saw how sampling produced structures with 0's for atom coordinates. We note that the goal of our experiments was to understand if we could make sense of some sampling in this more complicated data space and that constructing a model that could reliably sample atomic structures was therefore not the end goal. Regardless, this figure shows that there still is a long way to go in working on this problem. Thus, a main takeaway from our results is we failed to model the reverse diffusion process of the molecular structures fully using this coordinate modelling approach, suggesting that it needs more work to function properly.

One point of discussion related to the results of all our experiments is how denoising steps change behavior after around 140 to 160 frames. We currently do not have a solid explanation for this behavior and believe this should be researched further.

As mentioned in section 5.1 our coordinate experiment on a larger dataset was not performed using a tanh bounding for the update of atoms. Given how the latter stages of the denoising process produced NaN values for which coordinates were changed to 0's, we think there is potential for the tanh change to remedy this in case that these NaN values were produces by exceptionally large movements in atoms. Another reason we theorized but could not confirm was a potential divide by zero error somewhere in the code.

6.3 Complexity of the problem

The problem of generating atomic structures is not novel and is also explored in the methods we took inspiration from [12,26]. These methods are often operated on datasets like [2,4,21,22] that consist of molecules that are relatively small with some datasets only consisting of molecules with up to 9 atoms, while our problem requires us to model structures of larger sizes as seen in figure 2. The molecules from the referenced datasets also have more inherent structures by virtue of modelling bio-chemic structures. As part of the higher degree of structure, information about individual bonds between atoms are also more nuanced and cross correlated with the rest of the graph, raising the level of complexity of the molecule but also introducing more structure for a denoising model to exploit. The intuition being that the more ways a node or edge of a molecule can be structurally unsound because of noise, the easier it would be for a denoising network to predict said noise.

The larger structure size and lower degree of structure makes the reverse diffusion process a lot harder to model for our problem, since our possible solution space is much larger and the correct solution space is the same size, partly explaining the difficulty of successfully modeling the reverse diffusion process. Other methods like [15] also model structures like ours using variational models, they however omit the oxygen atom and only the monometallic structures.

6.4 Issues with simplifying the problem

To handle the complexity of the problem, we chose to simplify the problem and work up the complexity slowly. The simplified experiments did, however, not show any promise and so we never advanced from that stage. Overall, we made 2 simplifying assumptions, we chose to not model the node features and only their positions, and we tried training a model on a reduced dataset, just 1 sample, to make it overfit.

The first simplifying assumptions greatly reduce the possible solution space and makes the problem purely about generating feasible 3D positions of atoms. If this assumption proved necessary to solve the problem, then the problem of then labelling the nodes by their atom type could be solved by another network, as predicting the atom type is easy given, we got the positions correct.

The second simplifying assumption was intended to see if our network could be overfit to a single sample, with the notion being that it could not fit an entire distribution if it could not fit a single sample. We have reconsidered

if this simplifies the problem as the network does not predict the denoised sample directly, but rather the noise used to diffuse it. If we see the problem of learning to generate a single sample from the angle of stochastic differential equations (SDEs), we can see that the learned reverse diffusion process is a mapping from the prior to a single point in a high dimensional space. Intuitively, we can see that it should be easier to learn a single valid trajectory endpoint. But we do also see that since the reverse diffusion process is stochastic, adding in more valid endpoints to the trajectory by adding more samples to the training set would make learning the trajectories more robust. From our results this can be empirically hinted at by examining figures 5 and 6, where the model is trained on a dataset with more than one sample.[1]. It is only in these examples we see some learned structure in contrast to figure 4 where no such structure is learned.

6.5 Modelling atom types

The scope of this project does not include modelling the node features, which is an integral part of solving the overarching problem of generating mono-metal oxide structures. The node features are inherently different from the features we model, namely the positions, in that the node features discrete values, i.e., atom types. Approaches to circumvent this problem would be to model both the atom types and positions in the same space, e.g., both discrete as discussed further in appendix C, or both real valued. A simplistic way of introducing atom types to the model would be a binary value indicating whether it is oxygen or a metal, since all molecules consist of only oxygen atoms and a single type of metal. This is still a simplification as the reverse diffusion process could not model which type of metal the molecule represents, but it would make the input space a lot simpler than introducing a one-hot encoding of all metal types. Modelling the atom types as an integer value representing their atomic number is not theoretically sound since we would define a semantic space where arithmetic operations on atom types is defined, which does not make sense in this domain. The one-hot encoding has been used in [12] where they assume discrete values to be 0 or 1 and assume that the final sample x_0 will have values close to 0 or 1. This method is not well supported theoretically but has been shown to work. Another promising way of modelling categorical values in a Gaussian diffusion setting is explored in [6], which provides a more natural representation of a categorical value with a domain size larger than 2. We however believe that the most

[1]The cube structure is one sample but with added noise at each sampling time, effectively making it a dataset with a highly peaked distribution.

theoretically sound formulation of discrete and continuous values requires a noise model specific to the problem, which we will discuss in detail in section 6.7.

6.6 Alternatives to fixing the dimensionality

One property of the data modality used in this project and similar ones is that molecular structures come in varied sizes. To remedy this, the molecular structure is usually encoded as a sequence of features of nodes, edges, or both. These sequences have varying lengths which we do not handle in this project directly, but instead train on selected samples such that this is not an issue. For further experimentation we would have padded sequences with dummy vectors to allow for efficient computation of a batch with input of differing sizes. This, however, introduces some bias as we *require* our model to learn to ignore padding tokens in the case of the coordinate modeling. For the distance matrix modelling the self-attention mechanism has a natural way of ignoring padding tokens.

For sampling we need a pre-defined sequence size to generate the sample. For our experiments this was not a problem, as we would set the size to fit the sizes in the training set. If we were to model these explicitly, we only consider setting the size manually and thus letting it be part of the conditioning, making our model require molecule sizes for generation. Other approaches explored in [12, 13] first sample the molecule sizes before sampling them from the diffusion model, which would be a natural formulation for a distribution unconditioned on the molecule sizes. Another way would be to incorporate a change in dimensionality during the diffusion process by introducing jump diffusions as done in [5], which would incorporate the sequence size as a part of the generative process in a more natural way.

6.7 Custom noise models

Throughout this project we have utilized a Gaussian noise model because our modelling was done on positions that exist in a continuous domain. Other noise models have been discussed in section 2.5 and appendix C. For the coordinate modelling the Gaussian noise fits the problem well, but for the distance matrix modelling we ran into some incompatibilities. We believe it worth exploring whether a noise model that works on a distance matrix while preserving the metric properties is possible, while not explicitly calculating the Cartesian coordinates at each step. This custom noise would also require a tractable prior to sample from, and a closed form expression for

easy training.

Without modelling the node features the data only lives in the continuous domain, but in general the problem exists in a hybrid of continuous and discrete space. Atom types will always be inherently discrete while the positions or distances will always be real numbers. We see that the end goal of working with this problem would be developing a hybrid noise model that can work on a hybrid data representation of both continuous and discrete values.

7 Conclusion

In this project we approached the problem of generating candidate monometal oxide structures that could be mapped to a given PDF by modelling the conditional distribution using diffusion models. This project focused on learning the distribution of simulated atomic structures through probabilistic denoising diffusion models. To simplify the learning problem, we omitted atom types and focused on modelling only the relative positions of atom in the molecules.

We explored different methods of modelling the problem, treating structures as distance matrix and collections of Cartesian coordinates. The distance matrix modelling more correctly reflected the proper structure of the structures in terms of PDF simulation but proved too ill-defined for diffusion modelling to work without a lot more careful thought. Our initial experiments into this showed no hope of effectively sampling any structure.

The Cartesian coordinate modelling approach was well suited to diffusion modelling, but it proved too hard to learn the reverse diffusion process successfully with our approach. We experimented with overfitting on a single sample, which did not produce solid results. We also experimented on a small noisy cubic structure, on which our model did learn certain structures and was able to somewhat produce them. Finally, we experimented on a larger set of structures of similar number of atoms and with conditioning on PDFs and saw structural patterns in the generation made by the models, although this did not produce any meaningful atomic structure at sample end.

We reflected on the complexity of the problem posed, comparing it to other methods which have been shown to successfully model molecular structures, but on a much smaller scale. Our idea of overfitting on a single sample was also discussed, as only allowing all reverse diffusion trajectories to have the same endpoints, i.e., modelling a *very* highly peaked distribution, could make learning it harder. Furthermore, we discussed that further experimentation

should be done into this Cartesian modelling structure due to the promise it has shown, and that for the problem to be fully solved there also need to be made research into noise functions that can handle both continuous and discrete noise.

8 Future work

Our ideas for future work come in two overall categories, experimenting with other ways of modelling the molecular structures, and finding a better way to simplify the problem.

Discrete modelling would be a natural way to continue exploring the problem, hopefully leading to a theoretically sound formulation of a hybrid approach that models atom types and positions in their natural domains.

Further exploring how to make learning the reverse diffusion process easier, would allow for better prototyping and exploration of new methods. We would pose the question of how to make the reverse diffusion trajectories as simple to learn as possible, even at the cost of over-simplifying the original problem.

References

[1] Jacob Austin, Daniel D. Johnson, Jonathan Ho, Daniel Tarlow, and Rianne van den Berg. Structured denoising diffusion models in discrete state-spaces. *CoRR*, abs/2107.03006, 2021.

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

[3] Ingwer Borg and Patrick JF Groenen. *Modern multidimensional scaling: Theory and applications*. Springer Science & Business Media, 2005.

[4] Nathan Brown, Marco Fiscato, Marwin H.S. Segler, and Alain C. Vaucher. Guacamol: Benchmarking models for de novo molecular design. *Journal of Chemical Information and Modeling*, 59(3):1096–1108, 2019. PMID: 30887799.

[5] Andrew Campbell, William Harvey, Christian Weilbach, Valentin De Bortoli, Tom Rainforth, and Arnaud Doucet. Trans-dimensional generative modeling via jump diffusion models. *arXiv preprint arXiv:2305.16261*, 2023.

[6] Ting Chen, Ruixiang Zhang, and Geoffrey Hinton. Analog bits: Generating discrete data using diffusion models with self-conditioning. *arXiv preprint arXiv:2208.04202*, 2022.

[7] Troels Lindahl Christiansen, Susan R. Cooper, and Kirsten M. Ø. Jensen. There's no place like real-space: elucidating size-dependent atomic structure of nanomaterials using pair distribution function analysis. *Nanoscale Advances*, 2(6):2234–2254, 2020.

[8] William Falcon and The PyTorch Lightning team. PyTorch Lightning. *GitHub*, March 2019.

[9] Justin Gilmer, Samuel S Schoenholz, Patrick F Riley, Oriol Vinyals, and George E Dahl. Neural message passing for quantum chemistry. In *International conference on machine learning*, pages 1263–1272. PMLR, 2017.

[10] Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. *Advances in Neural Information Processing Systems*, 33:6840–6851, 2020.

[11] Jonathan Ho and Tim Salimans. Classifier-free diffusion guidance. *arXiv preprint arXiv:2207.12598*, 2022.

[12] Emiel Hoogeboom, Vıctor Garcia Satorras, Clément Vignac, and Max Welling. Equivariant diffusion for molecule generation in 3d. In *International Conference on Machine Learning*, pages 8867–8887. PMLR, 2022.

[13] Ilia Igashov, Hannes Stärk, Clément Vignac, Victor Garcia Satorras, Pascal Frossard, Max Welling, Michael Bronstein, and Bruno Correia. Equivariant 3d-conditional diffusion models for molecular linker design. *arXiv preprint arXiv:2210.05274*, 2022.

[14] Pavol Juhás, Christopher L. Farrow, Xiaohao Yang, Kevin R. Knox, and Simon J. L. Billinge. Complex modeling: a strategy and software program for combining multiple information sources to solve ill posed structure and nanostructure inverse problems. *Acta Crystallographica Section A*, 71(6):562–568, Nov 2015.

[15] Emil TS Kjær, Andy S Anker, Marcus N Weng, Simon JL Billinge, Raghavendra Selvan, and Kirsten MØ Jensen. Deepstruc: Towards structure solution from pair distribution function data using deep generative models. *Digital Discovery*, 2023.

[16] Joseph B Kruskal. Multidimensional scaling by optimizing goodness of fit to a nonmetric hypothesis. *Psychometrika*, 29(1):1–27, 1964.

[17] Calvin Luo. Understanding diffusion models: A unified perspective, 2022.

[18] Alex Nichol and Prafulla Dhariwal. Improved denoising diffusion probabilistic models, 2021.

[19] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. Pytorch: An imperative style, high-performance deep learning library. In *Advances in Neural Information Processing Systems 32*, pages 8024–8035. Curran Associates, Inc., 2019.

[20] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

[21] Daniil Polykovskiy, Alexander Zhebrak, Benjamin Sanchez-Lengeling, Sergey Golovanov, Oktai Tatanov, Stanislav Belyaev, Rauf Kurbanov, Aleksey Artamonov, Vladimir Aladinskiy, Mark Veselov, Artur Kadurin, Simon Johansson, Hongming Chen, Sergey Nikolenko, Alán Aspuru-Guzik, and Alex Zhavoronkov. Molecular sets (moses): A benchmarking platform for molecular generation models. *Frontiers in Pharmacology*, 11, 2020.

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

[23] Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. High-resolution image synthesis with latent diffusion models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 10684–10695, 2022.

[24] Vıctor Garcia Satorras, Emiel Hoogeboom, and Max Welling. E (n) equivariant graph neural networks. In *International conference on machine learning*, pages 9323–9332. PMLR, 2021.

[25] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

[26] Clement Vignac, Igor Krawczuk, Antoine Siraudin, Bohan Wang, Volkan Cevher, and Pascal Frossard. Digress: Discrete denoising diffusion for graph generation, 2023.

[27] X. Wang. Pair distribution function analysis: Fundamentals and application to battery materials. *Chinese Physics B*, 2020.

[28] Xuelong Wang, Sha Tan, Xiao-Qing Yang, and Enyuan Hu. Pair distribution function analysis: Fundamentals and application to battery materials. *Chinese Physics. B*, 29(2), 12 2019.

[29] Zhisheng Xiao, Karsten Kreis, and Arash Vahdat. Tackling the generative learning trilemma with denoising diffusion gans. *arXiv preprint arXiv:2112.07804*, 2021.

Parameter	Value
Optimizer	Adam
β_1	0.9
β_2	0.999
Learning rate	0.0001
Epochs	1000
Batch size	64

Figure 8: Training hyperparameters for self attention experiment

9 Appendix

A Model architectures and training hyperparameters

A.1 Distance matrix modelling overfitting on single sample

The model takes in tensors of shape *batch, n_nodes, n_nodes, edge_features* and has structure

$$\begin{aligned} &\textit{Linear(edge\_features, hidden\_dim)} \\ &\rightarrow \textit{SelfAttention(n\_heads, hidden\_dim)} \times 10 \\ &\rightarrow \textit{Linear(hidden\_dim, edge\_features)} \end{aligned}$$

Where *SelfAttention* takes edge embeddings and a time embedding which is encoded by

$$\begin{aligned} &\textit{PositionalEmbedding(t)} \\ &\rightarrow \textit{SiLU}() \\ &\rightarrow \textit{Linear}(t, \textit{hidden}_d\textit{im}) \end{aligned}$$

Where *edge_features* is one, corresponding to the single distance of the edge, *n_heads* is 16, and *hidden_dim* is 64. The training hyperparameters can be seen in figure 8 where the dataset consisted of the same sample cloned 128 times.

A.2 Coordinate modelling overfitting on single sample

The model structure consists of EQLayers which are described in [12], our model consists of a learned projection of the conditioning vectors. The pdf

Parameter	Value
Optimizer	Adam
β_1	0.9
β_2	0.999
Learning rate	0.0001
Epochs	10000
Batch size	64

Figure 9: Training hyperparameters for coordinate modelling on single sample

embedding is

$$\begin{aligned} &\textit{Linear(3000, 512)} \\ &\rightarrow GELU() \\ &\rightarrow \textit{Dropout}(0.1) \\ &\rightarrow \textit{Linear(512, 256)} \\ &\rightarrow GELU() \\ &\rightarrow \textit{Dropout}(0.1) \end{aligned}$$

The time embedding is done as in [12] with a gamma network, and the dummy nodes are encoded via

$$\begin{aligned} &\textit{Linear(1, 10)} \\ &\rightarrow GELU() \\ &\rightarrow \textit{Linear(10, 256)} \\ &\rightarrow GELU() \\ &\rightarrow \textit{Dropout}(0.1) \end{aligned}$$

The PDF, time, and node embedding are concatenated and passed through 9 EQLayers. The training hyperparameters can be found in figure 9.

A.3 Coordinate modelling on cubic structure

This experiment uses the same network as described in A.2, but the the dataset consists of a cube noise sampled from $\mathcal{N}(0, 0.001)$ at each sampling time. The dataset consists of 128 samples of this cube and the training hyperparameters can be found in figure 10.

Parameter	Value
Optimizer	Adam
β_1	0.9
β_2	0.999
Learning rate	0.0001
Epochs	10000
Batch size	64

Figure 10: Training hyperparameters for coordinate modelling on cube

Parameter	Value
Optimizer	Adam
β_1	0.9
β_2	0.999
Learning rate	0.0001
Epochs	1000
Batch size	64

Figure 11: Training hyperparameters for coordinate modelling on dataset

A.4 Coordinate modelling on multiple samples

This experiment uses the same network as described in A.2, but without the hyperbolic tangent trick described in section 5.1. The training hyperparameters can be found in figure 11

B Norm of latent variables

We will briefly describe why we see the relative distances and absolute positional values increase and decrease in the forward diffusion process as seen in figures 4, 5, and 6. We will do this through examining the norm of the molecule w.r.t. the different time steps t. Recall that $x_t = \sqrt{\bar{\alpha}_t}x_0 + \sqrt{1-\bar{\alpha}_T}\epsilon, \bar{\alpha} \in [0,1], \epsilon \sim \mathcal{N}(0,1)$, which is a non-linear interpolation and as such we will contrast it against a similar linear interpolation $x_t = \bar{\alpha}_t x_0 + (1-\bar{\alpha}_t)\epsilon, \bar{\alpha}_t \in [0,1]$. We immediately see that since $\bar{\alpha}_t \in [0,1]$ we have that $\sqrt{\bar{\alpha}_t} \geq \bar{\alpha}_t$ and $\sqrt{1-\bar{\alpha}_t} \geq 1-\bar{\alpha}_t$ which implies $\sqrt{\bar{\alpha}_t}+\sqrt{1-\bar{\alpha}_t} \geq \bar{\alpha}_t+(1-\bar{\alpha}_t) = 1$. We can conclude that for x_0 and ϵ with the same norm, the norm of x_0 will always be greater than or equal. A visualization of this can be seen in figure 12.

We briefly also explore the effect of relative different norm between x_0 and ϵ

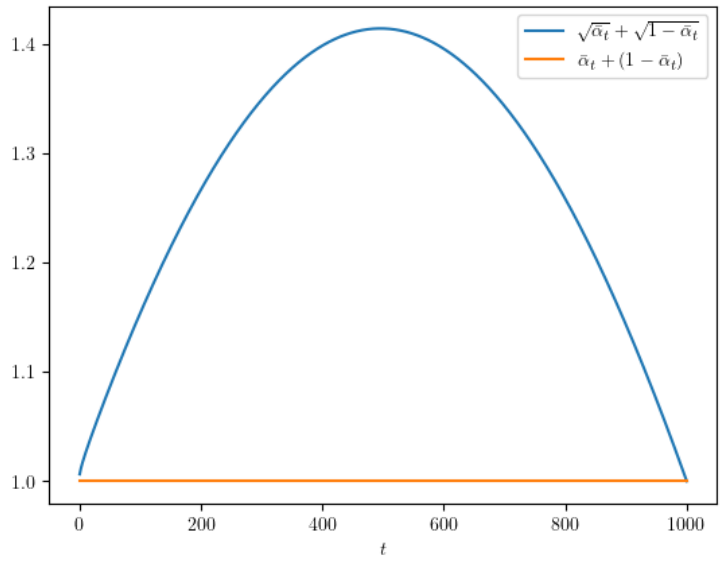


Figure 12: Visualization of linear-interpolation and the non-linear interpolation, $\bar{\alpha}_t$ is calculated from the cosine schedule described in section 2.6.

in figure 13. Here we see that the norm will peak at the same time we see the peaks in figures 4, 5, and 6.

C Discrete modelling

Early on we discussed using discrete diffusion modelling as an alternative to the continuous modelling covered previously. These methods were decided in tandem with our continuous methods. However, as we experimented with our continuous modelling methods and discovered inherent issues, we decided to focus on those before continuing with our discrete methods. As a result we did not end up performing experiments using these methods; however we still mention these as we believe they can be useful for a reader looking for inspiration.

C.1 Methods

All of the modeling performed has been done using gaussian noise on continuous noise features. In this section we cover the methods we considered and decided upon when experimenting with discrete diffusion modelling. Since the data we work with is a mixture of continuous and discrete data, covering methods with discrete nature is very relevant to this problem.

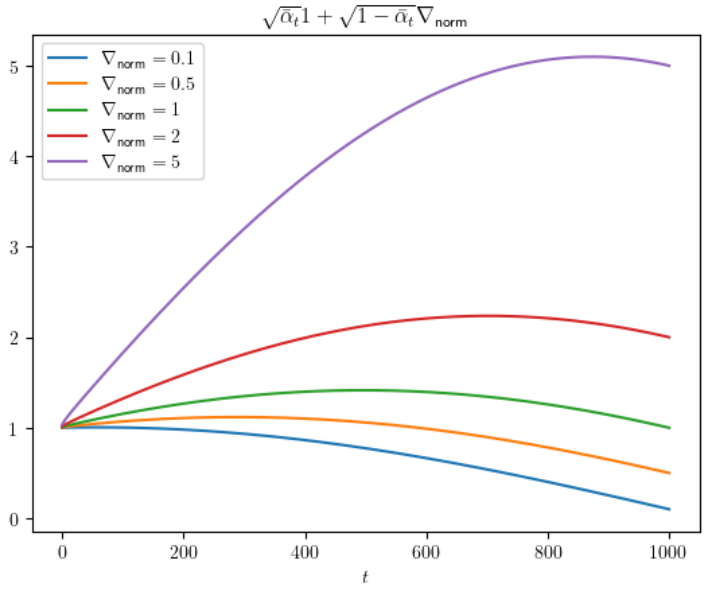


Figure 13: norm of x_t with different relative norms between x_0 and ϵ

In previous sections we covered ways of making this problem purely continuous by simplifying the problem. When modelling this problem discretely such simplifications become more challenging and explicit choices need to be made. In section 2.5 we covered theory behind discrete diffusion established by [26]. When performing experiments with their model they used data that was completely discrete. It consisted of discrete graph data where bonds between atoms were encoded as edge features, and each atom had no coordinate. In our case we are concerned with the distances between atoms, among other structural properties.

For each vertex of the graph we store the atomic number as a discrete state. Furthermore, for each edge we store the distance betweeen the two atoms connected by such distance. We here construct the graph to be fully connected. Knowing that the overall size of a structure never exceeded 30 Ångstrøm, each edge distance between 0 and 30 Ångstrøm. This is encoded as discretised bands between 0 and 30. When constructing this discretised band we make a tradeoff between how easily we can model it and how fine we can make it. Making the band have 300 states each indecates an increment of 0.1 Ångstrøm would allow us to model a fine granularity but would make modelling much more difficult than having a band of 30 states each indicating an integer between 1 and 30.

We use the model described in [26]. The inputs of the models are the fea-

ture matrices $\mathbf{X}$ and $\mathbf{E}$. The first step is to compute what the authors call structural and spectral features. They construct a feature $\mathbf{y}$ that is a concatenation of vectors with different types of information: How many k-cycles does the graph have from $k = 3$ to 6, what nodes belong to these cycles from $k = 3$ to 5. [26] included more features, the number of connected components and the 5 first non-zero eigenvalues, an estimation of the biggest connected component, and the first two eigenvectors associated with non-zero eigenvalues. They also use molecular features, this being the valency of each atom and the current molecular weight of the full molecules. However, to simplify modelling and because we did not have access to some of these features we decided not to include these.