# Masked Graph Modeling for Molecule Generation

Omar Mahmood,[†] Elman Mansimov,[‡] Richard Bonneau,[‡] and Kyunghyun Cho[*,‡]

[†]*Center for Data Science, New York University, 60 5th Avenue, New York, New York 10011, United States*

[‡]*Department of Computer Science, Courant Institute of Mathematical Sciences, New York University, 60 5th Avenue, New York, New York 10011, United States*

E-mail: kyunghyun.cho@nyu.edu

**Abstract**

De novo, in-silico design of molecules is a challenging problem with applications in drug discovery and material design. Here, we introduce a masked graph model which learns a distribution over graphs by capturing all possible conditional distributions over unobserved nodes and edges given observed ones. We train our masked graph model on existing molecular graphs and then sample novel molecular graphs from it by iteratively masking and replacing different parts of initialized graphs. We evaluate our approach on the QM9 and ChEMBL datasets using the distribution-learning benchmark from the GuacaMol framework. The benchmark contains five metrics: the validity, uniqueness, novelty, KL-divergence and Fréchet ChemNet Distance scores, the last two of which are measures of the similarity of the generated samples to the training, validation and test distributions. We find that KL-divergence and Fréchet ChemNet Distance scores are anti-correlated with novelty scores. By varying generation initialization and the fraction of the graph masked and replaced at each generation step, we can increase the Fréchet score at the cost of novelty. In this way, we show that our model offers transparent

and tunable control of the trade-off between these metrics, a key point of control in design applications currently lacking in other approaches to molecular graph generation. Our model outperforms previously proposed graph-based approaches and is competitive with SMILES-based approaches. Finally, we observe that minimizing validation loss on the training task is a suitable proxy for improving generation quality, which shows the suitability of optimizing the training objective for improving generation.

# Introduction

The design of de novo molecules in-silico with desired properties is an essential part of drug discovery and material 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 of molecular graphs. Once the underlying distribution is captured, new molecular graphs are sampled accordingly.

Each of these approaches makes unique assumptions about the underlying probabilistic structure of a molecular graph. Autoregressive models specify an ordering of atoms and bonds in advance to model the graph. Latent variable models such as variational autoencoders and adversarial autoencoders assume the existence of unobserved (latent) variables that capture complicated dependencies among the atoms and bonds. 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.

In this paper, 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. By using conditional distributions, we circumvent the assumptions made by previous approaches to model the unconditional distribution. Our approach is inspired by masked language models[9] 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[10–15] and text generation[16]. We build a model for graphs rather than use a language model because the ability of a language model to model molecules is limited by the string representation used[17]. By directly modeling molecular graphs, we bypass the need to find better ways of serializing molecules as strings.

We evaluate our approach on two popular molecular graph datasets, QM9[18,19] and ChEMBL[20], using a set of five distribution-learning metrics introduced in the GuacaMol benchmark[21]: the validity, uniqueness, novelty, KL-divergence[22] and Fréchet ChemNet Distance[23] scores. After careful analysis, 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 our masked graph model offers higher flexibility than other models by more effectively trading off the novelty for the validity, Fréchet ChemNet Distance, and KL-divergence scores. 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.

In order to verify the effectiveness of our training strategy for generation, we calculate the evaluation metrics for molecules generated from different training checkpoints, which correspond to different validation losses. We find that in general the values of the metrics increase as the validation loss decreases, demonstrating the suitability of the proposed training task for generation.

# Background

We frame the problem of graph generation as sampling a graph $G$ from a distribution $p^\star(G)$ defined over all possible graphs. 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 $f_\theta$ so that $p_\theta(G) = f_\theta(G)$. The parameters $\theta$ are 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 $p^\star(G)$, we approximate $KL(p^\star \| p_\theta)$ by using a training set $D = (G_1, G_2, ..., G_M)$ which consists of samples from $p^\star$. Once we have trained our model on this distribution, we carry out generation by sampling from the trained model.

One powerful approach for parameterizing and sampling from such an unconditional distribution is autoregressive modeling[2,3,24–26]. An autoregressive model decomposes 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[27].

Another approach for building a generative model of graphs is to introduce a set of 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[28]. 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 $\theta$ 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[4–6].

# Model

In this paper, 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. Our approach, to which we refer as *masked graph modeling*, instead parameterizes and learns conditional distributions $p(\eta|G_{\setminus\eta})$ where $\eta$ 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.

There are two advantages to the proposed approach. First, we do not need to specify an arbitrary order of graph components, unlike in autoregressive models. Second, learning is exact, unlike in latent variable models where it is often necessary to maximize a tractable lowerbound instead of the exact likelihood. In the remainder of this section, we describe in detail parameterization, learning and generation.

## Parameterization

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 $\eta$ 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 $\eta$ respectively.

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[29]. We then pass these representations to a message passing neural network (MPNN)[30]. 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.

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[31] refers to a gated recurrent unit which updates the representation of each node using its previous representation and accumulated message. LayerNorm[32] refers to layer normalization.

Similarly, the hidden states of each edge $h_{e_{i,j}}$ are 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[33,34], 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 $\Lambda^T$ and $\Lambda^R$ are probability simplices over node and edge types respectively. The result are the distributions $p(v|G_{\setminus\eta})$ and $p(e|G_{\setminus\eta})$ for all $v \in \mathcal{V}$ and all $e \in \mathcal{E}$.

## Learning

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.[9] 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 $\eta$ given the rest of the graph $G_{\setminus\eta}$. This is analogous to a masked language model[9], which predicts the masked words given the corrupted version of a sentence. This results in the following optimization problem:

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

## 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 $\alpha$ of components $\eta$ 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 $T$ steps, where $T$ is a hyperparameter.

# Methods

We evaluate the proposed masked graph modeling approach for molecular graph generation. Atoms and bonds in a molecule correspond to nodes and edges in a graph, respectively. In this section, we outline the experimental setup used to carry out this evaluation, including datasets, evaluation framework, model details and training and generation procedures.

## Datasets and Evaluation

We evaluate our approach using two widely used[35–37] datasets of small molecules: QM9[18,19] and ChEMBL[20]. 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 $T = 5$ types: B, 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. In the remainder of this paper, 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[21], a suite of benchmarks for evaluating molecular graph generation approaches. Specifically, we evaluate our model using distribution-learning metrics from GuacaMol: the validity, uniqueness, novelty, KL-divergence[22] and Fréchet ChemNet Distance[23] 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[23] 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[38].

## 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. 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 $T_\kappa$ is the number of categories for property $\kappa$. 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;
- **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.

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

### 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 Generation

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 use more layers for ChEMBL because more message passing iterations are needed to cover a larger graph. For both datasets, we use an embedding dimensionality $d_0 = 2048$. We use the Adam optimizer[39] 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.[†] 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 $\alpha$, while randomly varying $\alpha$ 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. During validation, we follow the same procedure but with $\alpha$ 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 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. 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.

## 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

[†]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.

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[‡] 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[35], GrammarVAE[40], GraphVAE[36], and MolGAN[41] results are taken from Cao and Kipf[41]. For ChEMBL: AAE[6], ORGAN[42], Graph MCTS[43], VAE, and LSTM results are taken from Brown et al.[21]. NAT GraphVAE results are taken from Kwon et al.[44] for both QM9 and ChEMBL.

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 | -0.56 | -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 |

[‡]`https://github.com/BenevolentAI/guacamol_baselines`

Table 2: Spearman's correlation coefficient between benchmark metrics for results using LSTM, Transformer Small and Transformer Regular 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 |

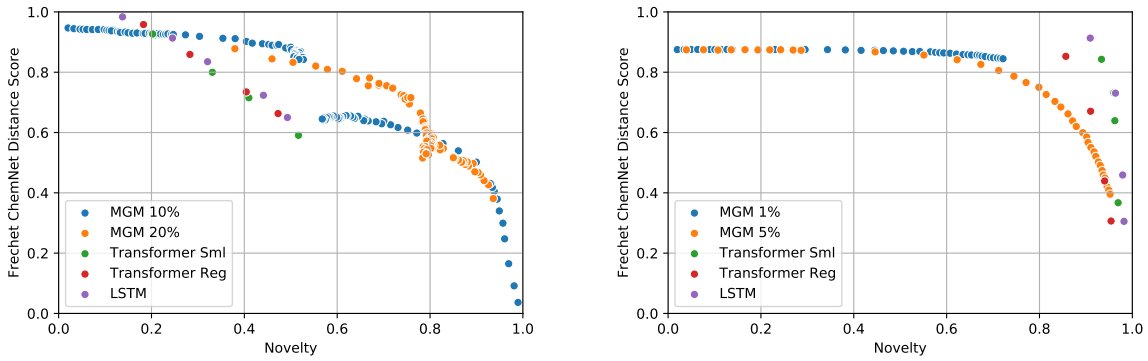
# Results and Discussion

## Mutual Dependence of Metrics from GuacaMol

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 QM9, presented in Table 1, while varying the masking rate, initialization strategy and number of sampling iterations. We carry out a similar run for the Transformer Small, Transformer Regular, and LSTM baselines as follows. 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.

From Tables 1–2, we make three observations. First, the validity, KL-divergence and Fréchet Distance scores correlate highly with each other. Second, these three metrics correlate




(a) QM9 (b) ChEMBL

Figure 1: Plots of the Fréchet ChemNet Distance score against novelty. The plots are generated by varying generation hyperparameters (number of generation iterations for the masked graph models and sampling temperature for autoregressive models).

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. In the next section, we 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. We observe how effectively each model trades these metrics off against each other.

## Analysis of Representative Metrics

To examine how the masked graph model and baseline autoregressive models trade off 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 respectively, and converge in between. For the ChEMBL MGM points, we use only training initialization.

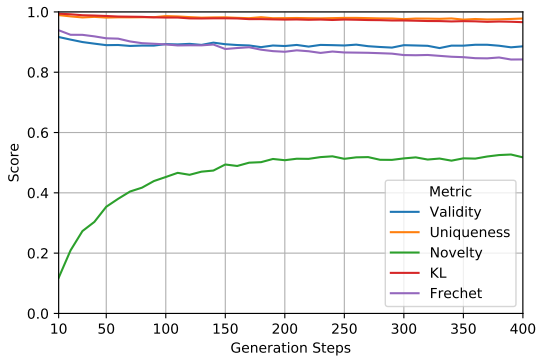
On both QM9 and ChEMBL, we see that as novelty increases, the Fréchet ChemNet Distance score decreases for the masked graph models as well as for the LSTM and Transformer models. We also see that the line's slope, which represents the marginal change in Fréchet ChemNet Distance score per unit change 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 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.

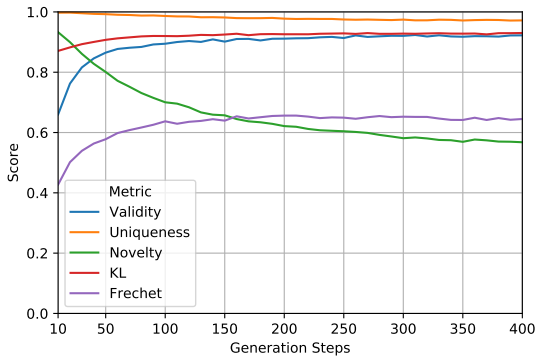
We also observe that for ChEMBL, which contains larger molecules, using a 1% masking rate yields points that are beyond the Pareto frontier of those obtained using a 5% masking rate. This further indicates that masking a large number of components hurts generation quality, even if this number represents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

## Effect of Generation Hyperparameters on Generation Quality

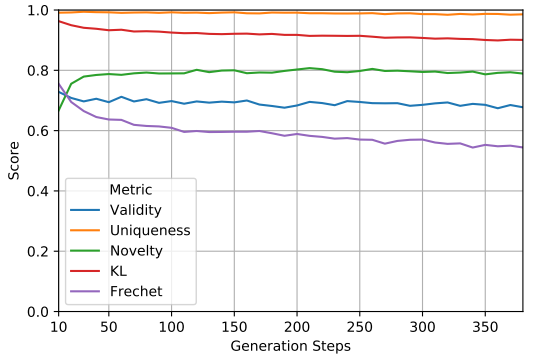
We analyze the effect of changing the masking rate and graph initialization on generation quality. In order to do so, we must choose results corresponding to a certain number of generation steps for each combination of masking rate and initialization. We therefore evaluate



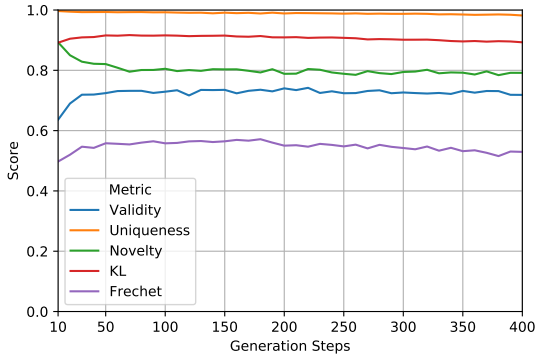

(a) Training initialization, 10% masking rate




(b) Marginal initialization, 10% masking rate




(c) Training initialization, 20% masking rate




(d) Marginal initialization, 20% masking rate

Figure 2: Plots of generation scores as a function of number of generation steps for each initialization and masking rate on QM9.

samples at intermediate steps of the generation process, as shown in Figure 2, to determine how the values of the evaluation metrics change as the number of generation steps increases.

For training initialization (Figures 2a and 2c), 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 (Figures 2b and 2d), 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

Table 3: Effect of varying masking rate and graph initialization on the benchmark results for our masked graph model on QM9 and ChEMBL.

| 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 |

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 Table 3. On QM9, we find that using marginal initialization leads to slightly higher validity and novelty scores however with lower KL-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 Table 3. 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 Table 3 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

Table 4: Distributional Results on QM9. CharacterVAE[35], GrammarVAE[40], GraphVAE[36] and MolGAN[41] results are taken from Cao and Kipf[41]. NAT GraphVAE[44] stands for non-autoregressive graph VAE. Our masked graph model results correspond to a 10% masking rate and training graph initialization, which has the highest geometric mean for all five benchmarks. Values of validity(↑), uniqueness(↑), novelty(↑), KL Div(↑) and Fréchet Dist(↑) metrics are between 0 and 1.

| | 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 | 0.945 | 0.343 | 0.806 | N/A | N/A |
| | MGM (ours proposed) | 0.886 | 0.978 | 0.518 | 0.966 | 0.842 |

Table 5: Distributional Results on ChEMBL. LSTM, Graph MCTS[43], AAE[45], ORGAN[42] and VAE[35] (with a bidirectional GRU[31] as encoder and autoregressive GRU[31] as decoder) results are taken from Brown et al.[21]. NAT GraphVAE[44] stands for non-autoregressive graph VAE. Our masked graph model results correspond to a 1% masking rate and training graph initialization, which has the highest geometric mean for all five benchmarks. Values of validity(↑), uniqueness(↑), novelty(↑), KL Div(↑) and Fréchet Dist(↑) metrics are between 0 and 1.

| | 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 |

Fréchet Distance scores. We can trade off between different metrics as desired by adjusting the initialization and masking rate.

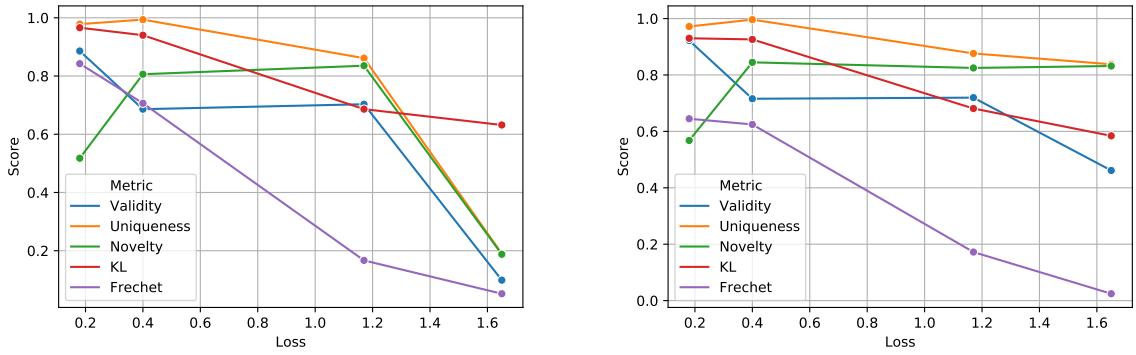
## Comparison with Baseline Models

We now compare our results on each dataset using our 'best' initialization strategy to baseline models. In previous sections, 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 Table 3 for each dataset corresponding to the highest geometric mean among all five metrics. The distributional benchmark results on QM9 and ChEMBL are shown in Table 4 and Table 5 respectively.

On QM9, our model performs comparably to existing methods. Our approach shows higher validity and uniqueness scores compared to CharacterVAE[35], GrammarVAE[40], GraphVAE[36] and MolGAN[41], while having a lower novelty score. Our model has a lower validity and novelty score than non-autoregressive graph VAE[44] while having a significantly higher uniqueness score. Compared to the autoregressive LSTM and Transformer models, our model has lower validity, KL-divergence and Fréchet Distance scores; however it exhibits slightly higher uniqueness and significantly higher novelty scores. Since KL-divergence and Fréchet scores are not available for the graph-based baselines as well as for CharacterVAE and GrammarVAE, we compare graph-based baselines to our model using these metrics on ChEMBL.

On ChEMBL, our approach outperforms existing graph-based methods. Compared to graph MCTS[43] and non-autoregressive graph VAE[44], 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




(a) Training Initialization

(b) Marginal Initialization

Figure 3: Benchmark metric results on QM9 corresponding to our model's checkpoints corresponding to different validation loss values. A masking rate of 10% was used.

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[31] 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.

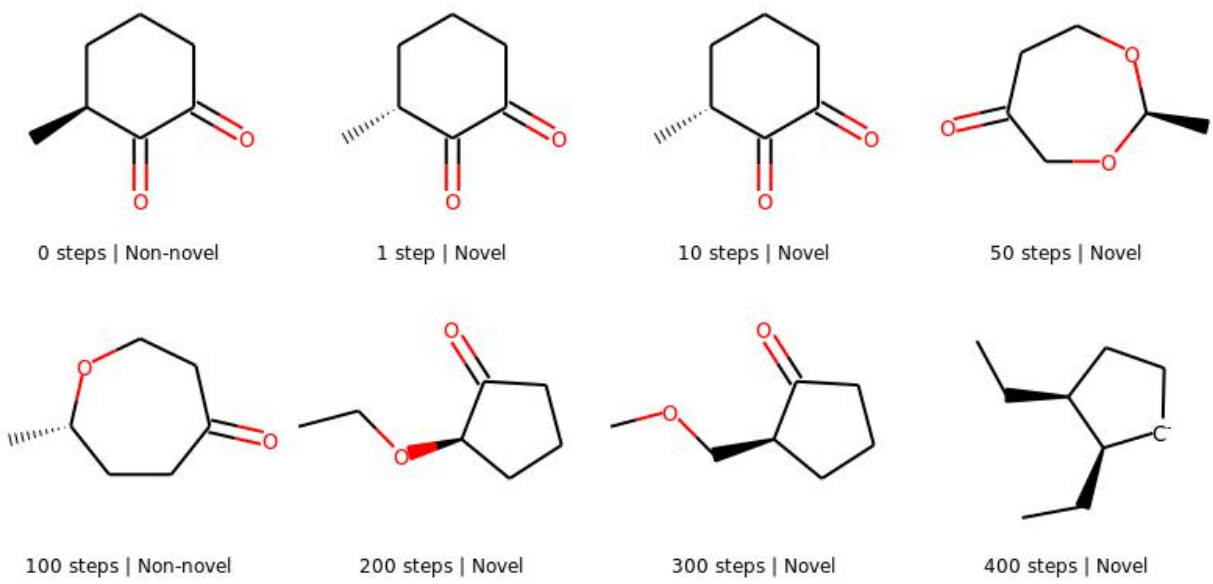
There are several differences between the QM9 and ChEMBL datasets that could account for this, including number of molecules, median molecule size and presence of chirality information. There has also been extensive work in developing language models compared to graph neural networks, which may account for the greater success of the LSTM and Transformers. We leave further investigation into the reasons behind the difference in performance to future work.

## Effect of Validation Loss on Generation Quality

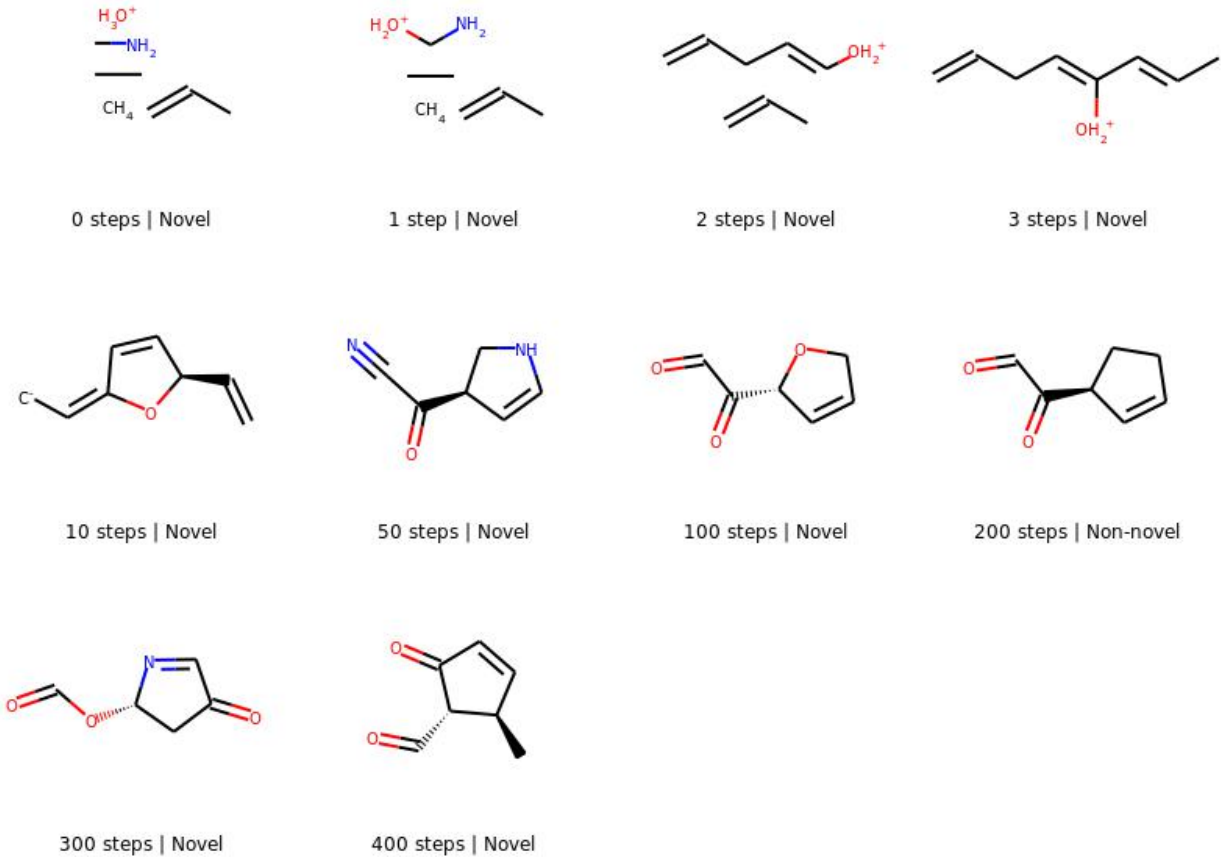
To determine whether validation loss is a suitable proxy for generation quality, we carry out generation from different training checkpoints of our 'best' QM9 model. During training, we carried out a hyperparameter search to find the configurations with the lowest validation loss, which we used as the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

Figure 3 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 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.

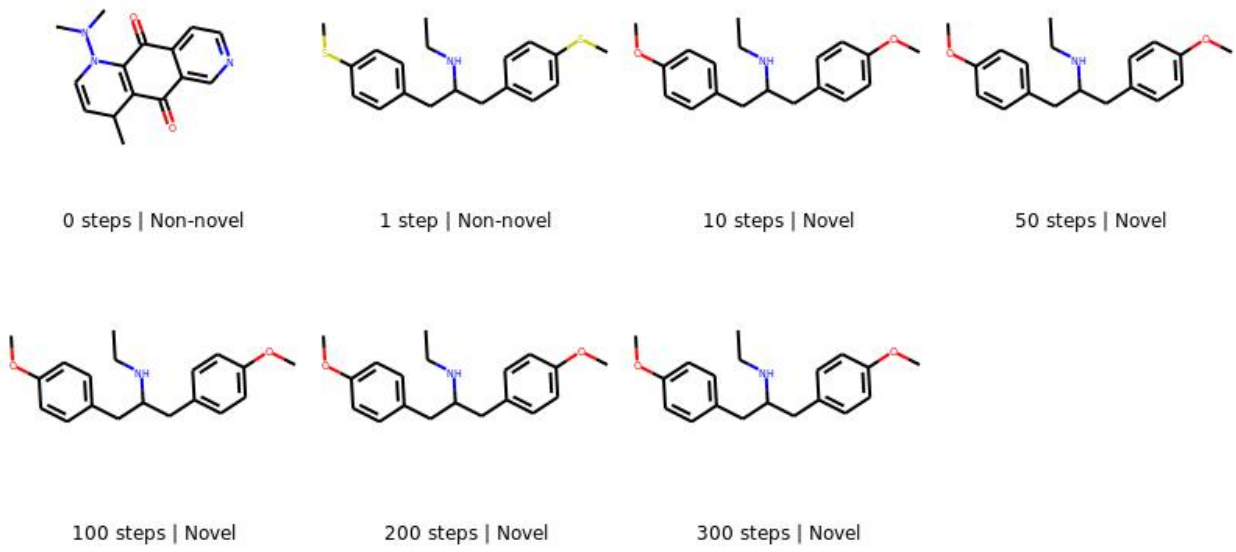



(a) Training initialization

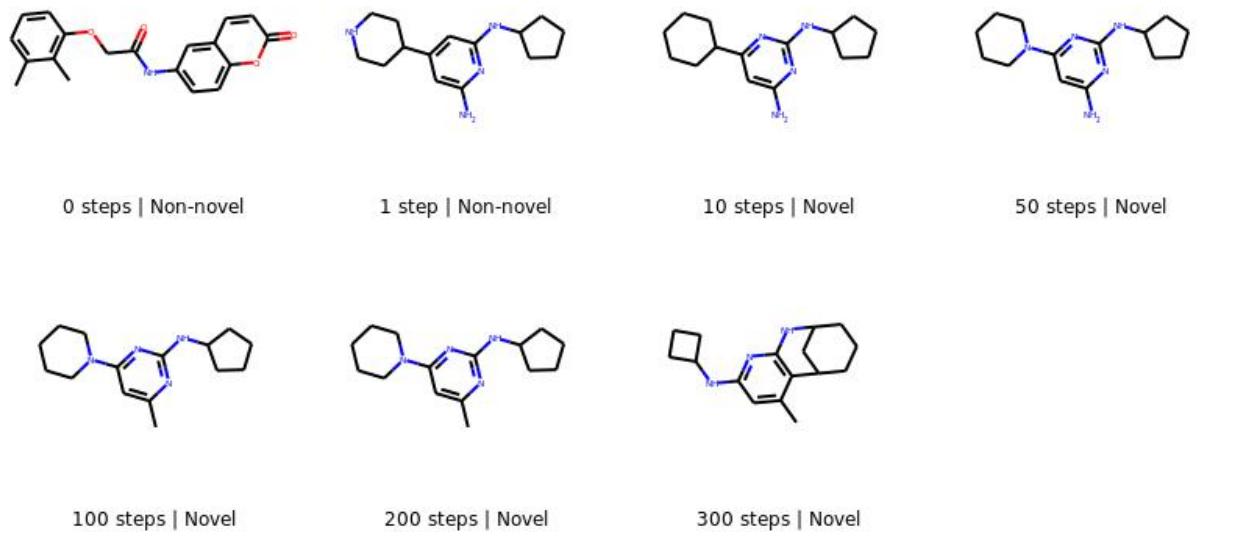



(b) Marginal initialization

Figure 4: Generation trajectory of a molecule each for training initialization and marginal initialization, for QM9 with a 10 % masking rate.




(a) 1% masking rate




(b) 5% masking rate

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

## Generation Trajectories

We present a few sampling trajectories of molecules from the proposed masked graph model in Figures 4–5. 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 4 shows a trajectory each for training and marginal initializations with a 10% masking rate. Figure 5 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 4a 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 4b 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 5a 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, following which it undergoes minor changes over the next few steps to yield a novel molecule. Figure 5b 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.

From these observations, we see that molecules converge towards the space of dataset molecules regardless of whether training or marginal initialization is used. This implies 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.

# 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.[35] 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.[40] explicitly added the grammar of SMILES strings to VAEs for molecule generation. Simonovsky and Komodakis[36] proposed a graph VAE to generate graph representations of molecules. Jin et al.[46] proposed using a VAE to generate a junction tree followed by the generation of the molecule itself. Kang and Cho[47] proposed a semi-supervised VAE trained on SMILES strings that performs joint molecular property prediction and molecule generation. Mahmood and Hernández-Lobato[48] proposed a constrained optimization method in the latent space of a VAE for goal-directed generation. Kwon et al.[44] proposed a non-autoregressive graph variational autoencoder trained with additional learning objectives for molecular graph generation. In addition to the previous work on extending VAEs for molecule generation, Wang et al.[49], Guimaraes et al.[42] and Cao and Kipf[41] 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 [43] showed that genetic algorithms based on Monte Carlo Tree Search (MCTS) could be competitive on the task of molecular generation. There has been some work applying reinforcement learning objectives to the task of molecular graph generation [50–52], which is orthogonal to our model.

**Generative Models of Graphs** Li et al. [53] proposed a deep generative model of graphs that predicts a sequence of transformation operations to generate a graph. You et al. [54] 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. [55] extended autoregressive models of graphs by adding blockwise parallel generation. Dai et al. [56] 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. [57] 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. [58] proposed a graph neural network model based on normalizing flows for memory-efficient prediction and generation.

**Masked Language Models** Masked language models, such as BERT [9], have been shown to bring significant improvements to a variety of discriminative language understanding tasks such as question answering [59,60] and natural language inference [61,62]. Wang and Cho [63], Ghazvininejad et al. [64] and Mansimov et al. [16] proposed ways to generate text directly from trained masked language models. Wang and Cho [63] proposed the use of Gibbs sampling, and Mansimov et al. [16] proposed the use of adaptive Gibbs sampling approaches for effective text generation using masked language models. Ghazvininejad et al. [64] 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.[65], Stern et al.[66] and Gu et al.[67] 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.[68] 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.[69] 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.[68] nor Maziarka et al.[69] investigated ways of generating novel molecular graphs with their trained models.

# Conclusion

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 iteratively sampling subsets of graph 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. We observe that by varying generation hyperparameters, we can trade off these metrics more efficiently than with state-of-the-art baseline models. We show that 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. We also establish the minimization of validation loss as a reasonable objective for improving generation quality. Finally, we examine molecule trajectories and observe convergence to molecules that are similar to those in the original datasets, indicating that our sampler converges to its target distribution.

Future avenues of work include adapting our model for goal-directed molecular generation and investigating the usefulness of representations learned by our model for downstream molecular property prediction tasks. As our approach is broadly applicable to generic graph structures, we also leave its application to non-molecular datasets to future work.

## Code Availability

Training and generation scripts for MGM and baseline models, as well as data and pretrained models can be found at `https://github.com/nyu-dl/dl4chem-mgm`.

## References


(1) Bohacek, R. S.; Mcmartin, C.; Guida, W. C. The art and practice of structure-based drug design: A molecular modeling perspective. *Medicinal Research Reviews* **1996**, *16*, 3–50.

(2) Hochreiter, S.; Schmidhuber, J. Long Short-Term Memory. *Neural Computation* **1997**, *9*, 1735–1780.

(3) Vaswani, A.; Shazeer, N.; Parmar, N.; Uszkoreit, J.; Jones, L.; Gomez, A. N.; Kaiser, L.; Polosukhin, I. Attention is All you Need. *ArXiv* **2017**, *abs/1706.03762*.

(4) Kingma, D. P.; Welling, M. Auto-Encoding Variational Bayes. *arXiv preprint 1312.6114* **2013**,

(5) Rezende, D. J.; Mohamed, S.; Wierstra, D. Stochastic Backpropagation and Approximate Inference in Deep Generative Models. ICML. 2014.

(6) Makhzani, A.; Shlens, J.; Jaitly, N.; Goodfellow, I. J. Adversarial Autoencoders. *ArXiv* **2015**, *abs/1511.05644*.

(7) Goodfellow, I. J.; Pouget-Abadie, J.; Mirza, M.; Xu, B.; Warde-Farley, D.; Ozair, S.; Courville, A. C.; Bengio, Y. Generative Adversarial Nets. NIPS. 2014.

(8) Elton, D.; Boukouvalas, Z.; Fuge, M. D.; Chung, P. W. Deep learning for molecular design—a review of the state of the art. 2019.

(9) Devlin, J.; Chang, M.-W.; Kenton Lee, K. T. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. NAACL. 2019.

(10) Wang, A.; Singh, A.; Michael, J.; Hill, F.; Levy, O.; Bowman, S. R. GLUE: A Multi-Task Benchmark and Analysis Platform for Natural Language Understanding. *ArXiv* **2018**, *abs/1804.07461*.

(11) Wang, A.; Pruksachatkun, Y.; Nangia, N.; Singh, A.; Michael, J.; Hill, F.; Levy, O.; Bowman, S. R. SuperGLUE: A Stickier Benchmark for General-Purpose Language Understanding Systems. *ArXiv* **2019**, *abs/1905.00537*.

(12) Nogueira, R.; Cho, K. Passage Re-ranking with BERT. *ArXiv* **2019**, *abs/1901.04085*.

(13) Liu, Y.; Ott, M.; Goyal, N.; Du, J.; Joshi, M.; Chen, D.; Levy, O.; Lewis, M.; Zettlemoyer, L.; Stoyanov, V. RoBERTa: A Robustly Optimized BERT Pretraining Approach. *ArXiv* **2019**, *abs/1907.11692*.

(14) Lan, Z.; Chen, M.; Goodman, S.; Gimpel, K.; Sharma, P.; Soricut, R. ALBERT: A Lite BERT for Self-supervised Learning of Language Representations. *ArXiv* **2020**, *abs/1909.11942*.

(15) Lample, G.; Conneau, A. Cross-lingual Language Model Pretraining. *ArXiv* **2019**, *abs/1901.07291*.

(16) Mansimov, E.; Wang, A.; Cho, K. A Generalized Framework of Sequence Generation with Application to Undirected Sequence Models. *arXiv preprint arXiv:1905.12790* **2019**,

(17) Krenn, M.; Häse, F.; Nigam, A.; Friederich, P.; Aspuru-Guzik, A. Self-Referencing Embedded Strings (SELFIES): A 100% robust molecular string representation. 2019.

(18) Ruddigkeit, L.; Van Deursen, R.; Blum, L. C.; Reymond, J.-L. Enumeration of 166 Billion Organic Small Molecules in the Chemical Universe Database GDB-17. *J. Chem. Inf. Model.* **2012**, *52*, 2864–2875.

(19) Ramakrishnan, R.; Dral, P. O.; Rupp, M.; Von Lilienfeld, O. A. Quantum Chemistry Structures and Properties of 134 Kilo Molecules. *Sci. Data* **2014**, *1*, 140022:1–7.

(20) Mendez, D. et al. ChEMBL: towards direct deposition of bioassay data. *Nucleic Acids Research* **2018**,

(21) Brown, N.; Fiscato, M.; Segler, M. H.; Vaucher, A. C. GuacaMol: Benchmarking Models for De Novo Molecular Design. *arXiv preprint 1811.09621* **2018**,

(22) Kullback, S.; Leibler, R. A. ON INFORMATION AND SUFFICIENCY. *Annals of Mathematical Statistics* **1951**, *22*, 79–86.

(23) Preuer, K.; Renz, P.; Unterthiner, T.; Hochreiter, S.; Klambauer, G. Fréchet ChemNet Distance: A Metric for Generative Models for Molecules in Drug Discovery. *Journal of chemical information and modeling* **2018**,

(24) Mikolov, T.; Kombrink, S.; Deoras, A.; Burget, L.; Cernocky, J. RNNLM - Recurrent Neural Network Language Modeling Toolkit. 2011.

(25) Bengio, Y.; Bengio, S. Modeling High-Dimensional Discrete Data with Multi-Layer Neural Networks. NIPS. 1999; pp 400–406.

(26) Larochelle, H.; Murray, I. The Neural Autoregressive Distribution Estimator. The Proceedings of the 14th International Conference on Artificial Intelligence and Statistics. 2011; pp 29–37.

(27) Vinyals, O.; Bengio, S.; Kudlur, M. Order Matters: Sequence to sequence for sets. *arXiv preprint arXiv:1511.06391* **2016**,

(28) Shu, R.; Lee, J.; Nakayama, H.; Cho, K. Latent-Variable Non-Autoregressive Neural Machine Translation with Deterministic Inference Using a Delta Posterior. 2019.

(29) Bengio, Y.; Ducharme, R.; Vincent, P.; Janvin, C. A Neural Probabilistic Language Model. *J. Mach. Learn. Res.* **2003**, *3*, 1137–1155.

(30) Gilmer, J.; Schoenholz, S. S.; Riley, P. F.; Vinyals, O.; Dahl, G. E. Neural Message Passing for Quantum Chemistry. Proceedings of the 34th International Conference on Machine Learning. 2017; pp 1263–1272.

(31) Cho, K.; Van Merriënboer, B.; Gulcehre, C.; Bahdanau, D.; Bougares, F.; Schwenk, H.; Bengio, Y. Learning Phrase Representations Using RNN Encoder-Decoder for Statistical Machine Translation. Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing. 2014; pp 1724–1734.

(32) Ba, J. L.; Kiros, J. R.; Hinton, G. E. Layer Normalization. 2016.

(33) Nair, V.; Hinton, G. E. Rectified Linear Units Improve Restricted Boltzmann Machines. ICML. 2010; pp 807–814.

(34) Glorot, X.; Bordes, A.; Bengio, Y. Deep Sparse Rectifier Neural Networks. Fort Lauderdale, FL, USA, 2011; pp 315–323.

(35) Gómez-Bombarelli, R.; Duvenaud, D.; Hernández-Lobato, J. M.; Aguilera-Iparraguirre, J.; Hirzel, T. D.; Adams, R. P.; Aspuru-Guzik, A. Automatic chemical design using a data-driven continuous representation of molecules. *arXiv preprint 1610.02415* **2016**,

(36) Simonovsky, M.; Komodakis, N. GraphVAE: Towards Generation of Small Graphs Using Variational Autoencoders. *arXiv preprint 1802.03480* **2018**,

(37) Li, Y.; Zhang, L.; Liu, Z. Multi-objective de novo drug design with conditional graph generative model. *Journal of Cheminformatics* **2018**, *10*.

(38) Goh, G. B.; Siegel, C.; Vishnu, A.; Hodas, N. O. ChemNet: A Transferable and Generalizable Deep Neural Network for Small-Molecule Property Prediction. *ArXiv* **2017**, *abs/1712.02734*.

(39) Kingma, D. P.; Ba, J. Adam: A Method for Stochastic Optimization. *CoRR* **2015**, *abs/1412.6980*.

(40) Kusner, M. J.; Paige, B.; Hernández-Lobato, J. M. Grammar Variational Autoencoder. ICML. 2017.

(41) Cao, N. D.; Kipf, T. MolGAN: An implicit generative model for small molecular graphs. *arXiv preprint 1805.11973* **2018**,

(42) Guimaraes, G. L.; Sanchez-Lengeling, B.; Farias, P. L. C.; Aspuru-Guzik, A. Objective-Reinforced Generative Adversarial Networks (ORGAN) for Sequence Generation Models. *CoRR* **2017**, *abs/1705.10843*.

(43) Jensen, J. H. Graph-based Genetic Algorithm and Generative Model/Monte Carlo Tree Search for the Exploration of Chemical Space. **2018**,

(44) Kwon, Y.; Yoo, J.; Choi, Y.; joon Son, W.; Lee, D.; Kang, S. Efficient learning of non-autoregressive graph variational autoencoders for molecular graph generation. *Journal of Cheminformatics* **2019**, *11*.

(45) Polykovskiy, D.; Zhebrak, A.; Vetrov, D.; Ivanenkov, Y.; Aladinskiy, V.; Mamoshina, P.; Bozdaganyan, M.; Aliper, A.; Zhavoronkov, A.; Kadurin, A. Entangled Conditional Adversarial Autoencoder for de Novo Drug Discovery. *Molecular Pharmaceutics* **2018**,

(46) Jin, W.; Barzilay, R.; Jaakkola, T. S. Junction Tree Variational Autoencoder for Molecular Graph Generation. ICML. 2018.

(47) Kang, S.; Cho, K. Conditional molecular design with deep generative models. *Journal of chemical information and modeling* **2019**, *59 1*, 43–52.

(48) Mahmood, O.; Hernández-Lobato, J. M. A COLD Approach to Generating Optimal Samples. *CoRR* **2019**, *abs/1905.09885*.

(49) Wang, H.; Wang, J.; Wang, J.; Zhao, M.; Zhang, W.; Zhang, F.; Xie, X.; Guo, M. GraphGAN: Graph Representation Learning with Generative Adversarial Nets. *CoRR* **2017**, *abs/1711.08267*.

(50) You, J.; Liu, B.; Ying, R.; Pande, V.; Leskovec, J. Graph Convolutional Policy Network for Goal-Directed Molecular Graph Generation. *ArXiv* **2018**, *abs/1806.02473*.

(51) Zhou, Z.; Kearnes, S. M.; Li, L.; Zare, R. N.; Riley, P. Optimization of Molecules via Deep Reinforcement Learning. *Scientific Reports* **2019**, *9*.

(52) Simm, G. N. C.; Pinsler, R.; Hernández-Lobato, J. M. Reinforcement Learning for Molecular Design Guided by Quantum Mechanics. 2020.

(53) Li, Y.; Vinyals, O.; Dyer, C.; Pascanu, R.; Battaglia, P. W. Learning Deep Generative Models of Graphs. *arXiv preprint arXiv:1803.03324* **2018**,

(54) You, J.; Ying, R.; Ren, X.; Hamilton, W. L.; Leskovec, J. GraphRNN: Generating Realistic Graphs with Deep Auto-regressive Models. ICML. 2018.

(55) Liao, R.; Li, Y.; Song, Y.; Wang, S.; Hamilton, W. L.; Duvenaud, D.; Urtasun, R.; Zemel, R. S. Efficient Graph Generation with Graph Recurrent Attention Networks. NeurIPS. 2019.

(56) Dai, H.; Nazi, A.; Li, Y.; Dai, B.; Schuurmans, D. Scalable Deep Generative Modeling for Sparse Graphs. *ArXiv* **2020**, *abs/2006.15502*.

(57) Grover, A.; Zweig, A.; Ermon, S. Graphite: Iterative Generative Modeling of Graphs. ICML. 2019.

(58) Liu, J.; Kumar, A.; Ba, J.; Kiros, J.; Swersky, K. Graph Normalizing Flows. NeurIPS. 2019.

(59) Rajpurkar, P.; Zhang, J.; Lopyrev, K.; Liang, P. SQuAD: 100, 000+ Questions for Machine Comprehension of Text. *ArXiv* **2016**, *abs/1606.05250*.

(60) Rajpurkar, P.; Jia, R.; Liang, P. Know What You Don't Know: Unanswerable Questions for SQuAD. *ArXiv* **2018**, *abs/1806.03822*.

(61) Bowman, S. R.; Angeli, G.; Potts, C.; Manning, C. D. A large annotated corpus for learning natural language inference. *ArXiv* **2015**, *abs/1508.05326*.

(62) Williams, A.; Nangia, N.; Bowman, S. R. A Broad-Coverage Challenge Corpus for Sentence Understanding through Inference. *ArXiv* **2018**, *abs/1704.05426*.

(63) Wang, A.; Cho, K. BERT has a Mouth, and It Must Speak: BERT as a Markov Random Field Language Model. *arXiv preprint arXiv:1902.04094* **2019**,

(64) Ghazvininejad, M.; Levy, O.; Liu, Y.; Zettlemoyer, L. Mask-Predict: Parallel Decoding of Conditional Masked Language Models. *arXiv preprint arXiv:1904.09324* **2019**,

(65) Welleck, S.; Brantley, K.; Daumé, H.; Cho, K. Non-Monotonic Sequential Text Generation. ICML. 2019.

(66) Stern, M.; Chan, W.; Kiros, J.; Uszkoreit, J. Insertion Transformer: Flexible Sequence Generation via Insertion Operations. ICML. 2019.

(67) Gu, J.; Liu, Q.; Cho, K. Insertion-based Decoding with Automatically Inferred Generation Order. *Transactions of the Association for Computational Linguistics*

(68) Hu, W.; Liu, B.; Gomes, J.; Zitnik, M.; Liang, P.; Pande, V. S.; Leskovec, J. Strategies for Pre-training Graph Neural Networks. *arXiv preprint arXiv:v* **2019**,

(69) Maziarka, L.; Danel, T.; Mucha, S.; Rataj, K.; Tabor, J.; Jastrzebski, S. Molecule Attention Transformer. *ArXiv* **2020**, *abs/2002.08264*.