De novo drug design using reinforcement learning with graph-based deep generative models

Sara Romeo Atance [1 2] **Juan Viguera Diez** [1 2] **Ola Engkvist** [1 2] **Simon Olsson** [2] **Rocío Mercado** [1]

Abstract

Machine learning methods have proven to be effective tools for molecular design, allowing for efficient exploration of the vast chemical space via deep molecular generative models. Here, we propose a graph-based deep generative model for *de novo* molecular design using reinforcement learning. We demonstrate how the reinforcement learning framework can successfully fine-tune the generative model towards molecules with various desired sets of properties, even when few molecules have the goal attributes initially. We explored the following tasks: decreasing/increasing the size of generated molecules, increasing their drug-likeness, and increasing protein-binding activity. Using our model, we are able to generate 95% predicted active compounds for a common benchmarking task, outperforming previously reported methods on this metric.

1. Introduction

Deep generative models (DGM) are being applied in an increasing amount of domains, and have successfully been used in a number of tasks including text (McKeown, 1992), music (Briot et al., 2017) and image (Gregor et al., 2015) synthesis. Applications of DGMs in the chemical sciences are also emerging with these models being used to generate promising molecules in fields such as drug discovery and materials design. The adoption of DGMs in chemistry has given rise to the sub-field of *generative chemistry* where the aim is to efficiently explore the vast chemical space and identify compounds with desired properties (Chen et al., 2018), such as new medicines (Stokes et al., 2020). For instance, RNNs (Segler et al., 2018; Li et al., 2018), VAEs (Gomez-Bombarelli et al., 2016; Ma et al., 2018; Jin et al., 2020), and GANs (Sanchez-Lengeling et al., 2017; De Cao & Kipf, 2018) have successfully been used in generative models for *de novo* molecular design.

Recent research has focused on addressing current limitations by using molecular graph representations in DGMs, where atoms and bonds in a molecule can naturally be represented as vertices and edges in a graph structure (Jiménez-Luna et al., 2021). Here, we describe a reinforcement learning (RL) strategy for fine-tuning graph-based DGMs for drug discovery applications. We test the proposed RL framework by fine-tuning a pre-trained DGM model to favour property profiles relevant to drug design tasks, including increasing pharmacological activity. We quantify activity using a quantitative structure activity relationship (QSAR) predictor of dopamine receptor D2 (DRD2) activity, a widely-used *de novo* design benchmark (Olivecrona et al., 2017; Blaschke et al., 2020; Arús-Pous et al., 2020). While RL has been applied to many string-based methods for *de novo* molecular design (Olivecrona et al., 2017; Popova et al., 2018; Guimaraes et al., 2017; Neil et al., 2018; Putin et al., 2018), our results encourage the possibility of future work in RL for graph-based molecular design using even more complex design objectives.

2. Related work

There is a variety of work applying RL to deep molecular generative models. While the majority of these models use string-based methods (Olivecrona et al., 2017; Popova et al., 2018; Guimaraes et al., 2017; Neil et al., 2018; Putin et al., 2018; Blaschke et al., 2020), RL has also been applied to select graph-based (You et al., 2018) and fingerprint-based (Zhou et al., 2019) models. We discuss the most relevant of these works in the subsections below.

2.1. Molecular DGMs

In addition to the aforementioned generative models, two closely-related molecular DGMs have inspired this work. The first is REINVENT (Blaschke et al., 2020), a string-based DGM, which uses RNNs to generate targeted molec-

[1]Molecular AI, Discovery Sciences, BioPharmaceuticals R&D, AstraZeneca, Gothenburg, Sweden [2]Chalmers University of Technology, Department of Computer Science and Engineering, Rännvägen 6, 41258 Göteborg, Sweden. Correspondence to: Sara Romeo Atance <atance@student.chalmers.se>, Rocío Mercado <rocio.mercado@astrazeneca.com>.

Reinforcement Learning for Real Life (RL4RealLife) Workshop in the 38th International Conference on Machine Learning, 2021. Copyright 2021 by the author(s).

ular strings via policy gradient RL. To take a graph-based approach, here we use the graph-based DGMs implemented in GraphINVENT (Mercado et al., 2021a), which use graph neural networks (GNNs) to generate molecular graphs, and combine them with an RL framework as in REINVENT. Graph-based models are not only less explored for deep molecular generation, but also allow direct learning from the graph structure, better handling of complex molecular ring systems, and simpler integration of 3D information (Jiménez-Luna et al., 2021).

2.2. Graph-based DGMs using RL

Previous work applying RL to molecular DGMs which explicitly treats molecules as graphs is limited, and consists of a graph convolutional network (GCN)-based model for targeted molecular graph generation using policy gradient methods (You et al., 2018).

As this work builds upon previous work, and we highlight here the key differences and improvements. In contrast to the graph convolutional policy network (GCPN) (You et al., 2018), the action space used by our underlying model, GraphINVENT, is split into 3 possible action types, while the GCPN uses 4 possible action types, which in both cases are concatenated to make the 'overall' action space. This difference is more of a design choice, however, as ultimately both models encode the action space similarly. Nonetheless, while GCPN uses the GCN implementation, our models use the gated graph neural network (GGNN) (Li et al., 2017), which was recently reported to outperform other GNN implementations in graph-based molecular generation applications (Mercado et al., 2021a). Finally, with the exception of quantitative estimate of drug-likeness (QED) optimisation, the other design tasks explored in this work are distinct from those explored previously; namely, the generation of potential active molecules was not explored with the GCPN.

3. Contributions

Using policy gradient RL, we extended a graph-based DGM for the generation of fine-tuned, drug-like molecules with desired properties. We propose the best agent reminder (BAR) loss and show that it significantly improves model training. We show consistency of our results using multiple different scoring functions to guide agents towards different design goals.

4. Methods

Our graph-based *de novo* design model consists of three main components:

1. a graph-based molecular DGM,
2. a RL framework with a memory-aware loss,
3. and the scoring model.

4.1. Graph-based molecular DGM

Following the GraphINVENT approach we use a Gated Graph Neural Network (GGNN)-based model (Mercado et al., 2021b). This model generates molecules by iteratively sampling 'actions' to build up an input graph. The problem of generating a molecular graph, $\mathcal{G}$, can be formulated as a Markov decision process, where an agent makes decisions sampling from the action probability distribution (APD), which encodes the action space (see Appendix B.1). Examples of graph construction actions are 'add node/edge' or 'terminate'. The APD is predicted by the generative model, conditioned on the current graph state.

To summarise, we build molecules using a sequence of n actions $\mathcal{A} = \{a_0, a_1, \ldots, a_{n-1}\}$, where $a_i \sim \mathrm{APD}_i$ and $f : \mathcal{G}_i \mapsto \mathrm{APD}_i$. Here, f represents our GGNN-based model, and APD_i is shorthand for $\mathrm{APD}(: |\mathcal{G}_i)$, where ':' stands for all possible actions to take. Starting from an empty graph $\mathcal{G}_0$, and ending with the final graph $\mathcal{G}_n$, the graph generation process proceeds as follows: $\mathcal{G}_0 \rightarrow a_0 \sim \mathrm{APD}_0 \rightarrow \mathcal{G}_1 \rightarrow \cdots \rightarrow a_{n-1} \sim \mathrm{APD}_{n-1} \rightarrow \mathcal{G}_n$.

The model is trained by minimising the Kullback-Leibler (KL) divergence between 'true' and predicted APDs. The set of chosen hyperparameters is the result of an exhaustive search and is detailed in Appendix A.1. The best model was selected at the epoch which minimised the validation loss and used as the 'prior' in the RL framework.

4.2. Memory-aware RL framework

We build on the previously reported REINVENT algorithm for fine-tuning (Olivecrona et al., 2017). The goal in REINVENT is to update the agent policy π from the prior policy π_{Prior} so as to increase the expected score for the action sequences used to build a graph. Here, the policy is parameterised using our graph-based model that predicts an APD given an input graph.

The loss we propose here uses a reward shaping mechanism (Buhmann et al., 2011). Briefly, compared to that of REINVENT, we introduce a loss term which keeps track of the best agent so far and is updated every few learning steps. By doing so, we remind the current agent of sets of actions that can lead to high-scoring compounds, in turn accelerating agent learning. The best agent reminder (BAR) loss takes the form

$$J(\boldsymbol{\theta}) = \frac{(1-\alpha)}{N} \sum_{m \in \mathcal{M}} J_{\mathrm{mol}}(\mathbb{A}, \mathbb{P}, \mathcal{A}_m; \boldsymbol{\theta}) + \frac{\alpha}{N} \sum_{\tilde{m} \in \tilde{\mathcal{M}}} J_{\mathrm{mol}}(\mathbb{A}, \tilde{\mathbb{A}}, \tilde{\mathcal{A}}_{\tilde{m}}; \boldsymbol{\theta}). \tag{1}$$

Above, α is a scaling factor that we treat as a hyperparameter.

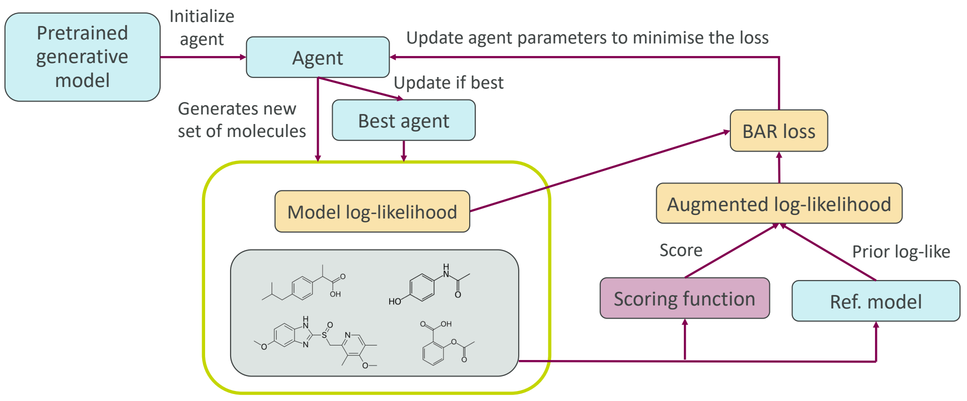

Figure 1. RL loop. *Augmented log-likelihood* refers to the second term, with the reference likelihood and the score, in Eq. 2.

$\mathbb{P}$ is the Prior model. $\mathcal{A}$ and $\tilde{\mathcal{A}}$ refer to the sets of actions taken to build a molecule by the current, $\mathbb{A}$, and best, $\tilde{\mathbb{A}}$, agents, respectively. $\mathcal{M}$ is the set of molecules m generated by the current agent. $\tilde{\mathcal{M}}$ is the set of molecules $\tilde{m}$ generated by the best agent. N is the number of molecules sampled by each model. Then, for each molecule:

$$J_{\text{mol}}(\mathbb{B}, \text{Ref.}, \mathcal{B}; \boldsymbol{\theta}) = \left[\log P(\mathcal{B})_{\mathbb{B}} - (\log P(\mathcal{B})_{\text{Ref.}} + \sigma S(\mathcal{B}))\right]^2 . \quad (2)$$

Above, σ is a scaling factor that we treat as a hyperparameter. Defining $\text{APD}_{\mathbb{B}}(b_i|\mathcal{G}_i)$ as the probability of sampling action b_i given the input graph $\mathcal{G}_i$, then $P(\mathcal{B})_{\mathbb{B}} = \prod_{i=0}^{n-1} \text{APD}_{\mathbb{B}}(b_i|\mathcal{G}_i)$ is the probability of taking the sequence of actions $\mathcal{B}$ given model $\mathbb{B}$, and $P(\mathcal{B})_{\text{Ref.}}$ is the analogous probability given by the reference model for the same sequence of actions. $S(\mathcal{B})$ is the score for the molecule generated following actions $\mathcal{B}$. The score modulates the log-probabilities given by the reference model and ensures that those of poorly scoring molecules are lowered relative to those of highly scoring molecules.

The learning process (Fig. 1) consists of the following steps:

1. Initialize the current and best agents to the prior model. For the prior, we use the pre-trained DGM.
2. Generate a batch of molecules with both the current and the best agents, keeping track of the actions.
3. Score all generated molecules.
4. Compute the probabilities that
   i. the prior model $\mathbb{P}$ and current agent $\mathbb{A}$ assign to $\mathcal{A}$, the set of actions taken by the current agent
   ii. the current agent $\mathbb{A}$ and best agent $\tilde{\mathbb{A}}$ assign to $\tilde{\mathcal{A}}$, the set of actions taken by the best agent.
5. Compute the BAR loss (Eq. 1).[1]
6. Update the current agent parameters so as to minimise the loss.
7. Continue the RL loop by going back to step 2 and updating the best agent every 5 learning steps.[2]

4.3. Scoring model

The scoring model should be designed for each specific optimisation task, and can range in complexity. Here, we implemented four different scoring functions. The goals of the scoring functions were to:

1. Change the average size (↑ or ↓) of molecules.
2. Promote ‘drug-likeness’ in molecules.
3. Promote DRD2 activity.

The first two scoring functions were used to test the operation of the RL framework. The final scoring function was designed to be more representative of the properties one seeks to optimise in a drug discovery project.

During scoring, molecules which are invalid, improperly terminated and/or duplicates are assigned a score of 0. We do not penalise undesired molecules; in this way, the model may learn to explore undesirable molecules that may lead to more desirable ones during the learning process.

[1]When computing the loss, we disregard duplicates in a batch of sampled molecules so as to not update twice in the same direction and encourage generation of repeated molecules. Fewer unique molecules are generated when we include duplicates in computing the loss.

[2]The best agent is updated if the average score of 1000 generated molecules is the largest observed (1000 molecules chosen as a trade-off between speed and sufficient sampling).

4.3.1. Reducing and increasing the average size of the molecules.

On average, molecules sampled from the prior contain 26 heavy atoms. As such, we began exploring the RL framework with the simple task of shifting the distribution of the number of nodes in the sampled molecules towards smaller and larger molecules.

We accomplished these two tasks by defining a scoring function that creates a maximum reward for molecules with 10 and 40 heavy atoms, respectively. More specifically:

$$S_{\text{size}}(\mathcal{A}) = \begin{cases} 0 & \text{if not } \{\text{PT, valid } \textit{and} \text{ unique}\}, \\ 1 - \frac{|n_{nodes} - n^{\star}_{nodes}|}{\max_{nodes} - n^{\star}_{nodes}} & \text{otherwise,} \end{cases} \tag{3}$$

where $n^{\star}_{nodes}$ is the target number of heavy atoms in sampled molecules and was set to 10 or 40 for the tasks of reducing and increasing molecular size, respectively. Here, $\mathcal{A}$ is the set of actions taken to build the molecule, PT stands for *properly terminated*, n_{nodes} is the number of heavy atoms in the molecule, and $\max_{nodes}$ is the maximum number of nodes allowed in the model (72 here).

4.3.2. Promoting drug-like molecules.

The next scoring function is based on the QED (Bickerton et al., 2012) implementation from RDKit (Landrum):

$$S_{\text{QED}}(\mathcal{A}) = \begin{cases} 0 & \text{if not } \{\text{PT, valid } \textit{and} \text{ unique}\}, \\ \text{QED}(\text{Mol}(\mathcal{A})) & \text{otherwise.} \end{cases} \tag{4}$$

Here, $\text{Mol}(\mathcal{A})$ refers to the molecule generated via actions $\mathcal{A}$. QED values can range between 0 and 1, with higher values indicating a molecule is more drug-like. The goal of this scoring function is to guide the DGM towards the generation of more drug-like molecules, although it should be noted that QED does not necessarily correlate with pharmacological activity.

4.3.3. Promoting DRD2 active molecules.

Finally, we investigated a scoring model to fine-tune our DGM towards the generation of drug-like, DRD2-active molecules. Here, we made use of a QSAR model (Kotsias et al., 2020a) to predict DRD2 activity in sampled compounds, as well as the QED discussed previously:

$$S_{\text{activity}}(\mathcal{A}) = \begin{cases} 1 \text{ if PT, valid, unique, QED} > 0.5 \\ \quad \textit{and} \text{ activity} > 0.5, \\ 0 \text{ otherwise.} \end{cases} \tag{5}$$

Like QED, predicted activity ranges from 0 to 1, with 1 indicating that a molecule is likely active. However, as the QED and QSAR models are not perfect (Bickerton et al., 2012), we used a threshold of 0.5 to classify molecules as either 'active' (QED *and* activity > 0.5) or 'inactive' (QED *or* activity < 0.5). We observed that, for instance, a molecule with a predicted activity score of 0.4 is not likely to be a 'true' active, and thus found a threshold of 0.5 to work well in preventing the model from learning from bad examples.

We compare the molecules generated using this scoring function with a dataset of predicted DRD2 active molecules, which consists of 3627 molecules which score 1 according to Eq. 5. Comparison to this set allows us to evaluate if the model can learn to generate known true DRD2 actives having seen no previous examples, as known actives were removed from the original training set.

4.4. Dataset details

The dataset used to train the prior was downloaded from (Kotsias et al., 2020b) and is a subset of ChEMBL (Mendez et al., 2018) with known DRD2 active molecules removed. Molecules in the remaining set are made up of {H, C, N, O, F, S, Cl, Br} and $<$ 50 heavy atoms (Kotsias et al., 2020c). $5 \cdot 10^5$ molecules were randomly selected from it to create the training set, with $5 \cdot 10^4$ for validation and $5 \cdot 10^4$ for testing. The DRD2 'predicted actives' dataset was downloaded from (Kotsias et al., 2020b).

5. Results

5.1. Using the BAR loss function

When analysing the behaviour of the reinforcement learning framework using different values of α in the loss function (Eq. 1) with the activity scoring function (Eq. 5), we observe that a value of $\alpha = 0.5$ helps to significantly improve learning (Fig. 2). As the score is discrete, the model learns only when molecules satisfying all the desired criteria are sampled, and the model does not generate many active molecules initially (see $\alpha = 0.0$ in Fig. 2). Therefore, it is especially helpful in this setting to have introduced a memory-mechanism to the loss via the term which depends on the best recent agent and is modulated by α. Without this term, the agent may forget combinations of actions which result in high activities/scores. We found that using $\alpha = 0.5$ not only accelerated and stabilised learning, but also led to a greater fraction of predicted actives sampled.

5.2. Tuning desired properties via the scoring function

We show here some results for the scoring functions defined above. To prove the ability of the RL framework to fine-tune the DGM towards the generation of molecules with desired properties, we used the scoring functions previously defined in Eqs. 3, 4, and 5. For hyperparameters, see Appendix A.2.

In Fig. 3 we show the evolution of the average score, the

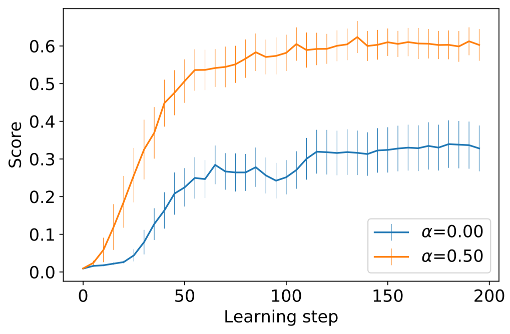

Figure 2. Comparison of the average score of the generated molecules as a function of learning step. The results in blue are analogous to using the loss proposed in REINVENT (Blaschke et al., 2020), which is recovered when $\alpha = 0.0$. The results in orange correspond to keeping contributions from the best recent agent in the loss with $\alpha = 0.5$ (Eq. 1).

fraction of valid and properly terminated molecules (those which do not violate any chemical rule and for which the last sampled action was 'terminate'), and the fraction of unique molecules (non repeated among the generated compounds) during learning. Several observations can be made:

- Our model improves the average score of sampled molecules using all four scoring functions. We highlight that the model was able to learn how to generate well-scoring molecules even when we searched for active DRD2 molecules, of which no known true positive examples were given during training.
- The percentage of valid and properly terminated molecules improves during learning as we penalise invalid and improperly terminated molecules.
- The fraction of unique molecules decreases during learning when reducing molecular size or promoting drug-like and active compounds. This behaviour is undesirable but unsurprising, as we are updating towards a smaller chemical space.
- The results are robust as most metrics exhibit very little noise.

We illustrate examples of molecules from the training set, samples from the pre-trained model, and samples from the fine-tuned models in Fig. 4. We find that all generated molecules look reasonable although some of them may be less stable due to the large macrocycles present in them. In particular, less stable molecules are sampled more often from the models which aim to increase the size and drug-likeness of molecules. Nonetheless, our model is successful at generating molecules using all four scoring functions. As can be seen, there is a remarkable change in the size of the molecules sampled when reducing and increasing the number of atoms in the molecules, especially compared to molecules sampled from the original GraphINVENT model. Additionally, the model is successful when fine-tuning molecules towards higher QED scores (Eq. 4), as they indeed look 'drug-like'. Finally, the results for the activity scoring function are remarkable, as 95% of sampled molecules are predicted to be active by the QSAR model.

We analyse further the results achieved by the most complex scoring function (the DRD2 activity score) in Table 1. For this experiment, we compared the fine-tuned models to the prior as follows:

1. First, we sampled 10K molecules from the prior model.
2. Then, we sampled 10K molecules from a single fine-tuned model.
3. Finally, we sampled and collected 1K molecules from 10 different fine-tuned models (same set of hyperparameters, but different training runs).

For each set of sampled molecules, we computed their average QED, average DRD2 activity, and how many are predicted actives. We also computed the number of known true actives generated by each model. We observe that both sets of fine-tuned molecules show similar values for the first metrics, and are substantially improved compared to the pre-trained model. Most importantly, while the prior model is not able to generate any known true DRD2 actives, both fine-tuned models are indeed able to sample known actives. Notably, when the 10K molecules come from 10 different fine-tuned models, the number of known actives sampled is 10-fold higher than when 10K molecules are sampled from a single model. This follows from the previous reasoning about the RL-trained models being heavily-dependent on the initial learning steps; as such, there is little overlap in sets of molecules generated during different RL runs.

6. Discussion

The goal of our model is to explore the chemical space in search of promising new molecules that demonstrate pharmacological activity. Use of the memory mechanism allows our model to train more smoothly, at the cost of introducing some bias to it. However, by keeping track of the best agent rather than the best molecules generated (another popular memory mechanism in DGMs; see Popova et al. 2018; Putin et al. 2018; Blaschke et al. 2020), we believe that the model is less biased, thus able to balance exploration and the generation of novel structures without forgetting actions that led to good molecules.

Of all the tasks explored, our model shows particular promise for the task of generating DRD2 actives, a pop-

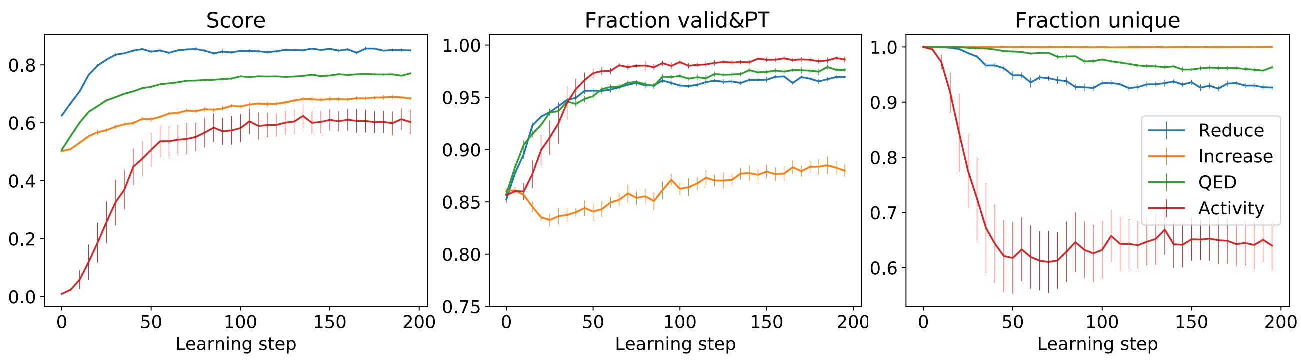

Figure 3. Learning curves for the four different scoring functions investigated. Left: Evolution of the average score of the generated molecules during learning. Centre: Evolution of the fraction of generated molecules which are both valid *and* properly terminated during learning. Right: Evolution of the fraction of unique molecules generated during learning. The values are computed in all cases for 1000 molecules, taking averages over 10 runs. The error bars correspond to the standard deviation. The hyperparameter values used are $\alpha = 0.5$ for all four scoring functions, and $\sigma = 10$ for $\{$Reduce, Increase, and QED$\}$ and $\sigma = 20$ for Activity.

Table 1. Comparison of various evaluation metrics for three sets of 10K generated molecules: one in which all are sampled from the pre-trained DGM without fine-tuning (*Prior*), another in which all are sampled from a single fine-tuned model (*Single*), and another in which 1K molecules are sampled from 10 separate fine-tuned models and combined (*Comb.*). *Active* refers to the percentage of molecules which have predicted QED and activity scores > 0.5. *Known true actives* refers to the percentage of molecules from the DRD2 dataset which have been re-generated by each model.

	Prior	**Single**	**Comb.**
Average QED	0.59	0.72	0.76
Average DRD2 activity	0.03	0.92	0.94
Active (%)	1	94	97
Unique (%)	100	48	60
Active and unique (%)	1	45	58
Known true actives (%)	0.00	0.08	0.83

ular benchmark for molecular DGMs as it simulates a ‘real’ drug discovery task. Compared to previous work (Kotsias et al., 2020c), our model is able to generate a much greater fraction of predicted active molecules after removal of duplicates: 95% active compounds, compared to only 54% in the best model from the aforementioned work. Although the percentage of known true actives that our models are able to recover is very small, we highlight that the DGM has not seen any examples of known true active molecules at any point during training and that there were no predicted actives generated before fine-tuning. This finding suggests that the model could be used to generate actives in a challenging but realistic drug discovery setting where little to no actives are known.

We believe the good performance of our model is due to the term in the BAR loss function which keeps track of the best agent so far. By keeping track of the best agent during training, we were able to stabilise learning and achieve better performance for all models. We speculate the origin of the improved performance of the BAR loss is similar to that seen in momentum-based optimisers in stochastic gradient descent. We leave a rigorous theoretical analysis of the loss for future work. The trained models are robust, and show little variation between runs in terms of the metrics of interest (Fig. 3), and only the fraction of unique samples varies notably between runs when aiming to generate DRD2 actives. This task is extremely difficult, as it depends strongly on the first active molecules generated by the model, which means the sets of actives generated by a model during different runs generally have negligible overlap.

We can compare our model to previous work, the GCPN (You et al., 2018), for the task of QED optimisation. Here, the authors report the top 3 QED values obtained from molecules generated by their fine-tuned model: 0.948, 0.947, and 0.946. Similarly, we find the top 3 QED values out of 1000 molecules sampled by our model after QED fine-tuning to be 0.948, 0.947, and 0.947. Furthermore, for 10 different runs of 1000 samples each, all top 3 QEDs are in the range of 0.940-0.948. The models thus show similar performance for this task, and suggest that 0.948 may be an upper limit for the task of QED optimisation.

The main drawback of the proposed model is the amount of time and computational power needed to pre-train the underlying GraphINVENT model (a few days on an NVIDIA Tesla K80); however, this is on par with that needed for other state-of-the-art molecular DGMs (Zhang et al., 2021), and only has to be done once per dataset. After pre-training, fine-tuning the model with RL is comparatively quick and requires only between $10 - 40$ minutes, where scoring the model is the main bottleneck. Furthermore, the same pre-trained model can be fine-tuned for multiple tasks, making our model competitive with other tools.

Some molecules generated by the models when increasing molecular size and QED appear to have a larger fraction of (undesirable) macrocycles and unstable moieties. Additionally, the percent valid and properly terminated does not increase as much when fine-tuning the model towards larger molecules as for the other scores (Fig. 3). We believe in these cases, the model has not seen many examples on how to predict reasonable APDs, making it difficult for it to learn actions that lead to large, *stable* molecules. We do not observe this trend when reducing the size of the molecules, and we believe it is because the model sees significantly more small sub-graphs during pre-training. QED is an equally challenging property to optimise as it is highly non-linear. These challenges motivated the use of the 0.5 threshold in Eq. 5, which proved to work well. However, exploring better estimates of molecular stability, drug-likeness, and synthetic accessibility in the scoring function is a possible way to minimise the sampling of undesirable structures, and is a topic of future work.

7. Conclusions

Here, we have used RL to develop a graph-based *de novo* molecular design tool. The proposed RL framework has shown a remarkable ability for fine-tuning the pre-trained DGM towards production of molecules with desired sets of properties, even in challenging situations where only a few examples of compounds with the desired properties were initially sampled. We have shown our model is able to perform well in several tasks most notably promoting the generation of DRD2 active molecules. While favouring certain properties, our RL framework also improves other performance metrics including increasing the percentage of valid and properly terminated molecules, reaching validity rates comparable to that of state-of-the-art models (Brown et al., 2019; Polykovskiy et al., 2020; Zhang et al., 2021).

Many properties a molecule exhibits directly depend on its molecular graph. As such, we believe the development of graph-based methods is key for the next generation of *de novo* design tools, as graphs can naturally encode structural information. Our tool is thus an important stepping stone towards the design of more advanced molecular DGMs and tools which will allow scientists to efficiently traverse the chemical space in search of promising molecules. We believe the use of DGMs in fields like drug design has the potential to help chemists come up with new ideas, and to accelerate the complex process of molecular discovery.

Software and Data

Code for this work is available at `https://github.com/olsson-group/RL-GraphINVENT`.

Acknowledgements

We thank Dr. Atanas Patronov and Vendy Fialkova for useful discussions on previous work done on REINVENT. We also thank Prof. Morteza Haghir Chehreghani for his useful feedback on the manuscript. This work was partially supported by the Wallenberg AI, Autonomous Systems and Software Program (WASP) funded by the Knut and Alice Wallenberg Foundation (to S.O.).

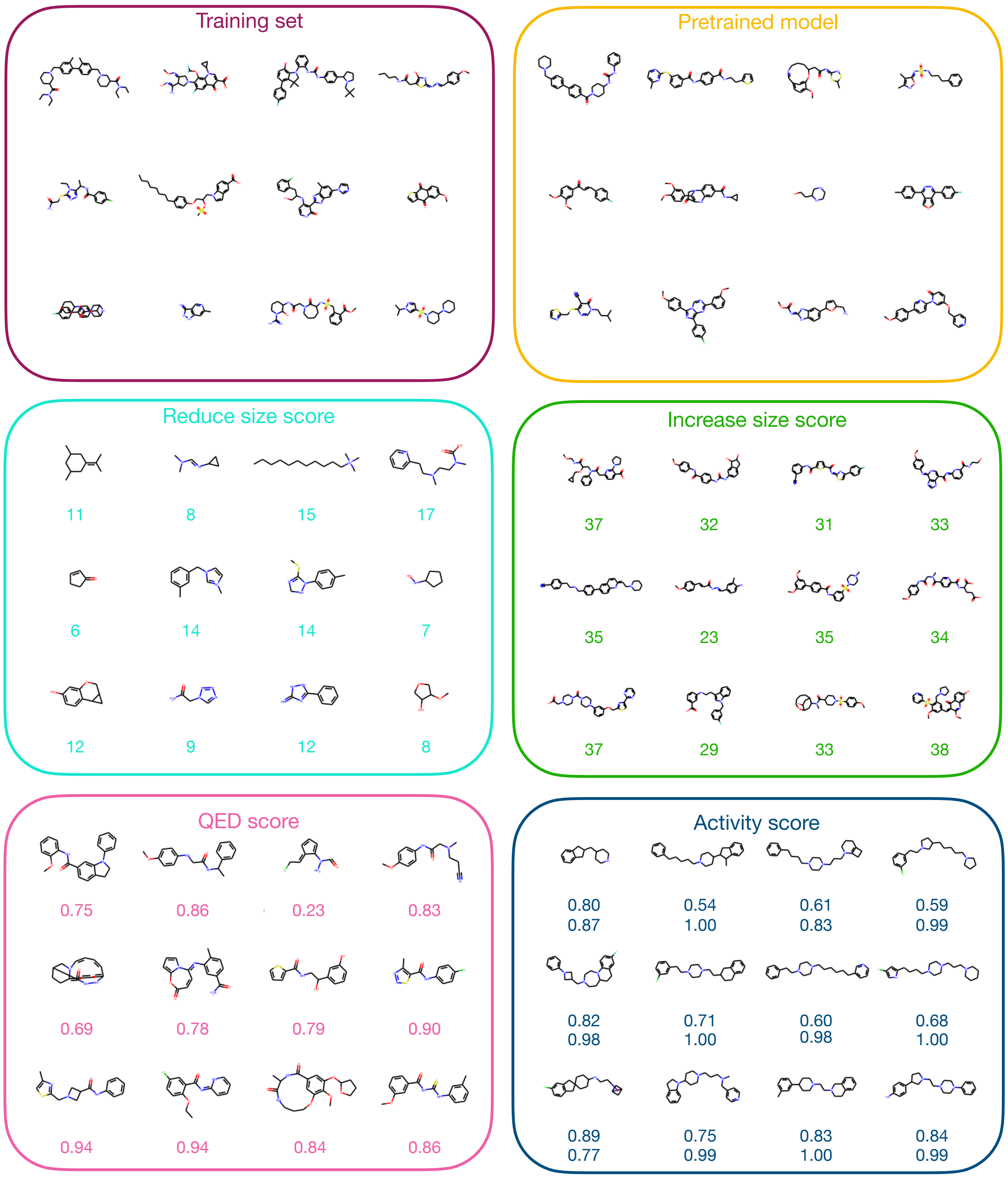

Figure 4. Top left: Examples of molecules in the training set. Top right: Examples of molecules generated by the pre-trained model. Centre left: Examples of molecules generated by the model after fine-tuning with the score defined in Eq. 3 for reducing the size of the generated molecules; the value below each molecule corresponds to its number of nodes. Centre right: Examples of molecules generated by the model after fine-tuning with the score defined in Eq. 3 for increasing the size of the generated molecules; the value below each molecule corresponds to its number of nodes. Bottom left: Examples of molecules generated by the model after fine-tuning with the score defined in Eq. 4 for promoting drug-like molecules (high QED); the value below each molecule corresponds to its QED. Bottom right: Examples of molecules generated by the model after fine-tuning the pre-trained model using the score defined in Eq. 5 for promoting the generation of drug-like, DRD2 active molecules; the numbers below each molecule corresponds to its QED (top) and activity estimate (bottom). All molecules shown are predicted to be active (QED > 0.5 and activity > 0.5).

References

Arús-Pous, J., Patronov, A., Bjerrum, E. J., Tyrchan, C., Reymond, J.-L., Chen, H., and Engkvist, O. Smiles-based deep generative scaffold decorator for de-novo drug design. *Journal of Cheminformatics*, 12:1–18, 2020.

Bickerton, G. R., Paolini, G. V., Besnard, J., Muresan, S., and Hopkins, A. L. Quantifying the chemical beauty of drugs. *Nature Chemistry*, 4(2):90–98, 2012.

Blaschke, T., Arús-Pous, J., Chen, H., Margreitter, C., Tyrchan, C., Engkvist, O., Papadopoulos, K., and Patronov, A. REINVENT 2.0: An AI tool for de novo drug design. *Journal of Chemical Information and Modeling*, 2020.

Briot, J.-P., Hadjeres, G., and Pachet, F.-D. Deep learning techniques for music generation–a survey. *arXiv preprint arXiv:1709.01620*, 2017.

Brown, N., Fiscato, M., Segler, M. H., and Vaucher, A. C. GuacaMol: Benchmarking models for de novo molecular design. *Journal of Chemical Information and Modeling*, 59(3):1096–1108, 2019.

Buhmann, M. D., Melville, P., Sindhwani, V., Quadrianto, N., Buntine, W. L., Torgo, L., Zhang, X., Stone, P., Struyf, J., Blockeel, H., Driessens, K., Miikkulainen, R., Wiewiora, E., Peters, J., Tedrake, R., Roy, N., Morimoto, J., Flach, P. A., and Fürnkranz, J. Reward shaping. In *Encyclopedia of Machine Learning*, pp. 863–865. Springer US, 2011. doi: 10.1007/978-0-387-30164-8_731. URL https://doi.org/10.1007/978-0-387-30164-8_731.

Chen, H., Engkvist, O., Wang, Y., Olivecrona, M., and Blaschke, T. The rise of deep learning in drug discovery. *Drug Discovery Today*, 23(6):1241–1250, 2018.

De Cao, N. and Kipf, T. MolGAN: An implicit generative model for small molecular graphs. *arXiv preprint arXiv:1805.11973*, 2018.

Glorot, X. and Bengio, Y. Understanding the difficulty of training deep feedforward neural networks. In *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics*, volume 9 of *Proceedings of Machine Learning Research*, pp. 249–256, Chia Laguna Resort, Sardinia, Italy, 13–15 May 2010. PMLR.

Gomez-Bombarelli, R., Duvenaud, D., and Miguel, J. Automatic chemical design using a data-driven continuous representation of molecules. arxiv. *arXiv preprint arXiv:1610.02415*, 2016.

Gregor, K., Danihelka, I., Graves, A., Rezende, D., and Wierstra, D. DRAW: A recurrent neural network for image generation. In *Proceedings of the 32nd International Conference on Machine Learning*, volume 37 of *Proceedings of Machine Learning Research*, pp. 1462–1471, Lille, France, 07–09 Jul 2015. PMLR.

Guimaraes, G. L., Sanchez-Lengeling, B., Outeiral, C., Farias, P. L. C., and Aspuru-Guzik, A. Objective-reinforced generative adversarial networks (organ) for sequence generation models. *arXiv preprint arXiv:1705.10843*, 2017.

Jiménez-Luna, J., Grisoni, F., Weskamp, N., and Schneider, G. Artificial intelligence in drug discovery: Recent advances and future perspectives. *Expert Opinion on Drug Discovery*, pp. 1–11, 2021.

Jin, W., Barzilay, R., and Jaakkola, T. Hierarchical generation of molecular graphs using structural motifs. *arXiv preprint arXiv:2002.03230*, 2020.

Kingma, D. P. and Ba, J. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

Klambauer, G., Unterthiner, T., Mayr, A., and Hochreiter, S. Self-normalizing neural networks. *arXiv preprint arXiv:1706.02515*, 2017.

Kotsias, P.-C., Arús-Pous, J., Chen, H., Engkvist, O., Tyrchan, C., and Bjerrum, E. J. DeepDrugCoder (DDC): Heteroencoder for molecular encoding and de novo generation, 2020a. URL https://github.com/pcko1/Deep-Drug-Coder/blob/master/models/qsar_model.pickle.

Kotsias, P.-C., Arús-Pous, J., Chen, H., Engkvist, O., Tyrchan, C., and Bjerrum, E. J. DeepDrugCoder (DDC): Heteroencoder for molecular encoding and de novo generation, 2020b. URL https://github.com/pcko1/Deep-Drug-Coder/tree/master/datasets.

Kotsias, P.-C., Arús-Pous, J., Chen, H., Engkvist, O., Tyrchan, C., and Bjerrum, E. J. Direct steering of de novo molecular generation with descriptor conditional recurrent neural networks. *Nature Machine Intelligence*, 2(5): 254–265, 2020c.

Landrum, G. RDKit: Open-source cheminformatics. URL http://www.rdkit.org.

Li, Y., Tarlow, D., Brockschmidt, M., and Zemel, R. Gated graph sequence neural networks. 2017.

Li, Y., Zhang, L., and Liu, Z. Multi-objective de novo drug design with conditional graph generative model. *Journal of Cheminformatics*, 10(1):33, 2018.

Ma, T., Chen, J., and Xiao, C. Constrained generation of semantically valid graphs via regularizing variational autoencoders. In *Advances in Neural Information Processing Systems*, volume 31, pp. 7113–7124. Curran Associates, Inc., 2018.

McKeown, K. *Text generation*. Cambridge University Press, 1992.

Mendez, D., Gaulton, A., Bento, A. P., Chambers, J., De Veij, M., Félix, E., Magariños, M., Mosquera, J., Mutowo, P., Nowotka, M., Gordillo-Marañón, M., Hunter, F., Junco, L., Mugumbate, G., Rodriguez-Lopez, M., Atkinson, F., Bosc, N., Radoux, C., Segura-Cabrera, A., Hersey, A., and Leach, A. ChEMBL: Towards direct deposition of bioassay data. *Nucleic Acids Research*, 47 (D1):D930–D940, 2018.

Mercado, R., Rastemo, T., Lindelöf, E., Klambauer, G., Engkvist, O., Chen, H., and Bjerrum, E. J. Graph networks for molecular design. *Machine Learning: Science and Technology*, 2(2):025023, 2021a.

Mercado, R., Rastemo, T., Lindelöf, E., Klambauer, G., Engkvist, O., and Bjerrum, E. J. GraphINVENT, 2021b. URL https://github.com/MolecularAI/GraphINVENT/releases/tag/v1.0.

Neil, D., Segler, M., Guasch, L., Ahmed, M., Plumbley, D., Sellwood, M., and Brown, N. Exploring deep recurrent models with reinforcement learning for molecule design. *OpenReview.net*, 2018.

Olivecrona, M., Blaschke, T., Engkvist, O., and Chen, H. Molecular de-novo design through deep reinforcement learning. *Journal of Cheminformatics*, 9(1):1–14, 2017.

Polykovskiy, D., Zhebrak, A., Sanchez-Lengeling, B., Golovanov, S., Tatanov, O., Belyaev, S., Kurbanov, R., Artamonov, A., Aladinskiy, V., Veselov, M., et al. Molecular sets (MOSES): A benchmarking platform for molecular generation models. *Frontiers in Pharmacology*, 11, 2020.

Popova, M., Isayev, O., and Tropsha, A. Deep reinforcement learning for de novo drug design. *Science Advances*, 4 (7):eaap7885, 2018.

Putin, E., Asadulaev, A., Ivanenkov, Y., Aladinskiy, V., Sanchez-Lengeling, B., Aspuru-Guzik, A., and Zhavoronkov, A. Reinforced adversarial neural computer for de novo molecular design. *Journal of Chemical Information and Modeling*, 58(6):1194–1204, 2018.

Sanchez-Lengeling, B., Outeiral, C., Guimaraes, G. L., and Aspuru-Guzik, A. Optimizing distributions over molecular space. an objective-reinforced generative adversarial network for inverse-design chemistry (organic). *ChemRxiv preprint 10.26434/chemrxiv.5309668.v3*, 2017.

Segler, M. H., Kogej, T., Tyrchan, C., and Waller, M. P. Generating focused molecule libraries for drug discovery with recurrent neural networks. *ACS Central Science*, 4 (1):120–131, 2018.

Smith, L. N. and Topin, N. Super-convergence: Very fast training of neural networks using large learning rates. In *Artificial Intelligence and Machine Learning for Multi-Domain Operations Applications*, volume 11006, pp. 1100612. International Society for Optics and Photonics, 2019.

Stokes, J. M., Yang, K., Swanson, K., Jin, W., Cubillos-Ruiz, A., Donghia, N. M., MacNair, C. R., French, S., Carfrae, L. A., Bloom-Ackermann, Z., et al. A deep learning approach to antibiotic discovery. *Cell*, 180(4): 688–702, 2020.

You, J., Liu, B., Ying, R., Pande, V., and Leskovec, J. Graph convolutional policy network for goal-directed molecular graph generation. *arXiv preprint arXiv:1806.02473*, 2018.

Zhang, J., Mercado, R., Engkvist, O., and Chen, H. Comparative study of deep generative models on chemical space coverage. *Journal of Chemical Information and Modeling*, 2021.

Zhou, Z., Kearnes, S., Li, L., Zare, R. N., and Riley, P. Optimization of molecules via deep reinforcement learning. *Scientific Reports*, 9(1):1–10, 2019.

A. Hyperparameters

A.1. Generative model: GraphINVENT hyperparameters

The GraphINVENT model consists of two main components: the GGNN and the global readout block. The hyperparameter values chosen in both models are those found to work best in the original publication (Mercado et al., 2021a). Taking into account that, in the GGNN, the message size must be equal to the number of hidden node features, the effective parameters to optimise are shown in Table 2.

Table 2. Effective model parameters and their optimal values.

Parameter	Value
Number of hidden node features	100
Graph embedding size	100
Hidden size of MLPs in GGNN	250
Depth of MLPs in GGNN	4
Dropout probability in GGNN	0
Number of message passes	3
Hidden size of MLPs in global readout	500
Depth of MLPs in global readout	4
Dropout probability in global readout	0

The parameter values used in the multi-layer perceptrons (MLPs) of the GGNN are detailed in Table 3. Their weights are initialised from Xavier uniform distributions (Glorot & Bengio, 2010). The activation functions used are SELUs (Klambauer et al., 2017). The number of message passes in the message passing phase is also shown in Table 3. These parameters relate the message passing and graph readout parameters in the following way: *hidden node features* = *input features*, *message size* = *output features*, and *graph embedding size* = *output features*.

Table 3. Model hyperparameters used in the MLPs and the message passing phase of the GGNN in GraphINVENT.

Parameter	Value
Input features	100
Hidden features	250
Output features	100
Depth	4
Dropout probability	0
Message passes	3

The parameter values chosen in the MLPs of the global readout block are detailed in Table 4. Again, weights are initialised from an Xavier normal distribution and the activation functions are SELUs. The remaining parameters of the MLPs are chosen as needed to encode all necessary information for the probabilities of adding an atom, connecting two nodes or terminating a graph. These sizes depend on another hyperparameter, the maximum number of nodes in molecules in a given dataset.

Table 4. Common hyperparameters used in the MLPs of the global readout block in GraphINVENT.

Parameter	Value
Hidden features	500
Depth	4
Dropout probability	0

Other parameters which need to be specified in GraphINVENT are those related to the dataset and its features. We use a 'simple' version of GraphINVENT which ignores aromatic bonds, chirality, and hydrogens (neither explicit nor implicit). Additionally, we use canonical node orderings and allow a maximum number of heavy atoms of 72 (largest molecule in the DRD2 actives set).

Training of GraphINVENT is done using the Adam optimiser (Kingma & Ba, 2014) with no weight decay and the OneCycleLR (Smith & Topin, 2019) learning rate scheduler implemented in PyTorch. In the scheduler, we have used the default parameters but disabled learning rate 'warm-up'. Furthermore, we take as many steps as epochs and set the fraction of steps for increasing the learning rate to 0.05. Other parameters such as the initial and final learning rates and the batch size must be adjusted for each specific dataset via hyperparameter optimisation. For optimal training, we trained for 30 epochs, used an initial learning rate of 10^{-4}, a final learning rate of 10^{-7}, and a batch size of 1000 sub-graphs.

A.2. RL framework hyperparameters

When training the agent, we again used the PyTorch Adam optimiser with no weight decay and the OneCycle learning rate scheduler (same settings as before). We found an initial learning rate of 10^{-4}, together with a final learning rate of 10^{-6}, to work best during RL-based training.

A batch size of 64 *molecules* was used in all RL settings ($64 \times 26 \sim 1164$ *sub-graphs*), except for models in which the scoring function aimed to increase the size of the molecules, where it was necessary to reduce the batch size to 32 molecules (~ 832 sub-graphs) due to memory constraints. However, these batch sizes are comparable to the ones used for training GraphINVENT, where the batch size consists of 1000 sub-graphs.

The hyperparameter values used in the activity scoring function (Eq. 5) were $\sigma = 20$ and $\alpha = 0.5$, and were the result of hyperparameter optimisation. $\alpha = 0.25$ and 0.75 were also tried, though $\alpha = 0.5$ was found to work best. For the other scoring functions (Eqs. 3 and 4) we used $\sigma = 10$ and $\alpha = 0.5$, although these were not as thoroughly optimised.

B. GraphINVENT details

B.1. Action space

GraphINVENT uses both the node- and graph-level information to predict the action probability distribution, or APD, in the final (global) readout block. The APD specifies how to grow the input subgraphs, and is made up of three components: f_{add}, f_{conn}, and f_{term}.

f_{add} contains probabilities for *adding* a new node to the graph. f_{conn} contains probabilities for *connecting* the last appended node in the graph to another existing node in the graph. f_{term} is the probability of *terminating* the graph. f_{add} and f_{conn} are multi-dimensional tensors as they must encode for a variety of properties, including which atom to connect to, with which each atom type, the identity of the new atom, etc. However, as the APD is a 'vector' property, f_{add} and f_{conn} are flattened for concatenation with f_{term} before forming the final APD. The shapes/indices of the three (unflattened) APD components are described in detail in the original publication (Mercado et al., 2021a).

As it is a probability distribution, the APD for each graph should sum to 1. Using the learned node and graph embeddings, H^L and g respectively, each APD is computed as follows:

$$\begin{aligned}
f'_{add} &= \texttt{MLP}^{add,1}\left(H^L\right) \\
f'_{conn} &= \texttt{MLP}^{conn,1}\left(H^L\right) \\
f_{add} &= \texttt{MLP}^{add,2}\left([f'_{add}, g]\right) \\
f_{conn} &= \texttt{MLP}^{conn,2}\left([f'_{conn}, g]\right) \\
f_{term} &= \texttt{MLP}^{term,2}\left(g\right) \\
APD &= \texttt{SOFTMAX}\left([f_{add}, f_{conn}, f_{term}]\right)
\end{aligned}$$

Note that in practice this is done for a mini-batch of graphs simultaneously on a GPU.

B.2. Iterative molecular generation

To demonstrate how the APD is used in GraphINVENT, we show a schematic of the generation loop in Figure 5.

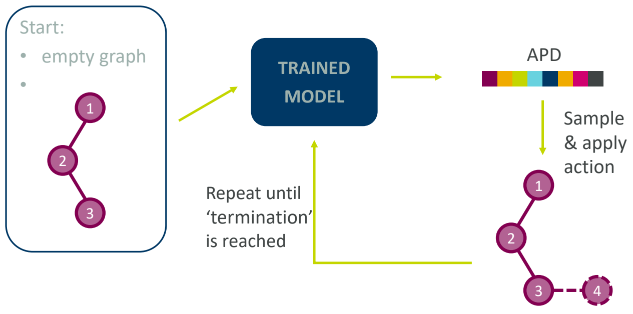

Figure 5. Schematic of the generation loop in GraphINVENT.