Masked Graph Modeling for Molecule Generation

Omar Mahmood[1], Elman Mansimov[3], Richard Bonneau[2], and Kyunghyun Cho[3,*]

[1]Center for Data Science, New York University, New York, NY, USA
[2]Center for Genomics and Systems Biology, New York University, New York, NY, USA
[3]Department of Computer Science, Courant Institute of Mathematical Sciences, New York, NY, USA
[*]Corresponding Author

Abstract

De novo, in-silico design of molecules is a challenging problem with applications in drug discovery and material design. We introduce a masked graph model, which learns a distribution over graphs by capturing conditional distributions over unobserved nodes (atoms) and edges (bonds) given observed ones. We train and then sample from our model by iteratively masking and replacing different parts of initialized graphs. We evaluate our approach on the QM9 and ChEMBL datasets using the GuacaMol distribution-learning benchmark. We find that validity, KL-divergence and Fréchet ChemNet Distance scores are anti-correlated with novelty, and that we can trade off between these metrics more effectively than existing models. On distributional metrics, our model outperforms previously proposed graph-based approaches and is competitive with SMILES-based approaches. Finally, we show our model generates molecules with desired values of specified properties while maintaining physiochemical similarity to the training distribution.

Introduction

The design of de novo molecules in-silico with desired properties is an essential part of drug discovery and materials design but remains a challenging problem due to the very large combinatorial space of all possible synthesizable molecules [1]. Recently, various deep generative models for the task of molecular graph generation have been proposed, including: neural autoregressive models [2, 3], variational autoencoders [4, 5], adversarial autoencoders [6], and generative adversarial networks [7, 8].

A unifying theme behind these approaches is that they model the underlying distribution $p^\star(G)$ of molecular graphs G. Once the underlying distribution is captured, new molecular graphs are sampled accordingly. As we do not have access to this underlying distribution, it is typical to explicitly model $p^\star(G)$ by a distribution $p_\theta(G)$. This is done using a function f_θ so that $p_\theta(G) = f_\theta(G)$. The parameters θ are then learned by minimizing the KL-divergence $KL(p^\star \| p_\theta)$ between the true distribution and the parameterized distribution. Since we do not have access to $p^\star(G)$, $KL(p^\star \| p_\theta)$ is approximated using a training set $D = (G_1, G_2, ..., G_M)$

which consists of samples from $p^\star$. Once the model has been trained on this distribution, it is used to carry out generation.

Each of these approaches makes unique assumptions about the underlying probabilistic structure of a molecular graph. Autoregressive models [2, 9, 3, 10, 11, 12] specify an ordering of atoms and bonds in advance to model the graph. They decompose the distribution $p(G)$ as a product of temporal conditional distributions $p(g_t|G_{<t})$, where g_t is the vertex or edge to be added to G at time t and $G_{<t}$ are the vertices and edges that have been added in previous steps. Generation from an autoregressive model is often done sequentially by ancestral sampling. Defining such a distribution requires fixing an ordering of the nodes and vertices of a graph in advance. Although directed acyclic graphs have canonical orderings based on breadth-first search (BFS) and depth-first search (DFS), graphs can take a variety of valid orderings. The choice of ordering is largely arbitrary, and it is hard to predict how a particular choice of ordering will impact the learning process [13].

Latent variable models such as variational autoencoders and adversarial autoencoders assume the existence of unobserved (latent) variables $Z = \{z_1, z_2, ..., z_k\}$ that aim to capture dependencies among the vertices V and edges E of a graph G. Unlike an autoregressive model, a latent variable model does not necessarily require a predefined ordering of the graph [14]. The generation process consists of first sampling latent variables according to their prior distributions, followed by sampling vertices and edges conditioned on these latent variable samples. However, learning the parameters θ of a latent variable model is more challenging than learning the parameters of an autoregressive model. It requires marginalizing latent variables to compute the marginal probability of a graph, i.e., $p(G) = \int_Z p(G|Z)p(Z)dZ$, which is often intractable. Recent approaches have focused on deriving a tractable lower-bound to the marginal probability by introducing an approximate posterior distribution $q(Z)$ and maximizing this lowerbound instead [4, 5, 6]. Unlike variational autoencoders, generative adversarial networks (GAN) do not use KL-divergence to measure the discrepancy between the model distribution and data distribution and instead estimate the divergence as a part of learning.

Here, we explore another approach to probabilistic graph generation based on the insight that we do not need to model the joint distribution $p(G)$ directly to be able to sample from it. We propose a *masked graph model*, a generative model of graphs that learns the conditional distribution of masked graph components given the rest of the graph, induced by the underlying joint distribution. This allows us to use a procedure similar to Gibbs sampling to generate new molecular graphs, as Gibbs sampling requires access only to conditional distributions. Concretely, our approach, to which we refer as *masked graph modeling*, parameterizes and learns conditional distributions $p(\eta|G_{\setminus\eta})$ where η is a subset of the components (nodes and edges) of G and $G_{\setminus\eta}$ is a graph without those components (or equivalently with those components masked out). With these conditional distributions estimated from data, we sample a graph by iteratively updating its components. At each generation iteration, this involves choosing a subset of components, masking them, and sampling new values for them according to the corresponding conditional distribution.

By using conditional distributions, we circumvent the assumptions made by previous approaches to model the unconditional distribution. We do not need to specify an arbitrary order of graph components, unlike in autoregressive models, and learning is exact, unlike in latent variable models. Our approach is inspired by masked language models [15] that model the conditional distribution of masked words given the rest of a sentence, which have shown to be successful in natural language understanding tasks [16, 17, 18, 19, 20, 21] and text generation [22]. As shown in previous works [23, 24, 22], sampling from a trained denoising autoencoder, which is analogous to sampling from our masked graph model, is theoretically equivalent to sampling from the full joint distribution. Therefore, even though we train our model on conditional distributions, sampling repeatedly from these distributions is equivalent to sampling from the full joint distribution of graphs. We use a graph-based model instead of a string-based model as the ability of a language model to model molecules is limited by the string representation used [25]. Directly modeling molecular graphs bypasses the need to find better ways of serializing molecules as strings. It also allows for the use of graph-specific features such as distances between atoms, which are not readily encoded as strings. Developing datasets and benchmarks that incorporate these features would enable more informative comparisons between models that use different molecular representations.

Our approach differs from existing graph-based generative models of molecules, which attempt to directly model the joint distribution. Some of these models follow the autoregressive framework earlier described. Li et al. [26] proposed a deep generative model of graphs that predicts a sequence of transformation operations to generate a graph. You et al. [27] proposed an RNN-based autoregressive generative model that generates components of a graph in breadth-first search (BFS) ordering. To speed up the autoregressive graph generation and improve scalability, Liao et al. [28] extended autoregressive models of graphs by adding blockwise parallel generation. Dai et al. [29] proposed an autoregressive generative model of graphs that utilizes sparsity to avoid generating the full adjacency matrix and generates novel graphs in log-linear time complexity. Grover et al. [30] proposed a VAE-based iterative generative model for small graphs. They restrict themselves to modeling only the graph structure, whereas we consider generating a full graph including node and edge features for molecule generation. Liu et al. [31] proposed a graph neural network model based on normalizing flows for memory-efficient prediction and generation. Mercado et al. [32] proposed a graph neural network-based generative model that learns functions corresponding to whether to add a node to a graph, connect two existing nodes or terminate generation. These learned functions are then used to generate de-novo graphs. The approach requires selecting an ordering of graph components, which the authors choose to be the BFS ordering.

There are also latent variable methods for graph generation. For example, Simonovsky and Komodakis [33] proposed a graph VAE to generate graph representations of molecules. Jin et al. [34] proposed using a VAE to generate a junction tree followed by the generation of the molecule itself. This approach is likely to generate valid chemical structures as it uses a predetermined vocabulary of valid molecular substructures. Kwon et al. [35] proposed a non-autoregressive graph variational autoencoder, which is trained with additional learning objectives to the standard VAE ELBO for unconditional and conditional molecular graph generation.

Along with these works on autoregressive and latent variable generative models of graphs, there is work applying reinforcement learning objectives to the task of molecular graph generation [36, 37, 38] and reaction-driven molecule design [39, 40, 41]. In addition, Yang et al. [42] proposed a target augmentation approach for improving molecular optimisation, a model-agnostic framework that can be used with any black box model. Hence several existing works on generating graph representations of molecules (see Section A of the Supplementary Information for more examples) directly model the joint distribution $p(G)$ or incorporate additional objectives that can be used with a variety of models including our own.

In this work, we evaluate our approach on two popular molecular graph datasets, QM9 [43, 44] and ChEMBL [45], using a set of five distribution-learning metrics introduced in the GuacaMol benchmark [46]: the validity, uniqueness, novelty, KL-divergence [47] (KLD) and Fréchet ChemNet Distance [48] (FCD) scores. The KL-divergence and Fréchet ChemNet

Distance scores are measures of the similarity between generated molecules and molecules from the combined training, validation and test distributions, which we call the dataset distribution. We find that the validity, Fréchet ChemNet Distance and KL-divergence scores are highly correlated with each other and inversely correlated with the novelty score. We show that state-of-the-art autoregressive models are ineffective in controlling the trade-off between novelty and the validity, Fréchet ChemNet Distance, and KL-divergence scores, whereas our masked graph model provides effective control over this trade-off. Overall, the proposed masked graph model, trained on the graph representations of molecules, outperforms previously proposed graph-based generative models of molecules and performs comparably to several SMILES-based models. Additionally, our model achieves comparable performance on validity, uniqueness, and KL-divergence scores compared to state-of-the-art autoregressive SMILES-based models, but with lower Fréchet ChemNet Distance scores. We also carry out conditional generation to obtain molecules with target values of specified physiochemical properties. This involves predicting the masked out components of a molecular graph given the rest of the graph, conditioned on the whole graph having a specified value of the physiochemical property of interest. Example target properties for this approach include the LogP measure of lipophilicity, and molecular weight. We find that our model produces molecules with values close to the target values of these properties without compromising other metrics. Compared with a baseline graph generation approach, the generated molecules maintain physiochemical similarity to the training distribution even as they are optimized for the specified metric. Finally, we find that our method is computationally efficient, needing little time to generate new molecules.

Results

Masked Graph Modeling Overview

A masked graph model (MGM) operates on a graph G, which consists of a set of N vertices $\mathcal{V} = \{v_i\}_{i=1}^{N}$ and a set of edges $\mathcal{E} = \{e_{i,j}\}_{i,j=1}^{N}$. A vertex is denoted by $v_i = (i, t_i)$, where i is the unique index assigned to it, and $t_i \in C_v = \{1, ..., T\}$ is its type, with T the number of node types. An edge is denoted by $e_{i,j} = (i, j, r_{i,j})$, where i, j are the indices to the incidental vertices of this edge and $r_{i,j} \in C_e = \{1, ..., R\}$ is the type of this edge, with R the number of edge types.

We use a single graph neural network to parameterize any conditional distribution induced by a given graph. We assume that the missing components η of the conditional distribution $p(\eta|G_{\setminus\eta})$ are conditionally independent of each other given $G_{\setminus\eta}$:

$$p(\eta|G_{\setminus\eta}) = \prod_{v\in\mathcal{V}} p(v|G_{\setminus\eta}) \prod_{e\in\mathcal{E}} p(e|G_{\setminus\eta}), \tag{1}$$

where $\mathcal{V}$ and $\mathcal{E}$ are the sets of all vertices and all edges in η respectively.

To train the model, we use fully observed graphs from a training dataset D. We corrupt each graph G with a corruption process $C(G_{\setminus\eta}|G)$, i.e. $G_{\setminus\eta} \sim C(G_{\setminus\eta}|G)$. In this work, following the work of Devlin et al. [15] for language models, we randomly replace some of the node and edge features with the special symbol MASK. After passing $G_{\setminus\eta}$ through our model we obtain the conditional distribution $p(\eta|G_{\setminus\eta})$. We then maximize the log probability $\log p(\eta|G_{\setminus\eta})$ of the masked components η given the rest of the graph $G_{\setminus\eta}$. This is analogous to a masked language model [15], which predicts the masked words given the corrupted version of a sentence. This results in the following optimization problem:

$$\arg\max_{\theta} \mathbb{E}_{G\sim D}\mathbb{E}_{G_{\setminus\eta}\sim C(G_{\setminus\eta}|G)} \log p_{\theta}(\eta|G_{\setminus\eta}). \tag{2}$$

Once we have trained the model, we use it to carry out generation. To begin generation, we initialize a molecule in one of two ways, corresponding to different levels of entropy. The first way, which we call training initialization, uses a random graph from the training data as an initial graph. The second way, which we call marginal initialization, initializes each graph component according to a categorical distribution over the values that component takes in our training set. For example, the probability of an edge having type $r \in C_e$ is equal to the fraction of edges in the training set of type r.

We then use an approach motivated by Gibbs sampling to update graph components iteratively from the learned conditional distributions. At each generation step, we sample uniformly at random a fraction α of components η in the graph and replace the values of these components with the MASK symbol. We compute the conditional distribution $p(\eta|G_{\setminus\eta})$ by passing the partially masked graph through the model, sampling new values of the masked components according to the predicted distribution, and placing these values in the graph. We repeat this procedure for a total of K steps, where K is a hyperparameter. A schematic of this procedure is given in Supplementary Figure 4.

We carry out conditional generation using a modified version of this approach. We frame this task as generating molecules with a target value of a given physiochemical property. We use the same training and generation procedures as for unconditional generation but with an additional, conditioning, input to the model. This input y is the molecule's graph-level property of interest. During training, y corresponds to the ground-truth value $y^{\star}$ of the molecule's graph-level property of interest. This results in a modified version of statement 2:

$$\arg\max_{\theta} \mathbb{E}_{G\sim D}\mathbb{E}_{G_{\setminus\eta}\sim C(G_{\setminus\eta}|G)} \log p_{\theta}(\eta|G_{\setminus\eta}.y = y^{\star}) \tag{3}$$

During generation, y instead corresponds to the target value $\hat{y}$ of this property. The initialisation process is the same as for unconditional generation. Iterative sampling involves updating the graph by computing the conditional distribution $p(\eta|G_{\setminus\eta}, y = \hat{y})$.

Mutual Dependence of Metrics from GuacaMol

We evaluate our model and baseline molecular generation models on unconditional molecular generation using the distribution-learning benchmark from the GuacaMol [46] framework. We first attempt to determine whether dependence exists between metrics from the Guacamol framework. We do this because we notice that some of these metrics may measure similar properties. For example, the Fréchet and KL scores are both measures of similarity between generated samples and a dataset distribution. If the metrics are not mutually independent, comparing models using a straightforward measure such as the sum of the metrics may not be a reasonable strategy.

To determine how the five metrics are related to each other, we calculate pairwise the Spearman (rank) correlation between all metrics on the QM9 dataset [43, 44], presented in Table 1, while varying the masking rate, initialization strategy and number of sampling iterations K. We carry out a similar run for three baseline autoregressive SMILES-based models that we train ourselves: two Transformer models [3] with different numbers of parameters(Transformer Small and Transformer Regular) and an LSTM. Each of these autoregressive models has a distribution output by a softmax layer over the SMILES vocabulary at each time step. We implement a sampling temperature parameter in this distribution to control its sharpness. By increasing the temperature, we decrease the sharpness, which increases the novelty. The Spearman correlation results for these baselines are shown in Table 2.

	Validity	Uniqueness	Novelty	KL Div	Fréchet Dist
Validity	1.00	-0.56	-0.83	0.73	0.75
Uniqueness	-0.56	1.00	0.50	-0.32	-0.37
Novelty	-0.83	0.50	1.00	-0.94	-0.95
KL Div	0.73	-0.32	-0.94	1.00	0.99
Fréchet Dist	0.75	-0.37	-0.95	0.99	1.00

Table 1: **Spearman's correlation coefficient between benchmark metrics for results using the masked graph model on the QM9 dataset.**

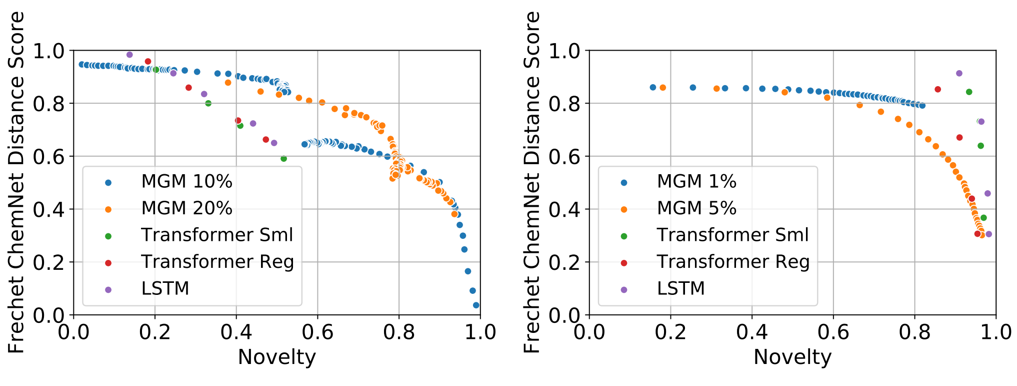
	Validity	Uniqueness	Novelty	KL Div	Fréchet Dist
Validity	1.00	0.03	-0.99	0.98	0.98
Uniqueness	0.03	1.00	0.00	0.03	0.03
Novelty	-0.99	0.00	1.00	-0.99	-0.99
KL Div	0.98	0.03	-0.99	1.00	1.00
Fréchet Dist	0.98	0.03	-0.99	1.00	1.00

Table 2: **Spearman's correlation coefficient between benchmark metrics for results using LSTM, Transformer Small and Transformer Regular on the QM9 dataset.**

From Tables 1 and 2, we make three observations. First, the validity, KL-divergence and Fréchet Distance scores correlate highly with each other. Second, these three metrics correlate negatively with the novelty score. Finally, uniqueness does not correlate strongly with any other metric. These observations suggest that we can look at a subset of the metrics, namely the uniqueness, Fréchet and novelty scores, to gauge generation quality. We now carry out experiments to determine how well MGM and baseline models perform on the anti-correlated Fréchet and novelty scores, which are representative of four of the five evaluation metrics.

Analysis of Representative Metrics

To examine how the masked graph model and baseline autoregressive models balance the Fréchet ChemNet Distance and novelty scores, we plot these two metrics against each other in Figure 1. To obtain the points for the masked graph models, we evaluate the scores after various numbers of generation steps. For the QM9 MGM points, we use both training and marginal initializations, which start from the top left and bottom right of the graph



(a) QM9 (b) ChEMBL

Figure 1: **Plots of the Fréchet ChemNet Distance score against novelty, two anti-correlated metrics from the GuacaMol [46] distribution-learning benchmark, on QM9 and ChEMBL.** Each point corresponds to the values of these two metrics for a set of molecules that are generated using the same model with the same generation hyperparameters. Different points of the same color correspond to different sets of molecules, with each set generated from the same model using different generation hyperparameters (number of generation iterations and masking rate for the masked graph models, sampling temperature for autoregressive models). The percentages indicated next to MGM in the figure legends indicate the masking rate at generation time. (For example, MGM 10% indicates an MGM model with a generation masking rate of 10%.) For each QM9 MGM model, the series of points originating at the top left of the graph corresponds to training initialization, whereas the series of points originating at the bottom right corresponds to marginal initialization. For ChEMBL, only training initialization was used to sample valid molecules due to computational constraints, as marginal initialization did not yield enough valid molecules to calculate reliable distributional metrics in a reasonable amount of time. This is likely because the masking rate is low so it would take a long time for the sampler to converge to the training distribution. Using a high masking rate would result in a large number of spurious edges, which would be problematic for the MPNN to handle. Finding a way to alleviate this issue would be a valuable direction for future work.

respectively, and converge in between. For the ChEMBL MGM points, we use only training initialization.

On both QM9 and ChEMBL, we see that as novelty increases, the Fréchet ChemNet Distance score decreases for the masked graph models as well as for the LSTM and Transformer models. We also see that the line's slope, which represents the marginal change in Fréchet ChemNet Distance score per unit change in novelty score, has a lower magnitude for the masked graph model than for the autoregressive models. This shows that our model trades off novelty for similarity to the dataset distributions (as measured by the Fréchet score) more effectively relative to the baseline models. This gives us a higher degree of controllability in generating samples that are optimized towards either metric to the extent desired.

On QM9, we see that our masked graph models with a 10% or 20% masking rate maintain a larger Fréchet ChemNet Distance score as the novelty increases, compared to the LSTM and Transformer models. Several of the MGM points on the plot are beyond the Pareto

	Model	Valid	Uniq	Novel	KL Div	Fréchet Dist
SMILES	CharacterVAE	0.103	0.675	0.900	N/A	N/A
	GrammarVAE	0.602	0.093	0.809	N/A	N/A
	LSTM (ours)	0.980	0.962	0.138	0.998	0.984
	Transformer Sml (ours)	0.947	0.963	0.203	0.987	0.927
	Transformer Reg (ours)	0.965	0.957	0.183	0.994	0.958
Graph	GraphVAE	0.557	0.760	0.616	N/A	N/A
	MolGAN	0.981	0.104	0.942	N/A	N/A
	NAT GraphVAE (ours)	0.875	0.317	0.895	0.843	0.509
	MGM (ours proposed)	0.886	0.978	0.518	0.966	0.842

Table 3: **Distributional results on QM9.** CharacterVAE [49], GrammarVAE [50], GraphVAE [33] and MolGAN [51] results are taken from Cao and Kipf [51]. NAT GraphVAE [35] stands for non-autoregressive graph VAE. Models labelled as 'ours' were trained by us and subsequently used to carry out generation. Our masked graph model results correspond to a 10% masking rate and training graph initialization, which has the highest geometric mean for all five benchmark metrics. (See Supplementary Information Sections B and C for details.) Values of validity(↑), uniqueness(↑), novelty(↑), KL Div(↑) and Fréchet Dist(↑) metrics are between 0 and 1.

frontier formed by each baseline model. On ChEMBL, the LSTM and Transformer models generally achieve a higher combination of novelty and Fréchet ChemNet Distance score than does the masked graph model with either masking rate. However, to the bottom right of Figure 1b, we can see a few points corresponding to the 5% masking rate that are beyond the Pareto frontier of the points formed by the Transformer Regular model.

We also observe that for ChEMBL, which contains larger molecules, using a 1% masking rate yields points that are beyond the Pareto frontier of those obtained using a 5% masking rate. This further indicates that masking a large number of components hurts generation quality, even if this number represents a small percentage of the graph.

We plot validity against novelty in Supplementary Figure 3 and observe that the same analysis holds for the trade-off between these two metrics. Hence even though state-of-the-art autoregressive models can trade off between representative metrics by changing the sampling strategy, the trade-off is poor and leads to a rapid decline in molecule quality as the novelty increases. MGM, on the other hand, is able to maintain a similar molecule quality as the novelty increases.

Comparison with Baseline Models

We now compare distributional benchmark results for MGM using our 'best' initialization strategy and masking rate (see Supplementary Information Sections B and C for details) to baseline models. The baseline models include models we train ourselves and those for which we obtain results from the literature. The distributional benchmark results on QM9 and ChEMBL are shown in Table 3 and Table 4 respectively.

On QM9, our model performs comparably to existing SMILES-based methods. Our approach shows higher validity and uniqueness scores compared to CharacterVAE [49] and GrammarVAE [50], while having a lower novelty score. Compared to the autoregressive LSTM and Transformer models, our model has lower validity, KL-divergence and Fréchet

	Model	Valid	Uniq	Novel	KL Div	Fréchet Dist
SMILES	AAE	0.822	1.000	0.998	0.886	0.529
	ORGAN	0.379	0.841	0.687	0.267	0.000
	VAE	0.870	0.999	0.974	0.982	0.863
	LSTM	0.959	1.000	0.912	0.991	0.913
	Transformer Sml (ours)	0.920	0.999	0.939	0.968	0.859
	Transformer Reg (ours)	0.961	1.000	0.846	0.977	0.883
Graph	Graph MCTS	1.000	1.000	0.994	0.522	0.015
	NAT GraphVAE	0.830	0.944	1.000	0.554	0.016
	MGM (ours proposed)	0.849	1.000	0.722	0.987	0.845

Table 4: **Distributional results on ChEMBL.** LSTM, Graph MCTS [52], AAE [53], ORGAN [54] and VAE [49] (with a bidirectional GRU [55] as encoder and autoregressive GRU [55] as decoder) results are taken from Brown et al. [46]. NAT GraphVAE [35] stands for non-autoregressive graph VAE. Models labelled as 'ours' were trained by us and subsequently used to carry out generation. Our masked graph model results correspond to a 1% masking rate and training graph initialization, which has the highest geometric mean for all five benchmark metrics. (See Supplementary Information Sections B and C for details.) Values of validity(↑), uniqueness(↑), novelty(↑), KL Div(↑) and Fréchet Dist(↑) metrics are between 0 and 1.

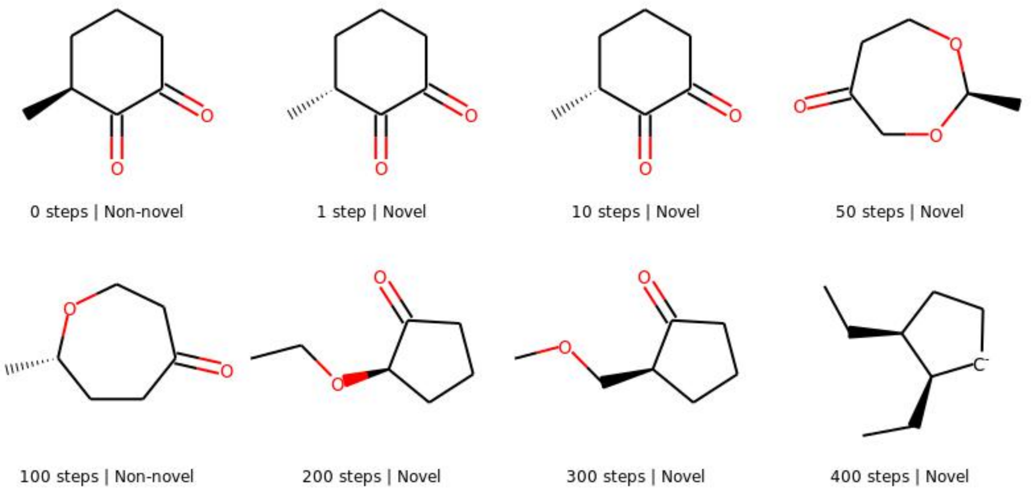
Distance scores; however it exhibits slightly higher uniqueness and significantly higher novelty scores.

Compared to the graph-based models, our approach performs similarly to or better than existing approaches. Our approach has higher validity and uniqueness scores compared to GraphVAE [33] and MolGAN [51], and a lower novelty score. KLD and Fréchet Distance scores are not provided for these two models. Our model outperforms the non-autoregressive graph VAE [35] on all metrics except novelty.

On ChEMBL, our approach outperforms existing graph-based methods. Compared to graph MCTS [52] and non-autoregressive graph VAE [35], our approach shows lower novelty scores while having significantly higher KL-divergence and Fréchet Distance scores. The baseline graph-based models do not capture the properties of the dataset distributions, as shown by their low KL-divergence scores and almost-zero Fréchet scores. This demonstrates that our proposed approach outperforms graph-based methods in generating novel molecules that are similar to the dataset distributions.

The proposed masked graph model is competitive with models that rely on the SMILES representations of molecules. It outperforms the GAN-based model (ORGAN) across all five metrics and outperforms the adversarial autoencoder model (AAE) on all but the uniqueness score (both have the maximum possible score) and the novelty score. It performs comparably to the VAE model with an autoregressive GRU [55] decoder on all metrics except novelty. Our approach lags behind the LSTM, Transformer Small and Transformer Regular SMILES-based models on the ChEMBL dataset. It outperforms both Transformer models on KL-divergence score but underperforms them on validity, novelty and Fréchet score. Our approach also results in lower scores across most of the metrics when compared to the LSTM model.

Some examples of generated molecules after the final sampling iteration are shown in Supplementary Figures 6 and 7. Full lists of molecules can be accessed via the Data Availability section.



(a) Training initialization
(b) Marginal initialization

Figure 2: **Generation trajectory of a molecule each for training initialization and marginal initialization, for QM9 with a 10% masking rate.**

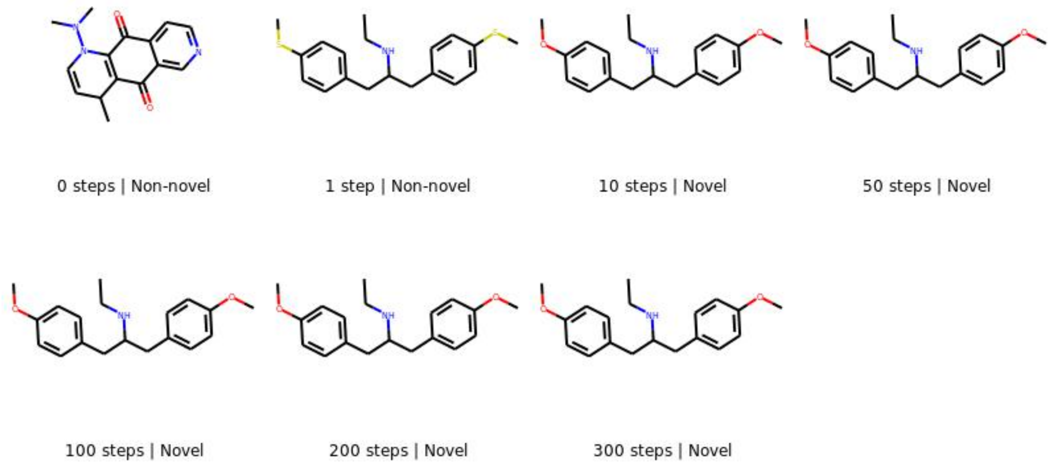
Generation Trajectories

We present a few sampling trajectories of molecules from the proposed masked graph model in Figures 2–3. Each image represents the molecule after a certain number of sampling iterations; the first image in a figure is the molecular graph initialization before any sampling steps are taken. Figure 2 shows a trajectory each for training and marginal initializations with a 10% masking rate. Figure 3 shows a trajectory each for 1% and 5% masking rates with training initialization. All molecules displayed in the figures are valid, but molecules corresponding to some of the intermediate steps not shown may not be.

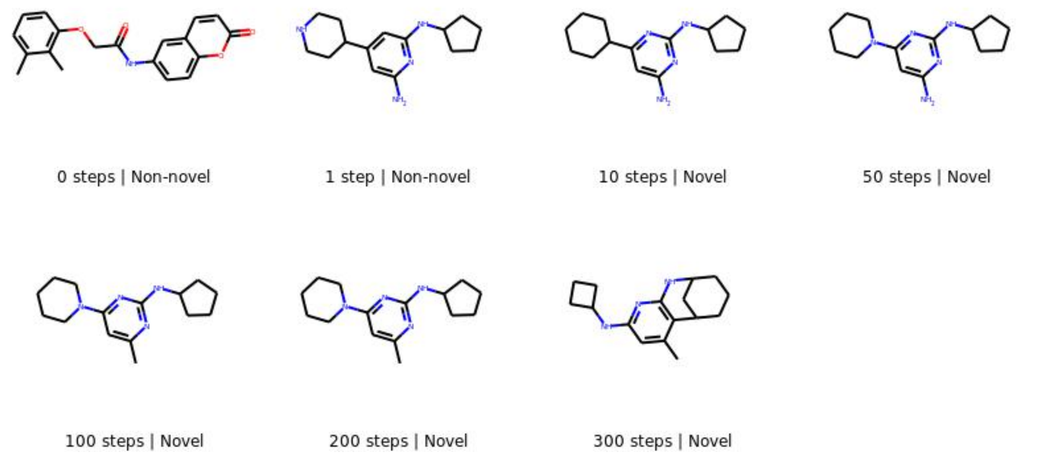
Figure 2a shows the trajectory of a molecule initialized as a molecule from the QM9 training set. As generation progresses, minor changes are made to the molecule, yielding novel molecules. After 100 generation steps, the molecule has converged to another non-novel molecule. Further generation steps yield novel molecules once again, with the molecule's structure gradually moving further away from the initialized molecule.

Figure 2b shows the trajectory of a molecule initialized from the marginal distribution of the QM9 training set. The initialized graph consists of multiple disjoint molecular fragments. Over the first three generation steps, the various nodes are connected to form a connected graph. These changes are more drastic than those in the first few steps of generation with training initialization. The molecule undergoes significant changes over the next few steps until it forms a ring and a chiral center by the 10-th step. The molecule then evolves slowly until it converges to a non-novel molecule by 200 steps. Further generation steps yield a series of novel molecules once again.

Figure 3a shows the trajectory of a ChEMBL molecule with a 1% masking rate. In the first step, the molecule changes from one training molecule to another non-novel molecule,



(a) 1% masking rate



(b) 5% masking rate

Figure 3: **Generation trajectory of a molecule each for a 1% and 5% masking rate, for ChEMBL with training initialization.**

following which it undergoes minor changes over the next few steps to yield a novel molecule. Figure 3b shows the trajectory of a ChEMBL molecule with a 5% masking rate. In the first step, this molecule also changes from one training molecule to another non-novel molecule. Following this, further changes yield a novel molecule. The molecule evolves again in further iterations, albeit forming unexpected ring structures after 300 steps.

Target Condition	Model	G-mean	Unique Count	Property Value	KLD Score
MolWt = 120	NAT GraphVAE	0.623	3048	124.47 ± 7.58	0.843
	MGM	0.522	8800	120.02 ± 7.66	0.811
	MGM - Final Step	0.404	8509	119.42 ± 7.67	0.761
	Dataset	-	-	-	0.679
MolWt = 125	NAT GraphVAE	0.565	2326	127.21 ± 7.05	0.827
	MGM	0.561	9983	125.00 ± 8.48	0.850
	MGM - Final Step	0.354	9293	122.48 ± 7.20	0.936
	Dataset	-	-	-	0.835
MolWt = 130	NAT GraphVAE	0.454	1204	129.12 ± 6.79	0.614
	MGM	0.501	9465	128.85 ± 8.85	0.705
	MGM - Final Step	0.369	8892	126.85 ± 7.43	0.789
	Dataset	-	-	-	0.695
LogP = -0.4	NAT GraphVAE	0.601	2551	-0.409 ± 0.775	0.739
	MGM	0.424	9506	-0.349 ± 0.503	0.803
	MGM - Final Step	0.300	9495	-0.337 ± 0.523	0.876
	Dataset	-	-	-	0.811
LogP = 0.2	NAT GraphVAE	0.562	2188	0.051 ± 0.746	0.803
	MGM	0.378	9524	0.200 ± 0.468	0.846
	MGM - Final Step	0.376	9487	0.202 ± 0.462	0.895
	Dataset	-	-	-	0.816
LogP = 0.8	NAT GraphVAE	0.515	1837	0.588 ± 0.759	0.807
	MGM	0.418	9360	0.769 ± 0.473	0.826
	MGM - Final Step	0.300	9294	0.745 ± 0.442	0.857
	Dataset	-	-	-	0.797

Table 5: **Conditional generation results on QM9.** Results for MGM are chosen from a range of sampling iterations and both initialization strategies. The results shown here correspond to the best mean property value (MGM) or the final sampling iteration with initialisation chosen according to the better geometric mean among the five GuacaMol metrics (MGM - Final Step). Results for the NAT GraphVAE baseline model [35] that we trained are also shown. 'Dataset' rows refer to molecules sampled from the dataset with MolWt within ±1 for the MolWt conditions and LogP within ±0.1 for the LogP conditions. G-mean refers to the geometric mean of validity, uniqueness and novelty.

Conditional Generation

In accordance with the framework proposed by Kwon et al. [35], we generate molecules conditioned on three different target values of the molecular weight (MolWt) and Wildman-Crippen partition coefficient (LogP) properties. We also compute KLD scores for the generated molecules. The KLD score is expected to decrease compared to unconditional generation since MolWt and LogP are two of the properties used to calculate this score; as these properties become skewed towards the target values, the similarity to the dataset will decrease. If a model maintains a reasonably high KLD score while achieving a mean property value close to the target value, it indicates that the other physiochemical properties of the generated molecules are similar to those of the dataset molecules. Conditional generation results for our model and the baseline Kwon et al. [35] model are shown in Table 5.

MGM generates molecules with property values close to the target value of the desired property. For the MolWt=120, MolWt=125, LogP=0.2 and LogP=0.8 conditions, the mean target property of the molecules generated by MGM is closer to the target value than of those generated by NAT GraphVAE. For the MolWt=130 and LogP=-0.4 conditions, the mean is slightly further. For LogP, MGM has lower standard deviations whereas for MolWt, NAT GraphVAE has slightly lower standard deviations. A lower standard deviation corresponds to more reliable generation of molecules with the target property. The G-means of validity, uniqueness and novelty are similar for both models on MolWt, and better for NAT GraphVAE on LogP.

The molecules generated by MGM have similar properties to the dataset molecules. This is reflected by the KL-divergence scores, which are generally higher for MGM than for NAT GraphVAE and greater than 0.8 in all cases but one. The KLD scores in the Dataset rows of Table 5 are considerably less than 1, showing the decrease in similarity to the full dataset as the MolWt or LogP values are skewed. MGM achieves a higher KL-divergence score than Dataset in the majority of cases. This indicates that MGM produces molecules that are optimized for the target property while maintaining physiochemical similarity to the dataset distribution.

The results for MGM - Final Step approach slightly differ from those for MGM. By design, the mean values of the target properties are a little further from the target values than for MGM. Compared with MGM, the standard deviations and G-means for MGM - Final Step are generally lower while the KL-divergence scores are higher.

Computational Efficiency

Time taken to train and generate from models is shown in Table 6. For each sample, generation time per sampling iteration is low (on the order of milliseconds), as the forward pass through the neural network is computationally cheap and many molecules can be processed in parallel. The ChEMBL model takes longer than the QM9 model for generation as it has more MPNN layers and also because ChEMBL molecules are on average larger than QM9 molecules. The ChEMBL model takes longer to train per epoch than the QM9 model for the same reasons and also because ChEMBL has many more molecules than QM9. Note that training time for ChEMBL could be significantly lowered if dynamic batching strategies are used so that the batch size is not constrained by the size of the largest molecule in the dataset. See the Datasets and Evaluation part of the Methods section for more details on the datasets. We use one Nvidia Tesla P100-SXM2 GPU with 16 GB of memory for all our experiments; the use of multiple GPUs or a GPU with larger memory would further increase computational speed.

Discussion

In this work, we propose a masked graph model for molecular graphs. We show that we can sample novel molecular graphs from this model by iterative sampling of subsets of graph

Dataset	Training time per epoch (min)	Generation time/sample/sampling iteration (sec)
QM9	6	0.00542
ChEMBL	280	0.00622

Table 6: **Time taken for training and generation.** Generation time/sample/sampling iteration is measured as: $\frac{\text{time taken to carry out 100 sampling iterations for a batch of J samples}}{100J}$. For QM9, $J = 2500$ whereas for ChEMBL, $J = 1500$ due to memory constraints.

components. Our proposed approach models the conditional distribution of subsets of graph components given the rest of the graph, avoiding many of the previously proposed models' drawbacks such as expensive marginalization and fixing an ordering of variables.

We evaluate our approach on the GuacaMol distribution-learning benchmark on the QM9 and ChEMBL datasets. We find that the benchmark metrics are correlated with each other, so models and generation configurations with higher validity, KL-divergence and Fréchet ChemNet Distance scores usually have lower novelty scores. Hence evaluating models based on the trade-off between different metrics may be more informative than evaluating them based on a heuristic such as the sum of the metrics. We observe that by varying generation hyperparameters, our model balances these metrics more efficiently than previous state-of-the-art baseline models.

For some applications, it is convenient to evaluate results based on one masking rate rather than evaluating this trade-off. A discussion of how to choose this generation hyperparameter is given under the Model Architecture, Training and Unconditional Generation Details part of the Methods section. We recommend using a generation masking rate corresponding to masking out 5-10 edges of a complete graph having the median number of nodes in the dataset.

We show that on distribution-learning metrics, overall our model outperforms baseline graph-based methods. We also observe that our model is comparable to SMILES-based approaches on both datasets, but underperforms the LSTM, Transformer Small and Transformer Regular SMILES-based autoregressive models on ChEMBL. There are several differences between the QM9 and ChEMBL datasets (see the Datasets and Evaluation part of the Methods section) that could account for this, including number of molecules, median molecule size and presence of chirality information. There has also been extensive work in developing language models compared to graph neural networks, which may account for the greater success of the LSTM and Transformers. Furthermore, the ChEMBL dataset is provided as SMILES strings and the GuacaMol benchmark requires that graph representations be converted into SMILES strings before evaluation. This may advantage approaches that work with SMILES strings directly rather than converting to and from graph representations of molecules. Although there are molecular benchmarks for evaluating different aspects of machine learning-based molecular generation [56, 46], they use string representations of molecules and do not evaluate graph-level properties. Developing datasets and benchmarks that incorporate graph-level information that is not readily encoded as strings, such as spatial information, would alleviate this issue. We leave further investigation into the reasons behind the difference in performance to future work.

From our observations of molecular trajectories, we see that molecules converge towards the space of dataset molecules regardless of whether training or marginal initialization is used. This verifies that the sampler produces molecules from the distribution that it was trained on. We also see that using a higher masking rate results in greater changes between sampling iterations and molecules that are less similar to the dataset used. We hypothesize that this is the case for two reasons. First, a greater proportion of the graph is updated at each step. Second, the predictive distributions are formed from a graph with a greater proportion of masked components, resulting in higher entropy.

We carry out conditional generation, observing that our model captures the target properties of molecules better than a baseline graph-based generative model while maintaining similarity of the generated molecules to the distribution of dataset molecules.

Finally, we observe the computational cost of our models and note that generation time per molecule is low after training the model.

Future avenues of work include incorporating additional information such as inter-atomic distances into our graph representations. In the GuacaMol benchmark [46], for example, the data is provided as strings and must be converted back into strings for evaluation. Hence features that are not readily encoded as strings are not used by either the text-based or graph-based models, and cannot be a part of evaluation. The development of benchmarks that account for the spatial nature of molecules, for example by incorporating 3D coordinates, would help highlight the advantages of graph-based generative models compared to SMILES-based models.

As discussed in the Model Architecture, Training and Unconditional Generation Details part of the Methods section, using the same masking rate for molecules of different sizes results in a disproportionately large number of 'prospective' edges being masked out for large molecules, which is problematic for our MPNN to handle. Finding a way to address this problem would be beneficial in scaling this work to larger molecules.

Another direction is to make our model semi-supervised. This would allow us to work with target properties for which the ground-truth cannot be easily calculated at test time and only a few training examples are labelled. Our work can also be extended to proteins, with amino acids as nodes and a contact map as an adjacency matrix. Conditional generation could be used in this framework to redesign proteins to fulfil desired functions. Furthermore, although we use the principle of denoising a corrupted graph for learning the joint distribution, the same procedure could be adapted for lead optimization. Finally, as our approach is broadly applicable to generic graph structures, we leave its application to non-molecular datasets to future work.

Methods

Model Architecture

A diagram of our model including featurization details is given in Figure 4. We start by embedding the vertices and edges in the graph $G_{\setminus\eta}$ to get continuous representations $h_{v_i} \in \mathbb{R}^{d_0}$ and $h_{e_{i,j}} \in \mathbb{R}^{d_0}$ respectively, where d_0 is the dimensionality of the continuous representation space [57]. We then pass these representations to a message passing neural network (MPNN) [58]. We use an MPNN as the fundamental component of our model because of its invariance to graph isomorphism. An MPNN layer consists of an aggregation step that aggregates messages from each node's neighboring nodes, followed by an update step that uses the aggregated messages to update each node's representation. We stack L layers on top of each other to build an MPNN; parameters are tied across all L layers. For all except the last layer, the updated node and edge representations output from layer l are fed into layer $l + 1$. Unlike the original version of the MPNN, we also maintain and update each edge's representation at each layer. Any variant of a graph neural network that effectively models the relationships between node and edge features can be used, such as an MPNN. Our specific design is described below.

Diagrams of our MPNN's node and edge update steps are given in Supplementary Figure 5. At each layer l of the MPNN, we first update the hidden state of each node v_i by computing its accumulated message $u_{v_i}^{(l)}$ using an aggregation function J_v and a spatial residual connection R between neighboring nodes:

$$\begin{aligned} u_{v_i}^{(l)} &= J_v(h_{v_i}^{(l-1)}, \{h_{v_j}^{(l-1)}\}_{j \in N(i)}, \{h_{e_{i,j}}^{(l-1)}\}_{j \in N(i)}) + R(\{h_{v_j}^{(l-1)}\}_{j \in N(i)}), \\ J_v(h_{v_i}^{(l-1)}, \{h_{v_j}^{(l-1)}\}_{j \in N(i)}, \{h_{e_{i,j}}^{(l-1)}\}_{j \in N(i)}) &= \sum_{j \in N(i)} h_{e_{i,j}}^{(l-1)} \cdot h_{v_j}^{(l-1)}, \\ R(\{h_{v_j}^{(l-1)}\}_{j \in N(i)}) &= \sum_{j \in N(i)} h_{v_j}^{(l-1)}, \\ h_{v_i}^{(l)} &= \text{LayerNorm}(\text{GRU}(h_{v_i}^{(l-1)}, u_{v_i}^{(l)})), \end{aligned}$$

where $N(i)$ is the set of indices corresponding to nodes that are in the one-hop neighbourhood of node v_i. GRU [55] refers to a gated recurrent unit which updates the representation of each node using its previous representation and accumulated message. LayerNorm [59] refers to layer normalization.

Similarly, the hidden state of each edge $h_{e_{i,j}}$ is updated using the following rule for all $j \in N(i)$:

$$h_{e_{i,j}}^{(l)} = J_e(h_{v_i}^{(l-1)} + h_{v_j}^{(l-1)}).$$

The sum of the two hidden representations of the nodes incidental to the edge is passed through J_e, a two-layer fully connected network with ReLU activation between the two layers [60, 61], to yield a new hidden edge representation.

The node and edge representations from the final layer are then processed by a node projection layer $A_v : \mathbb{R}^{d_0} \to \Lambda^T$ and an edge projection layer $A_e : \mathbb{R}^{d_0} \to \Lambda^R$, where Λ^T and Λ^R are probability simplices over node and edge types respectively. The result is the distributions $p(v|G_{\setminus\eta})$ and $p(e|G_{\setminus\eta})$ for all $v \in \mathcal{V}$ and all $e \in \mathcal{E}$.

Property Embeddings

Node Property Embeddings We represent each node using six node properties indexed as $\{\kappa \in \mathbb{Z} : 1 \leq \kappa \leq 6\}$, each with its own one-hot embedding. The properties are obtained using RDKit [62]. Each node in a graph corresponds to a heavy atom in a molecule. During the forward pass, each of these embeddings is multiplied by a separate weight matrix $W_\kappa \in \mathbb{R}^{T_\kappa \times d_0}$, where T_κ is the number of categories for property κ. The resulting continuous embeddings are summed together to form an overall embedding of the node. The entries of the one-hot embeddings for each of the properties are:

- **Atom type:** chemical symbol (e.g. C, N, O) of the atom;
- **Number of hydrogens:** number of hydrogen atoms bonded to the atom;
- **Charge:** net charge on the atom, where the first index represents the minimum charge on an atom in the dataset and the last index represents the maximum;

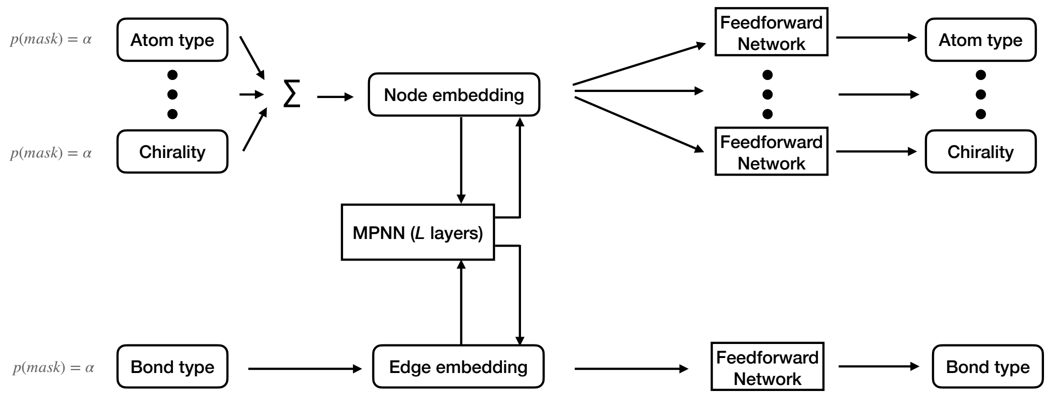


Figure 4: **Model architecture.** A description of the node and edge features is given in the Property Embeddings part of the Methods section.

- **Chirality type:** unspecified, tetrahedral clockwise, tetrahedral counter-clockwise, other;
- **Is-in-ring:** atom is or is not part of a ring structure;
- **Is-aromatic:** atom is or is not part of an aromatic ring.

Each one-hot embedding also has an additional entry corresponding to the MASK symbol.

After processing the graph with the MPNN, we pass the representation of each node through six separate fully-connected two-layer networks with ReLU activation between the layers. For each node, the output of each network is a distribution over the categories of the initial one-hot vector for one of the properties. During training, we calculate the cross-entropy loss between the predicted distribution and the ground-truth for all properties that were masked out by the corruption process.

The choice of nodes for which a particular property is masked out is independent of the choice made for all other properties. The motivation for this is to allow the model to more easily learn relationships between different property types. The atom-level property information that we use in our model is the same as that provided in the SMILES string representation of a molecule. We also tried masking out all features for randomly selected nodes, but this yielded a significantly higher cross-entropy loss driven largely by the atom type and hydrogen terms.

Since the ChEMBL dataset does not contain chirality information, the chirality type embedding is superfluous for ChEMBL.

We note from preliminary experiments that using fewer node features, specifically only the atom type and number of hydrogens, results in a substantially higher cross-entropy loss than using all the node features listed above.

Edge Property Embeddings We use the same framework as described for node property embeddings. We only use one edge property with the weight matrix $\mathcal{W} \in \mathbb{R}^{R \times d_0}$, whose one-hot embedding is defined as follows:

- **Bond type:** no, single, double, triple or aromatic bond.

Model Architecture, Training and Unconditional Generation Details

For the QM9 dataset, we use one 4-layer MPNN, with parameter sharing between layers. For the ChEMBL dataset, we use one 6-layer MPNN with parameter sharing. We experiment with using more layers for ChEMBL in case more message passing iterations are needed to cover a larger graph. The results of an extensive hyperparameter search on ChEMBL are given in Supplementary Table 2. For both datasets, we use an embedding dimensionality $d_0 = 2048$. We use the Adam optimizer [63] with learning rate set to 0.0001, $\beta_1 = 0.9$ and $\beta_2 = 0.98$. We use a batch size of 800 molecules for QM9 and 512 molecules for ChEMBL. For ChEMBL, we perform 16 forward-backward steps with minibatches of 32 each to compute the gradient of the minibatch of 512 molecules, in order to cope with the limited memory size on a GPU. We clip the gradient for its norm to be at most 10.

During training, we uniformly at random mask each node feature (including atom type) and edge feature (including bond type) with probability α, while randomly varying α uniformly between 0 and 0.2. Nodes are considered as neighbors in the MPNN if they are connected by an edge that is either masked out, or does not have bond type no-bond. For the purposes of masking, the total number of edges in the graph is $\frac{|V|(|V|-1)}{2}$ i.e. every possible node pair (excluding self-loops) in the symmetric graph is considered as a 'prospective edge' that can be masked out. During validation, we follow the same procedure but with α fixed at 0.1, so that we can clearly compare model checkpoints and choose the checkpoint with the lowest validation loss for generation.

For QM9, we carry out generation experiments while using a masking rate of either 10% or 20%, corresponding to the mean and maximum masking rates during training respectively. For ChEMBL, we use a masking rate of either 1% or 5%, as we found that the higher masking rates led to low validity scores in our preliminary experiments. The number of prospective edges masked and replaced for a median ChEMBL molecule with a 1% masking rate and for a median QM9 molecule with a 10% masking rate are both approximately 4. This indicates that the absolute number rather than portion of components masked out directly impacts generation quality. For a constant masking rate, the number of masked out prospective edges scales as the square of the number of nodes in the graph. The number of bonds in a molecule does not scale in this way; larger molecules are likely to have sparser adjacency matrices than small molecules. Masking out a very large number of prospective edges could degrade performance as this would yield an unnaturally dense graph to the MPNN. This is because every prospective edge of type 'no edge' that is masked out would appear as an edge to the MPNN. This would result in message passing between many nodes in the input graph that are far apart in the sparse molecule. We therefore propose a masking rate corresponding to masking out a similar number of prospective edges (approximately 5-10) when using MGM on other datasets. Nevertheless, finding an automated way of setting the masking rate would be a valuable direction for future research.

We use the same independence constraint during generation as we use during training when choosing which properties to mask out for each node or edge. We vary the initialization strategy between training and marginal initialization.

For QM9, we run 400 sampling iterations sequentially to generate a sequence of sampled graphs. For ChEMBL, we run 300 iterations. We calculate the GuacaMol evaluation metrics for our samples after every generation step for the first 10 steps, and then every 10-20 steps, in order to observe how generation quality changes with the number of generation steps.

Conditional Generation Details

We carry out conditional generation corresponding to two different molecular properties: molecular weight (MolWt) and the Wildman-Crippen partition coefficient (LogP). We train a separate model for each property on QM9, with the same hyperparameters as used for the unconditional case. For each property, we first normalize the property values by subtracting the mean and dividing by the standard deviation across the training data. We obtain an embedding of dimension d_0 for the property by passing the one-dimensional standardised property value through a two-layer fully-connected network with ReLU activation between the two layers. We add this embedding to each node embedding and then proceed with the forward pass as in the unconditional case. For generation, we use a 10% masking rate and carry out 400 sampling iterations with both training and marginal initializations.

We evaluate 10,000 generated molecules using the framework outlined by Kwon et al. [35] in their work on non-autoregressive graph generation. This involves computing summary statistics of the generated molecules for target property values of 120, 125 and 130 for MolWt, and -0.4, 0.2 and 0.8 for LogP. We choose results corresponding to the initialization and number of sampling iterations that yield the mean property value that is closest to the target value. We also provide results from the final generation step with the initialization corresponding to the higher geometric mean among the five GuacaMol metrics.

Finally, we calculate KLD scores for molecules from the QM9 dataset with property values close to the target values. For the MolWt conditions, we sample 10,000 molecules from the dataset that have a MolWt within 1 of the target MolWt. For the LogP conditions, we sample 10,000 molecules from the dataset that have LogP value within 0.1 of the target LogP value.

Details of Baseline Models

We train two variants of the Transformer [3] architecture: Small and Regular. The Transformer Regular architecture consists of 6 layers, 8 attention heads, embedding size of 1024, hidden dimension of 1024, and dropout of 0.1. The Transformer Small architecture consists of 4 layers, 8 attention heads, embedding size of 512, hidden dimension of 512, and dropout of 0.1. Both Transformer-Small and -Regular are trained with a batch size of 128 until the validation cross-entropy loss stops improving. We set the learning rate of the Adam optimizer to 0.0001, $\beta_1 = 0.9$ and $\beta_2 = 0.98$. The learning rate is decayed based on the inverse square root of the number of updates. We use the same hyperparameters for the Transformer Small and Regular models on both QM9 and ChEMBL.

We follow the open-source implementation of the GuacaMol benchmark baselines at `https://github.com/BenevolentAI/guacamol_baselines` for training an LSTM model on QM9. Specifically, we train the LSTM with 3 layers of hidden size 1024, dropout of 0.2 and batch size of 64, using the Adam optimizer with learning rate 0.001, $\beta_1 = 0.9$ and $\beta_2 = 0.999$. We do not train the rest of the baseline models ourselves. For QM9: CharacterVAE [49], GrammarVAE [50], GraphVAE [33], and MolGAN [51] results are taken from Cao and Kipf [51]. For ChEMBL: AAE [6], ORGAN [54], Graph MCTS [52], VAE, and LSTM results are taken from Brown et al. [46]. NAT GraphVAE results are taken from Kwon et al. [35] for ChEMBL. To carry out unconditional and conditional generation from NAT GraphVAE on QM9, we train a model using the publicly available codebase provided by the paper's authors at `https://github.com/seokhokang/graphvae_approx` .

Datasets and Evaluation

We evaluate our approach using two widely used [49, 33, 64] datasets of small molecules: QM9 [43, 44], and a subset of the ChEMBL database [45] (Version 24) as defined by Fiscato et al. [65] and used by Brown et al. [46]. All references to ChEMBL in this paper are references to this subset of the database. Heavy atoms and bonds in a molecule correspond to nodes and edges in a graph, respectively.

The QM9 dataset consists of approximately 132,000 molecules with a median and maximum of 9 heavy atoms each. Each atom is of one of the following $T = 4$ types: C, N, O, and F. Each bond is either a no-bond, single, double, triple or aromatic bond ($R = 5$). The ChEMBL dataset contains approximately 1,591,000 molecules with a median of 27 and a maximum of 88 heavy atoms each. It contains 12 types of atoms ($T = 12$): B, C, N, O, F, Si, P, S, Cl, Se, Br, and I. Each bond is either a no-bond, single, double, triple or aromatic bond ($R = 5$).

The QM9 dataset is split into training and validation sets, while the ChEMBL dataset is split into training, validation and test sets. We use the term dataset distribution to refer to the distribution of the combined training and validation sets for QM9, and the combined training, validation and test sets for ChEMBL. Similarly, we use the term dataset molecule to refer to a molecule from the combined QM9 or ChEMBL dataset.

To numerically evaluate our approach, we use the GuacaMol benchmark [46], a suite of benchmarks for evaluating molecular graph generation approaches. The GuacaMol framework operates on SMILES strings, so we convert our generated graphs to SMILES strings before evaluation. Specifically, we evaluate our model using distribution-learning metrics from GuacaMol: the validity, uniqueness, novelty, KL-divergence [47] and Fréchet ChemNet Distance [48] scores. GuacaMol uses 10,000 randomly sampled molecules to calculate each of these scores. Validity measures the ratio of valid molecules, uniqueness estimates the proportion of generated molecules that remain after removing duplicates and novelty measures the proportion of generated molecules that are not dataset molecules. The KL-divergence score compares the distributions of a variety of physiochemical descriptors estimated from the dataset and a set of generated molecules. The Fréchet ChemNet Distance score [48] measures the proximity of the distribution of generated molecules to the distribution of the dataset molecules. This proximity is measured according to the Fréchet Distance in the hidden representation space of ChemNet, which is trained to predict the chemical properties of small molecules [66].

Data Availability

Data, pretrained models and lists of generated molecules can be found via `https://github.com/nyu-dl/dl4chem-mgm` .

Code Availability

Code, training and generation scripts for MGM and baseline models can be found at `https://github.com/nyu-dl/dl4chem-mgm` .

References

[1] Regine S. Bohacek, Colin Mcmartin, and Wayne C. Guida. The art and practice of structure-based drug design: A molecular modeling perspective. *Medicinal Research Reviews*, 16(1):3–50, 1996. doi: 10.1002/(sici)1098-1128(199601)16:1<3::aid-med1>3.3.co;2-d.

[2] S. Hochreiter and J. Schmidhuber. Long short-term memory. *Neural Computation*, 9: 1735–1780, 1997.

[3] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, L. Kaiser, and Illia Polosukhin. Attention is all you need. *ArXiv*, abs/1706.03762, 2017.

[4] Diederik P. Kingma and Max Welling. Auto-encoding variational bayes. *arXiv preprint 1312.6114*, 2013.

[5] Danilo Jimenez Rezende, S. Mohamed, and Daan Wierstra. Stochastic backpropagation and approximate inference in deep generative models. In *ICML*, 2014.

[6] Alireza Makhzani, Jonathon Shlens, Navdeep Jaitly, and Ian J. Goodfellow. Adversarial autoencoders. *ArXiv*, abs/1511.05644, 2015.

[7] Ian J. Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron C. Courville, and Yoshua Bengio. Generative adversarial nets. In *NIPS*, 2014.

[8] D. Elton, Zois Boukouvalas, Mark D. Fuge, and P. W. Chung. Deep learning for molecular design—a review of the state of the art. 2019.

[9] Tomas Mikolov, Stefan Kombrink, Anoop Deoras, Lukas Burget, and Jan Cernocky. Rnnlm - recurrent neural network language modeling toolkit. 2011.

[10] Yoshua Bengio and Samy Bengio. Modeling high-dimensional discrete data with multi-layer neural networks. In *NIPS*, pages 400–406, 1999. URL `http://papers.nips.cc/paper/1679-modeling-high-dimensional-discrete-data-with-multi-layer-neural-networks`.

[11] Hugo Larochelle and Iain Murray. The neural autoregressive distribution estimator. In *The Proceedings of the 14th International Conference on Artificial Intelligence and Statistics*, volume 15 of *JMLR: W&CP*, pages 29–37, 2011.

[12] Marwin H. S. Segler, Thierry Kogej, Christian Tyrchan, and Mark P. Waller. Generating focussed molecule libraries for drug discovery with recurrent neural networks. *CoRR*, abs/1701.01329, 2017. URL `http://arxiv.org/abs/1701.01329`.

[13] Oriol Vinyals, Samy Bengio, and Manjunath Kudlur. Order matters: Sequence to sequence for sets. *arXiv preprint arXiv:1511.06391*, 2016.

[14] Raphael Shu, Jason Lee, Hideki Nakayama, and Kyunghyun Cho. Latent-variable non-autoregressive neural machine translation with deterministic inference using a delta posterior, 2019.

[15] Jacob Devlin, Ming-Wei Chang, and Kristina Toutanova Kenton Lee. Bert: Pre-training of deep bidirectional transformers for language understanding. In *NAACL*, 2019.

[16] Alex Wang, Amanpreet Singh, Julian Michael, F. Hill, Omer Levy, and Samuel R. Bowman. Glue: A multi-task benchmark and analysis platform for natural language understanding. *ArXiv*, abs/1804.07461, 2018.

[17] Alex Wang, Yada Pruksachatkun, Nikita Nangia, Amanpreet Singh, Julian Michael, F. Hill, Omer Levy, and Samuel R. Bowman. Superglue: A stickier benchmark for general-purpose language understanding systems. *ArXiv*, abs/1905.00537, 2019.

[18] Rodrigo Nogueira and Kyunghyun Cho. Passage re-ranking with bert. *ArXiv*, abs/1901.04085, 2019.

[19] Y. Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, M. Lewis, Luke Zettlemoyer, and Veselin Stoyanov. Roberta: A robustly optimized bert pretraining approach. *ArXiv*, abs/1907.11692, 2019.

[20] Zhenzhong Lan, Mingda Chen, Sebastian Goodman, Kevin Gimpel, Piyush Sharma, and Radu Soricut. Albert: A lite bert for self-supervised learning of language representations. *ArXiv*, abs/1909.11942, 2020.

[21] Guillaume Lample and Alexis Conneau. Cross-lingual language model pretraining. *ArXiv*, abs/1901.07291, 2019.

[22] Elman Mansimov, Alex Wang, and Kyunghyun Cho. A generalized framework of sequence generation with application to undirected sequence models. *arXiv preprint arXiv:1905.12790*, 2019.

[23] Guillaume Alain and Yoshua Bengio. What regularized auto-encoders learn from the data generating distribution, 2014.

[24] Pascal Vincent, Hugo Larochelle, Isabelle Lajoie, Yoshua Bengio, and Pierre-Antoine Manzagol. Stacked denoising autoencoders: Learning useful representations in a deep network with a local denoising criterion. *J. Mach. Learn. Res.*, 11:3371–3408, December 2010. ISSN 1532-4435.

[25] Mario Krenn, Florian Häse, AkshatKumar Nigam, Pascal Friederich, and Alán Aspuru-Guzik. Self-referencing embedded strings (selfies): A 100% robust molecular string representation, 2019.

[26] Yujia Li, Oriol Vinyals, Chris Dyer, Razvan Pascanu, and Peter W. Battaglia. Learning deep generative models of graphs. *arXiv preprint arXiv:1803.03324*, 2018.

[27] Jiaxuan You, Rex Ying, Xiang Ren, William L. Hamilton, and Jure Leskovec. Graphrnn: Generating realistic graphs with deep auto-regressive models. In *ICML*, 2018.

[28] Renjie Liao, Yujia Li, Yang Song, Shenlong Wang, William L. Hamilton, David Duvenaud, Raquel Urtasun, and Richard S. Zemel. Efficient graph generation with graph recurrent attention networks. In *NeurIPS*, 2019.

[29] Hanjun Dai, Azade Nazi, Yujia Li, Bo Dai, and D. Schuurmans. Scalable deep generative modeling for sparse graphs. *ArXiv*, abs/2006.15502, 2020.

[30] Aditya Grover, Aaron Zweig, and Stefano Ermon. Graphite: Iterative generative modeling of graphs. In *ICML*, 2019.

[31] Jenny Liu, Aviral Kumar, Jimmy Ba, Jamie Kiros, and Kevin Swersky. Graph normalizing flows. In *NeurIPS*, 2019.

[32] Rocío Mercado, Tobias Rastemo, Edvard Lindelöf, Günter Klambauer, Ola Engkvist, Hongming Chen, and Esben Jannik Bjerrum. Graph networks for molecular design, Aug 2020. URL `https://chemrxiv.org/articles/preprint/Graph_Networks_for_Molecular_Design/12843137/1`.

[33] Martin Simonovsky and Nikos Komodakis. Graphvae: Towards generation of small graphs using variational autoencoders. *arXiv preprint 1802.03480*, 2018.

[34] Wengong Jin, Regina Barzilay, and Tommi S. Jaakkola. Junction tree variational autoencoder for molecular graph generation. In *ICML*, 2018.

[35] Youngchun Kwon, Jiho Yoo, Y. Choi, Won joon Son, Dongseon Lee, and Seokho Kang. Efficient learning of non-autoregressive graph variational autoencoders for molecular graph generation. *Journal of Cheminformatics*, 11, 2019.

[36] Jiaxuan You, B. Liu, Rex Ying, V. Pande, and J. Leskovec. Graph convolutional policy network for goal-directed molecular graph generation. *ArXiv*, abs/1806.02473, 2018.

[37] Zhenpeng Zhou, Steven M. Kearnes, L. Li, Richard N. Zare, and P. Riley. Optimization of molecules via deep reinforcement learning. *Scientific Reports*, 9, 2019.

[38] Gregor N. C. Simm, Robert Pinsler, and José Miguel Hernández-Lobato. Reinforcement learning for molecular design guided by quantum mechanics, 2020.

[39] Ling Wang, Chengyun Zhang, Renren Bai, Jianjun Li, and Hongliang Duan. Heck reaction prediction using a transformer model based on a transfer learning strategy. *Chem. Commun.*, 56:9368–9371, 2020. doi: 10.1039/D0CC02657C. URL `http://dx.doi.org/10.1039/D0CC02657C`.

[40] Bowen Liu, Bharath Ramsundar, Prasad Kawthekar, Jade Shi, Joseph Gomes, Quang Luu Nguyen, Stephen Ho, Jack Sloane, Paul Wender, and Vijay Pande. Retrosynthetic reaction prediction using neural sequence-to-sequence models. *ACS Central Science*, 3(10):1103–1113, 2017. doi: 10.1021/acscentsci.7b00303. URL `https://doi.org/10.1021/acscentsci.7b00303`. PMID: 29104927.

[41] John Bradshaw, Brooks Paige, Matt J. Kusner, Marwin H. S. Segler, and José Miguel Hernández-Lobato. Barking up the right tree: an approach to search over molecule synthesis dags, 2020.

[42] Kevin Yang, Wengong Jin, Kyle Swanson, Regina Barzilay, and Tommi Jaakkola. Improving molecular design by stochastic iterative target augmentation, 2020.

[43] Lars Ruddigkeit, Ruud Van Deursen, Lorenz C. Blum, and Jean-Louis Reymond. Enumeration of 166 billion organic small molecules in the chemical universe database GDB-17. *J. Chem. Inf. Model.*, 52(11):2864–2875, 2012.

[44] Raghunathan Ramakrishnan, Pavlo O. Dral, Matthias Rupp, and O. Anatole Von Lilienfeld. Quantum chemistry structures and properties of 134 kilo molecules. *Sci. Data*, 1: 140022:1–7, 2014.

[45] Anna Gaulton, Anne Hersey, Michał Nowotka, A. Patrícia Bento, Jon Chambers, David Mendez, Prudence Mutowo, Francis Atkinson, Louisa J. Bellis, Elena Cibrián-Uhalte, Mark Davies, Nathan Dedman, Anneli Karlsson, María Paula Magariños, John P. Overington, George Papadatos, Ines Smit, and Andrew R. Leach. The ChEMBL database in 2017. *Nucleic Acids Research*, 45(D1):D945–D954, 11 2016.

[46] Nathan Brown, Marco Fiscato, Marwin H.S. Segler, and Alain C. Vaucher. Guacamol: Benchmarking models for de novo molecular design. *arXiv preprint 1811.09621*, 2018.

[47] S. Kullback and R. A. Leibler. On information and sufficiency. *Annals of Mathematical Statistics*, 22:79–86, 1951.

[48] Kristina Preuer, P. Renz, Thomas Unterthiner, Sepp Hochreiter, and G. Klambauer. Fréchet chemnet distance: A metric for generative models for molecules in drug discovery. *Journal of chemical information and modeling*, 2018.

[49] Rafael Gómez-Bombarelli, David Duvenaud, José Miguel Hernández-Lobato, Jorge Aguilera-Iparraguirre, Timothy D. Hirzel, Ryan P. Adams, and Alán Aspuru-Guzik. Automatic chemical design using a data-driven continuous representation of molecules. *arXiv preprint 1610.02415*, 2016.

[50] Matt J. Kusner, Brooks Paige, and José Miguel Hernández-Lobato. Grammar variational autoencoder. In *ICML*, 2017.

[51] Nicola De Cao and Thomas Kipf. Molgan: An implicit generative model for small molecular graphs. *arXiv preprint 1805.11973*, 2018.

[52] Jan H. Jensen. Graph-based genetic algorithm and generative model/monte carlo tree search for the exploration of chemical space. 2018.

[53] Daniil Polykovskiy, Alexander Zhebrak, Dmitry Vetrov, Yan Ivanenkov, Vladimir Aladinskiy, Polina Mamoshina, Marine Bozdaganyan, Alexander Aliper, Alex Zhavoronkov, and Artur Kadurin. Entangled conditional adversarial autoencoder for de novo drug discovery. *Molecular Pharmaceutics*, 2018.

[54] Gabriel Lima Guimaraes, Benjamin Sanchez-Lengeling, Pedro Luis Cunha Farias, and Alán Aspuru-Guzik. Objective-reinforced generative adversarial networks (ORGAN) for sequence generation models. *CoRR*, abs/1705.10843, 2017. URL `http://arxiv.org/abs/1705.10843`.

[55] Kyunghyun Cho, Bart Van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. Learning phrase representations using rnn encoder-decoder for statistical machine translation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing*, pages 1724–1734, 2014.

[56] Josep Arús-Pous, Simon Viet Johansson, Oleksii Prykhodko, Esben Jannik Bjerrum, Christian Tyrchan, Jean-Louis Reymond, Hongming Chen, and Ola Engkvist. Randomized smiles strings improve the quality of molecular generative models, Jul 2019. URL `https://chemrxiv.org/articles/preprint/Randomized_SMILES_Strings_Improve_the_Quality_of_Molecular_Generative_Models/8639942/2`.

[57] Yoshua Bengio, Réjean Ducharme, Pascal Vincent, and Christian Janvin. A neural probabilistic language model. *J. Mach. Learn. Res.*, 3(null):1137–1155, March 2003. ISSN 1532-4435.

[58] Justin Gilmer, Samuel S. Schoenholz, Patrick F. Riley, Oriol Vinyals, and George E. Dahl. Neural message passing for quantum chemistry. In *Proceedings of the 34th International Conference on Machine Learning*, pages 1263–1272, 2017.

[59] Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E. Hinton. Layer normalization, 2016.

[60] Vinod Nair and Geoffrey E. Hinton. Rectified linear units improve restricted boltzmann machines. In *ICML*, pages 807–814, 2010. URL `https://icml.cc/Conferences/2010/papers/432.pdf`.

[61] Xavier Glorot, Antoine Bordes, and Yoshua Bengio. Deep sparse rectifier neural networks. volume 15 of *Proceedings of Machine Learning Research*, pages 315–323, Fort Lauderdale, FL, USA, 11–13 Apr 2011. JMLR Workshop and Conference Proceedings. URL `http://proceedings.mlr.press/v15/glorot11a.html`.

[62] Rdkit: Open-source cheminformatics. URL `http://www.rdkit.org`.

[63] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *CoRR*, abs/1412.6980, 2015.

[64] Yibo Li, Liangren Zhang, and Zhenming Liu. Multi-objective de novo drug design with conditional graph generative model. *Journal of Cheminformatics*, 10(1), Jul 2018. ISSN 1758-2946. doi: 10.1186/s13321-018-0287-6. URL `http://dx.doi.org/10.1186/s13321-018-0287-6`.

[65] Marco Fiscato, Alain C. Vaucher, and Marwin Segler. Guacamol all smiles, Nov 2018. URL `https://figshare.com/articles/dataset/GuacaMol_All_SMILES/7322252/2`.

[66] Garrett B. Goh, C. Siegel, A. Vishnu, and Nathan Oken Hodas. Chemnet: A transferable and generalizable deep neural network for small-molecule property prediction. *ArXiv*, abs/1712.02734, 2017.

[67] Hongwei Wang, Jia Wang, Jialin Wang, Miao Zhao, Weinan Zhang, Fuzheng Zhang, Xing Xie, and Minyi Guo. Graphgan: Graph representation learning with generative adversarial nets. *CoRR*, abs/1711.08267, 2017. URL `http://arxiv.org/abs/1711.08267`.

[68] Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. Squad: 100, 000+ questions for machine comprehension of text. *ArXiv*, abs/1606.05250, 2016.

[69] Pranav Rajpurkar, Robin Jia, and Percy Liang. Know what you don't know: Unanswerable questions for squad. *ArXiv*, abs/1806.03822, 2018.

[70] Samuel R. Bowman, Gabor Angeli, Christopher Potts, and Christopher D. Manning. A large annotated corpus for learning natural language inference. *ArXiv*, abs/1508.05326, 2015.

[71] Adina Williams, Nikita Nangia, and Samuel R. Bowman. A broad-coverage challenge corpus for sentence understanding through inference. *ArXiv*, abs/1704.05426, 2018.

[72] Alex Wang and Kyunghyun Cho. Bert has a mouth, and it must speak: Bert as a markov random field language model. *arXiv preprint arXiv:1902.04094*, 2019.

[73] Marjan Ghazvininejad, Omer Levy, Yinhan Liu, and Luke Zettlemoyer. Mask-predict: Parallel decoding of conditional masked language models. *arXiv preprint arXiv:1904.09324*, 2019.

[74] Sean Welleck, Kianté Brantley, Hal Daumé, and Kyunghyun Cho. Non-monotonic sequential text generation. In *ICML*, 2019.

[75] Mitchell Stern, William Chan, J. Kiros, and Jakob Uszkoreit. Insertion transformer: Flexible sequence generation via insertion operations. In *ICML*, 2019.

[76] Jiatao Gu, Qi Liu, and Kyunghyun Cho. Insertion-based decoding with automatically inferred generation order. *Transactions of the Association for Computational Linguistics*, 2019.

[77] Weihua Hu, Bowen Liu, Joseph Gomes, Marinka Zitnik, Percy Liang, Vijay S. Pande, and Jure Leskovec. Strategies for pre-training graph neural networks. *arXiv preprint arXiv:v*, 2019.

[78] Lukasz Maziarka, Tomasz Danel, S. Mucha, K. Rataj, J. Tabor, and Stanislaw Jastrzebski. Molecule attention transformer. *ArXiv*, abs/2002.08264, 2020.

Supplementary Information

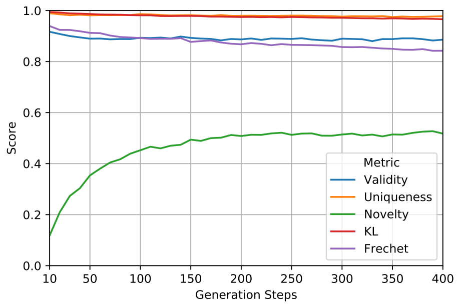
A Related Work

In-Silico Molecular Generation Many of the previously proposed generative models of molecules focused on extending the variational autoencoder (VAE) for molecular generation. Gómez-Bombarelli et al. [49] proposed the first variational autoencoder (VAE) [4] based model for generating molecules in their SMILES representations. To address the issue of VAEs generating syntactically invalid SMILES strings, Kusner et al. [50] explicitly added the grammar of SMILES strings to VAEs for molecule generation. Wang et al. [67], Guimaraes et al. [54] and Cao and Kipf [51] used a generative adversarial network (GAN) [7] to build a generative model of small molecular graphs. Unlike most recent work that has focused on neural network-based approaches, Jensen [52] showed that genetic algorithms based on Monte Carlo Tree Search (MCTS) could be competitive on the task of molecular generation.

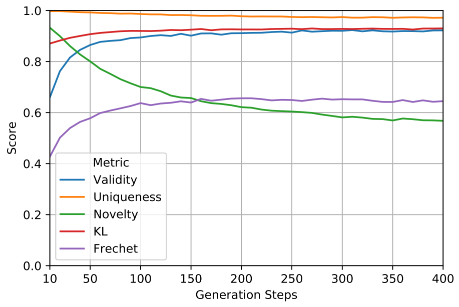
Masked Language Models Masked language models, such as BERT [15], have been shown to bring significant improvements to a variety of discriminative language understanding tasks such as question answering [68, 69] and natural language inference [70, 71]. Wang and Cho [72], Ghazvininejad et al. [73] and Mansimov et al. [22] proposed ways to generate text directly from trained masked language models. Wang and Cho [72] proposed the use of Gibbs sampling, and Mansimov et al. [22] proposed the use of adaptive Gibbs sampling approaches for effective text generation using masked language models. Ghazvininejad et al. [73] used conditional masked language models for parallel decoding in machine translation. They first predict all target words in parallel, and then repeatedly mask out and regenerate the subset of words that the model is least confident about for a fixed number of iterations. In parallel to the work investigating masked language models for text generation, Welleck et al. [74], Stern et al. [75] and Gu et al. [76] proposed methods for non-monotonic sequential text generation. Although these methods could be applied for generating molecular graphs in flexible ordering, there has not been work empirically validating this. Due to the popularity of masked language models in natural language processing tasks, there has been recent work investigating a similar approach for learning graph representations. Hu et al. [77] investigated the transfer to downstream tasks of graph neural networks that were trained to predict the masked node and edge attributes of graphs. Maziarka et al. [78] proposed the molecule attention transformer architecture that was pretrained to predict masked input nodes and investigated its transfer to downstream property prediction tasks. Unlike our work, neither Hu et al. [77] nor Maziarka et al. [78] investigated ways of generating novel molecular graphs with their trained models.

B Effect of Generation Hyperparameters on Generation Quality

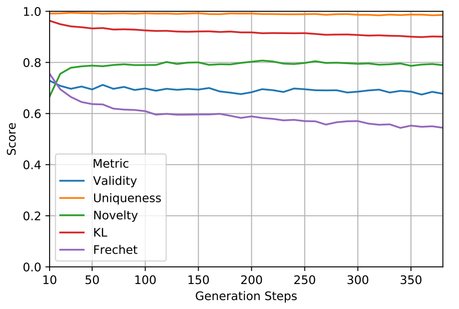
We analyze the effect of changing the masking rate and graph initialization on generation quality. In order to do so, we must choose results corresponding to a certain number of generation steps for each combination of masking rate and initialization. We therefore evaluate samples at intermediate steps of the generation process, as shown in Supplementary Figure 1, to determine how the values of the evaluation metrics change as the number of generation steps increases.



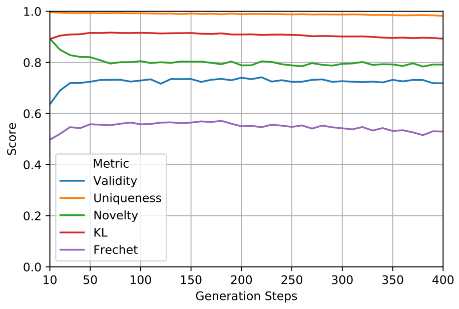
(a) Training initialization, 10% masking rate



(b) Marginal initialization, 10% masking rate



(c) Training initialization, 20% masking rate



(d) Marginal initialization, 20% masking rate

Supplementary Figure 1: **Plots of generation scores as a function of number of generation steps for each initialization and masking rate on QM9.**

For training initialization (Supplementary Figures 1a and 1c), the initialized molecules have perfect validity, uniqueness, KL and Fréchet scores, and zero novelty score. As generation proceeds, changes are made to the training molecules, yielding some invalid molecules, so the validity decreases. Some of the changes yield new, valid molecules, so the novelty increases. These molecules are less similar to the dataset distributions than the training molecules are themselves, so the KL and Fréchet scores decrease. On the other hand, for marginal initializations (Supplementary Figures 1b and 1d), the initialized molecules are less likely to be valid or similar to the dataset molecules. The probability of obtaining duplicate molecules is low as well. Over time, the molecules converge to valid structures similar to the dataset molecules, so the validity, KL and Fréchet scores increase. For both training and marginal initializations, different initialized molecules may converge to the same molecule over time, lowering uniqueness.

For all configurations and all metrics, the slope of the score with respect to the number of generation steps tends to flatten over time. When presenting the results of our model for different masking rates and initializations, we use the benchmark scores at the final generation step.

We now use these results to analyze the effect of changing the masking rate and graph initialization for generation in Supplementary Table 1. On QM9, we find that using marginal initialization leads to slightly higher validity and novelty scores however with lower KL-

Dataset	Mask Rate	Graph Init	Valid	Uniq	Novel	KL Div	Fréchet Dist
QM9	10%	train	0.886	0.978	0.518	0.966	0.842
	10%	marginal	0.922	0.972	0.568	0.930	0.645
	20%	train	0.678	0.988	0.789	0.901	0.544
	20%	marginal	0.719	0.982	0.792	0.893	0.529
ChEMBL	1%	train	0.849	1.000	0.722	0.987	0.845
	5%	train	0.558	1.000	0.952	0.869	0.396

Supplementary Table 1: **Effect of varying masking rate and graph initialization on the benchmark results for our masked graph model on QM9 and ChEMBL.**

divergence and Fréchet ChemNet Distance scores compared with using training initialization. When using marginal initialization, the masked graph model generates marginally more novel molecules at the expense of not capturing the properties of dataset molecules as well. On ChEMBL, the marginal initialization strategy results in validity scores close to 0, which is why we only consider the training initialization strategy in Supplementary Table 1. On both QM9 and ChEMBL, novelty increases significantly when increasing the masking rate while the validity, KL-divergence and Fréchet Distance scores drop.

Close observation of the results in Supplementary Table 1 suggests that the choice of masking rate and initialization strategy impacts the balance among the five metrics. Most significantly, increasing the masking rate results in a higher novelty score, and lower KL-divergence and Fréchet Distance scores. We can trade off between different metrics as desired by adjusting the initialization and masking rate.

C Selecting Best Unconditional Generation Results

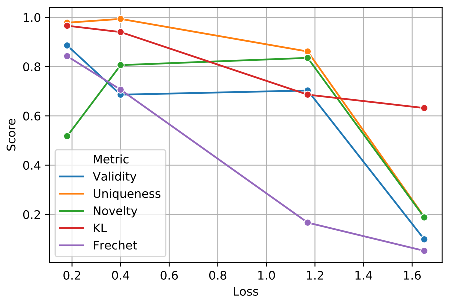
We have shown that the GuacaMol benchmark metrics are correlated and that our model can efficiently trade these metrics off against each other. Thus we cannot say that one generation strategy definitively outperforms another unless it achieves a higher score on each of the five metrics. However, for the sake of comparison with baseline models, we pick one generation strategy as follows: we select results from Supplementary Table 1 for each dataset corresponding to the highest geometric mean among all five metrics.

For QM9, the 'best' MGM results correspond to training initialization with a 10% masking rate. For ChEMBL, the 'best' MGM results correspond to training initialization with a 1% masking rate.

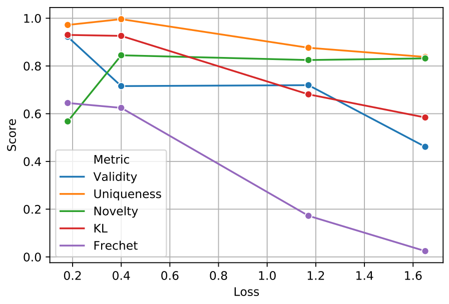
D Effect of Validation Loss on Generation Quality

To determine whether validation loss is a suitable proxy for generation quality, we carry out generation from different training checkpoints of our 'best' QM9 model. During training, we carried out a hyperparameter search to find the configurations with the lowest validation loss, which we used as the criterion to select the best model for generation. The experiments in this subsection explore whether this choice is justified.

Supplementary Figure 2 shows the values of all five benchmark metrics corresponding to different loss values (i.e., different checkpoints) of our model. In general, as the validation



(a) Training Initialization



(b) Marginal Initialization

Supplementary Figure 2: **Benchmark metric results on QM9 corresponding to our model's checkpoints corresponding to different validation loss values.** A masking rate of 10% was used.

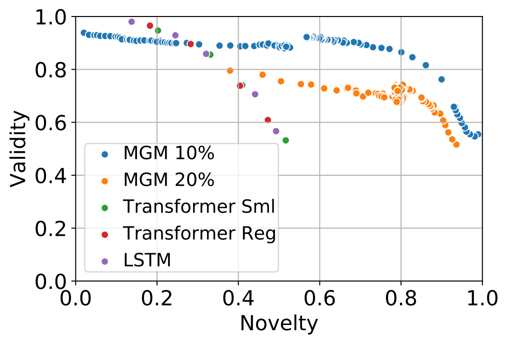
loss increases, the metrics' values decrease. We attribute the decrease in validity to the fact that a less well-trained model is less likely to have learned enough about the relationship between different parts of a graph to predict masked components that respect the chemical constraints inherent in this type of data. The increase in novelty and decrease in KL and Fréchet scores are explained by better-trained models being more likely to predict masked components from the most similar context in the training/validation data. Occasionally this causes our model to generate an exact copy of a molecule from the training dataset, lowering the novelty; in general, it produces molecules whose local neighborhoods are similar to those of molecules in the training/validation data, thereby increasing the KL and Fréchet scores. The sharp decrease in novelty and uniqueness as the loss increases from 1.17 to 1.65 can be attributed to the low validity, as GuacaMol implicitly penalizes all metrics when the validity drops below 0.5.

We conclude that selecting the model with the lowest validation loss for generation is a reasonable strategy. This implies that using more powerful graph neural networks within our *masked graph modeling* framework could improve generation quality. Finding model architectures that lower the validation loss is a good direction for future work.

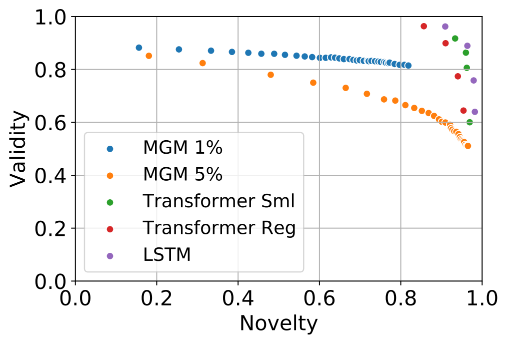
E Further Supplementary Tables and Figures

d_0	MPNNs	Layers per MPNN	Batch Size	Learning Rate	LR Decay	Validation Loss
2048	1	4	100	0.0005	no	0.29
2048	1	4	800	0.0005	no	0.20
2048	**1**	**6**	**512**	**0.0001**	**no**	**0.12**
2048	1	6	512	0.0005	no	0.17
2048	1	6	512	0.005	yes	0.38
2048	1	6	1024	0.0005	no	0.16
2048	1	8	50	0.0005	no	0.32
2048	1	8	100	0.0005	no	0.23
2048	1	8	400	0.0005	no	0.19
2048	1	16	25	0.0005	no	0.40
2048	1	16	100	0.0005	no	0.23
2048	2	2	400	0.0005	no	0.22
2048	2	3	512	0.005	yes	0.38
2048	2	4	400	0.0005	no	0.17
4096	1	4	400	0.0005	no	0.28
4096	1	6	512	0.005	yes	1.00
4096	1	6	1024	0.0001	no	0.15
4096	1	6	1024	0.0005	no	0.19
4096	1	6	2048	0.0005	no	0.19

Supplementary Table 2: **Hyperparameter configurations and corresponding validation set loss on the ChEMBL dataset.** The rows are arranged in ascending order, greedily by column from left to right. LR decay stands for learning rate decay and corresponds to decreasing the learning rate to a minimum of 0.0005 by halving the current learning rate every 204,800 data points. The hyperparameter configuration corresponding to the lowest loss is given in bold font and was used to generate the ChEMBL results presented in this paper.

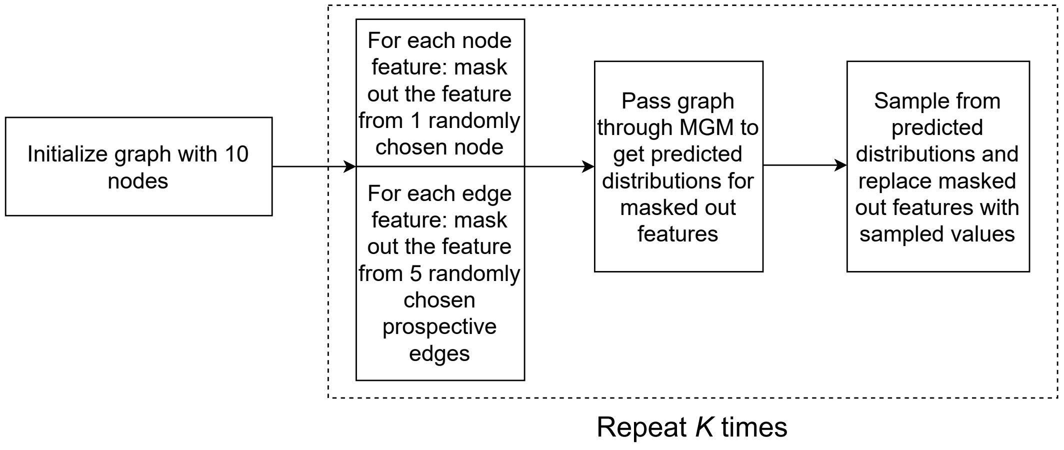


(a) QM9

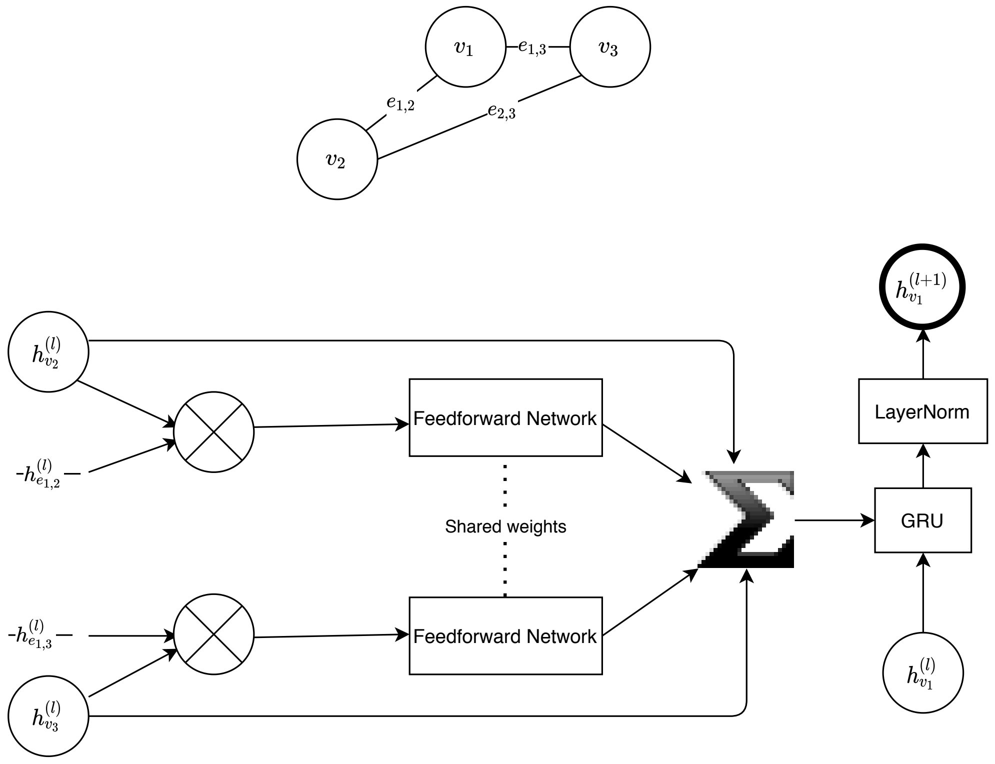


(b) ChEMBL

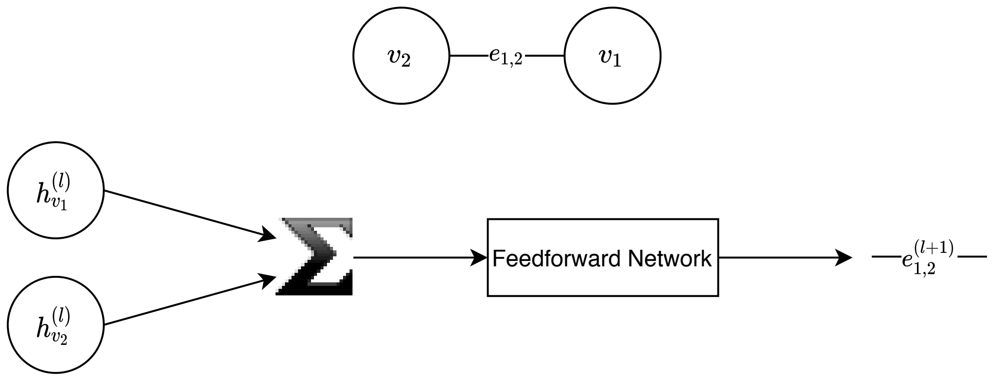
Supplementary Figure 3: **Plots of validity against novelty, two anti-correlated metrics from the GuacaMol [46] distribution-learning benchmark.** The plots are generated in the same way as for Figure 1 in the main text.


Initialize graph with 10 nodes
For each node feature: mask out the feature from 1 randomly chosen node
For each edge feature: mask out the feature from 5 randomly chosen prospective edges
Pass graph through MGM to get predicted distributions for masked out features
Sample from predicted distributions and replace masked out features with sampled values
Repeat K times

Supplementary Figure 4: **Schematic for unconditional generation with an initial graph with 10 nodes and a 10% masking rate.** The initial graph can either be taken from the training set (training initialization) or initialized using the training set distribution (marginal initialization). At each of the K sampling iterations, $\frac{10}{100} * 10 = 1$ node and $\frac{10}{100} * \frac{10(10-1)}{2} \approx 5$ prospective edges are masked out and replaced.

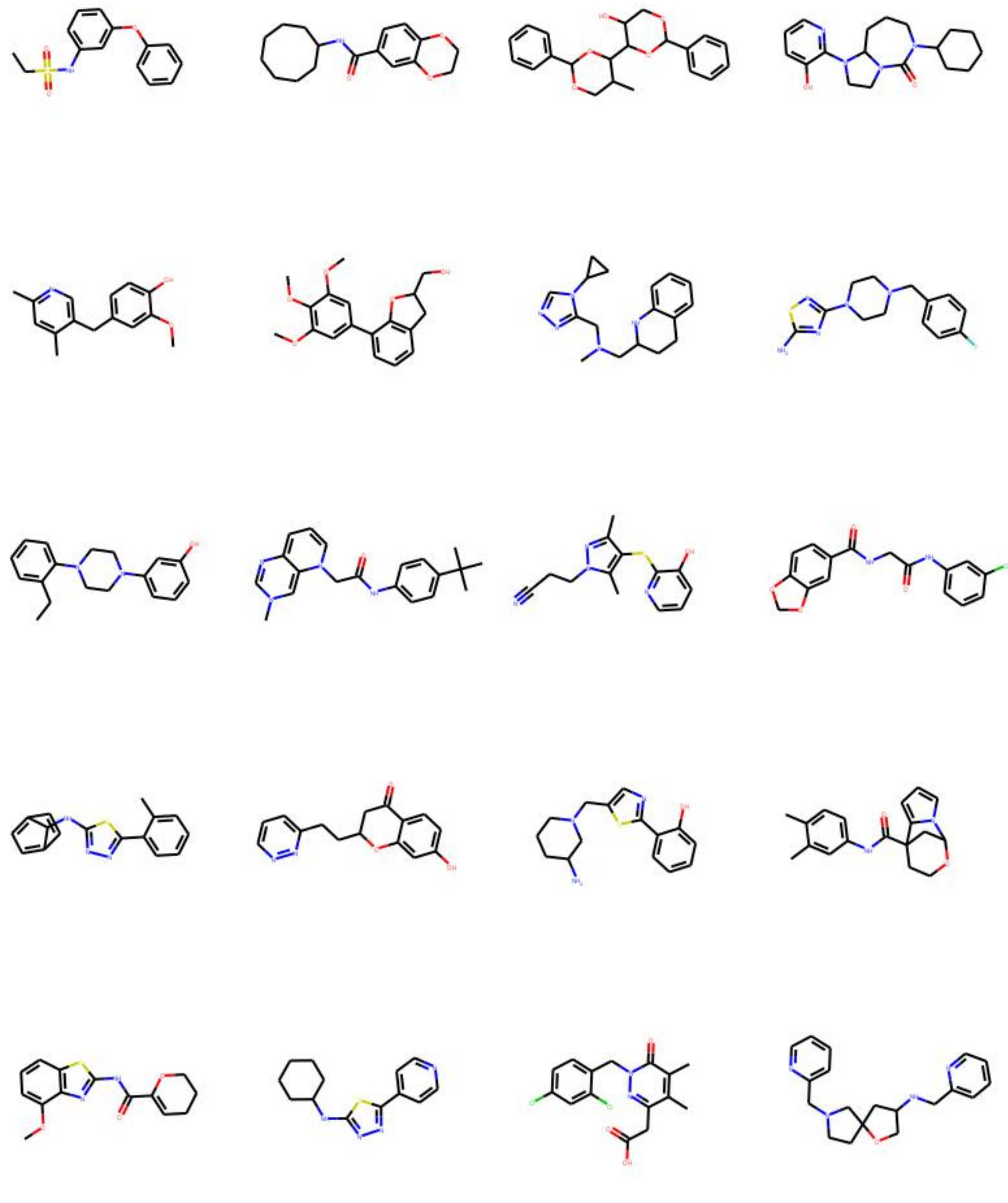


(a) Node Update Step. The diagram shows the calculation of the updated representation of node v_1 in the graph at the top of the figure. $\otimes$ denotes elementwise multiplication.

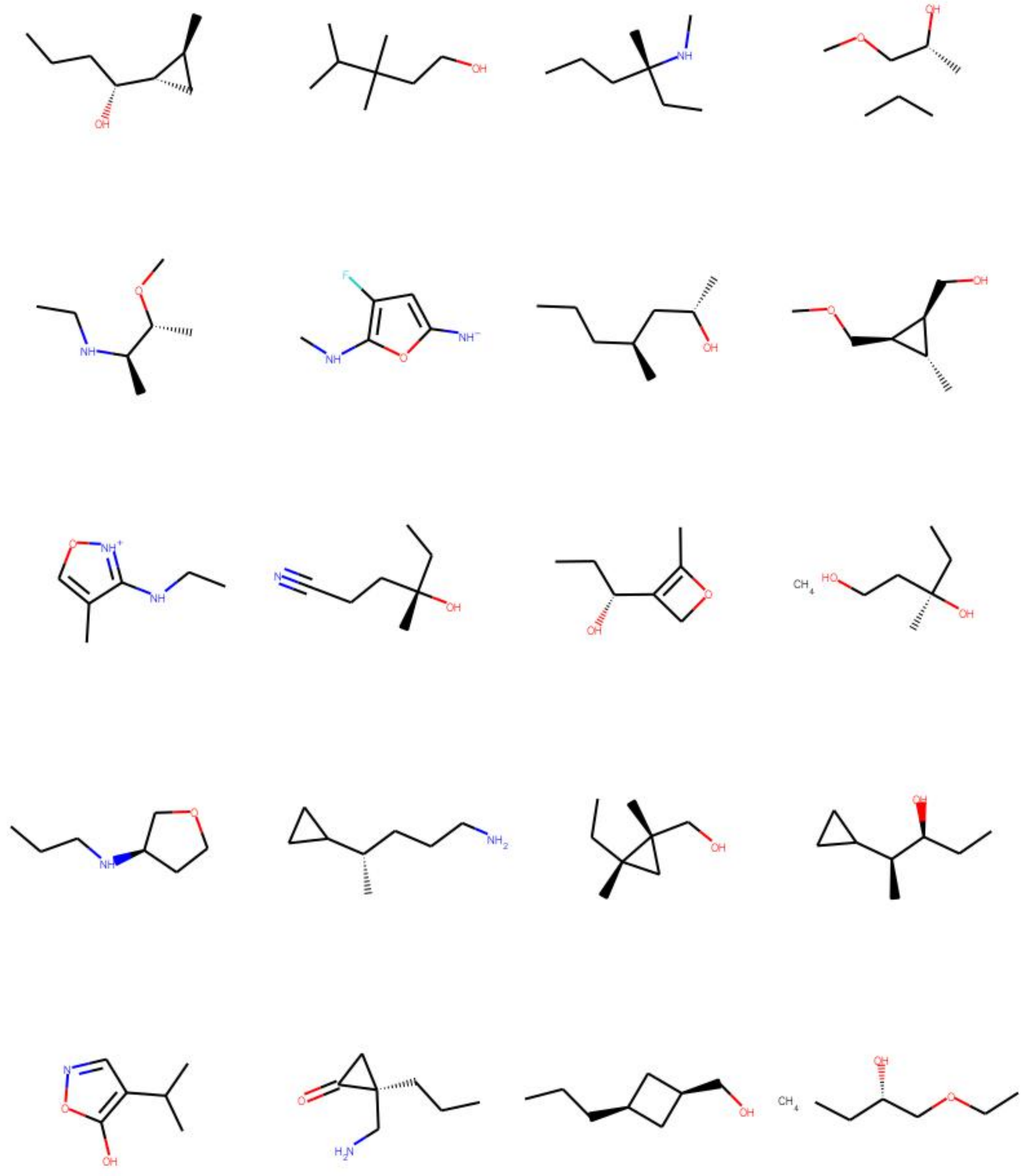


(b) Edge Update Step. The diagram shows the calculation of the updated representation of edge $e_{1,2}^{(l+1)}$ in the graph at the top of the figure.

Supplementary Figure 5: **MPNN update steps**



Supplementary Figure 6: **A selection of unconditionally generated novel molecules from ChEMBL.** The molecules are randomly chosen from the subset of novel generated molecules with QED > 0.9.



Supplementary Figure 7: **A selection of unconditionally generated novel molecules from QM9.** The molecules are randomly chosen from the subset of novel generated molecules with QED > 0.6.