nature machine intelligence

Article

https://doi.org/10.1038/s42256-023-00636-2

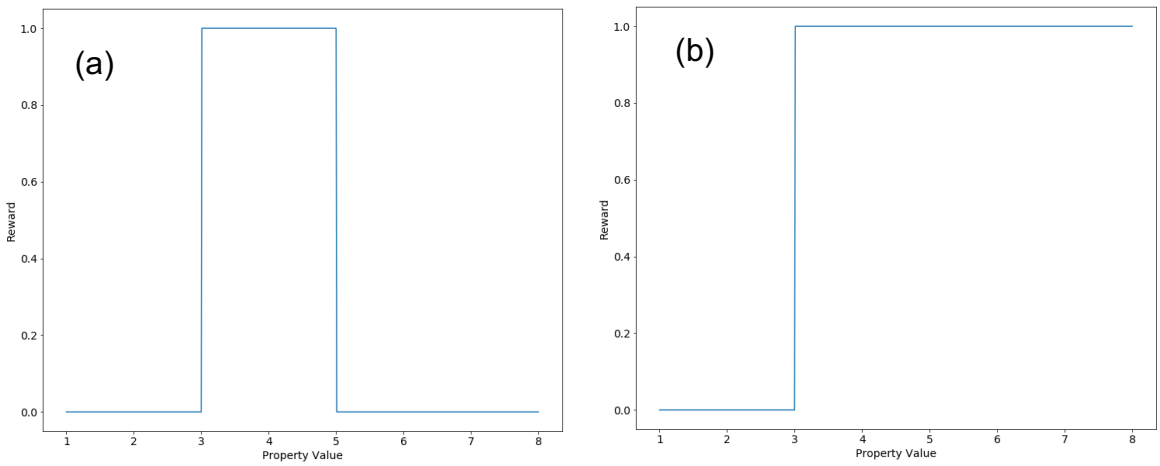
Testing the limits of SMILES-based de novo molecular generation with curriculum and deep reinforcement learning

In the format provided by the authors and unedited

Contents

1. Reinforcement Learning Reward Profiles

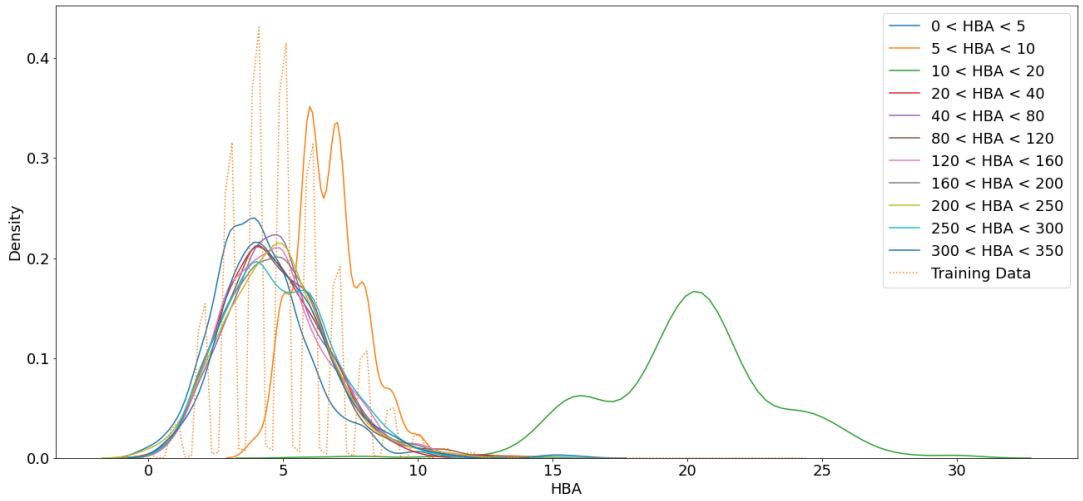
We used simple step reward functions in all our experiments. For each property, we selected a range of values that would return a positive reward, and then all SMILES generated within that range returned a value of one. All SMILES outside the reward range returned zero. For non-numerical optimisation tasks (for example, substructure generation), if optimisation was successful, the model returned a value of one; otherwise, zero was returned. Supplementary Information Figure 1 shows two example reward functions. Supplementary Information Figure 1a shows the profile of a reward range; all values within this range received a reward of one. Supplementary Information Figure 1b shows a reward profile with a limit where any value greater than 3 received a reward of one. Supplementary Information Table 1shows the reward profiles for all our property optimisation experiments.



Supplementary Information Figure 1: Example reward functions used in our study. (a) reward profile for experiment with a reward range. All molecules inside the range return a reward of one, all outside return zero. (b) reward profile for experiment with reward threshold. All molecules greater than the threshold return one, all other SMILES return zero. For a limit, the same shape would be observed, instead all SMILES less than the limit would return one, all other smiles would return zero.

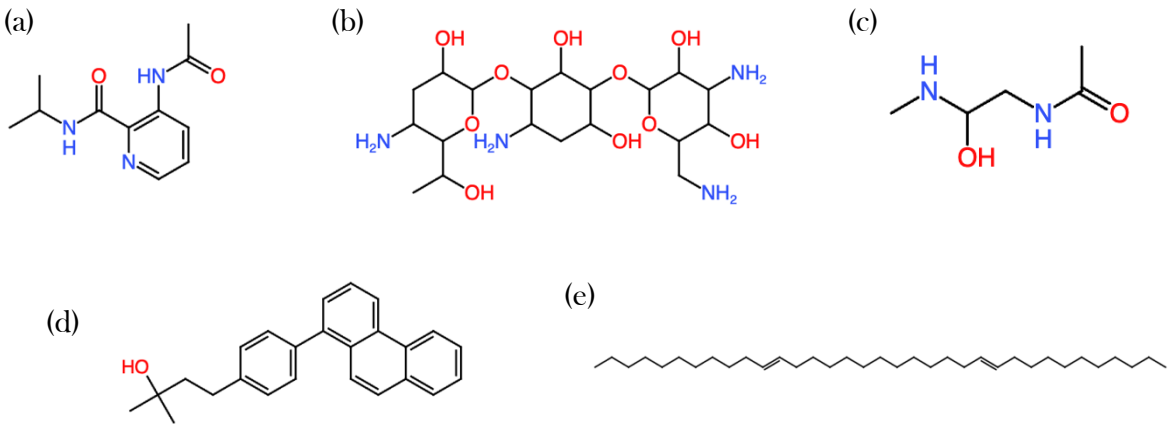
2. Control of generated libraries

Our results show how we can often control the property distribution of generated molecular sets by changing the reward profile during reinforcement learning. We show this to be possible between property-specific values; however, outside of these values, optimisation fails. We demonstrate this with cLogP and hydrogen-bond acceptor (HBA) count optimisation. Supplementary Information Figure 2 shows the distribution of all optimisation attempts for HBA count between 0 and 350 HBA's.



Supplementary Information Figure 2: Distribution of generated set for HBA count optimisation. Each line corresponds to the property distribution of the molecules sampled from an agent trained with a reward range detailed in the legend. The distribution of training data (dotted) is also provided.

Supplementary Information Figure 3 shows a series of molecules each generated with increasing cLogP reward profiles. We see a gradual increase in the cLogP of each molecule as the reward range increases; however, at the low extreme (-15 < cLogP < -10) optimisation fails and a molecule similar to the training data (cLogP between 0 and 10) is produced. Similar results are seen in Supplementary Information Figure 2 during hydrogen-bond (HBA) count optimisation. We saw gradual increases in the HBA count until about. 20, beyond this optimisation fails and molecules similar to the training data are produced.



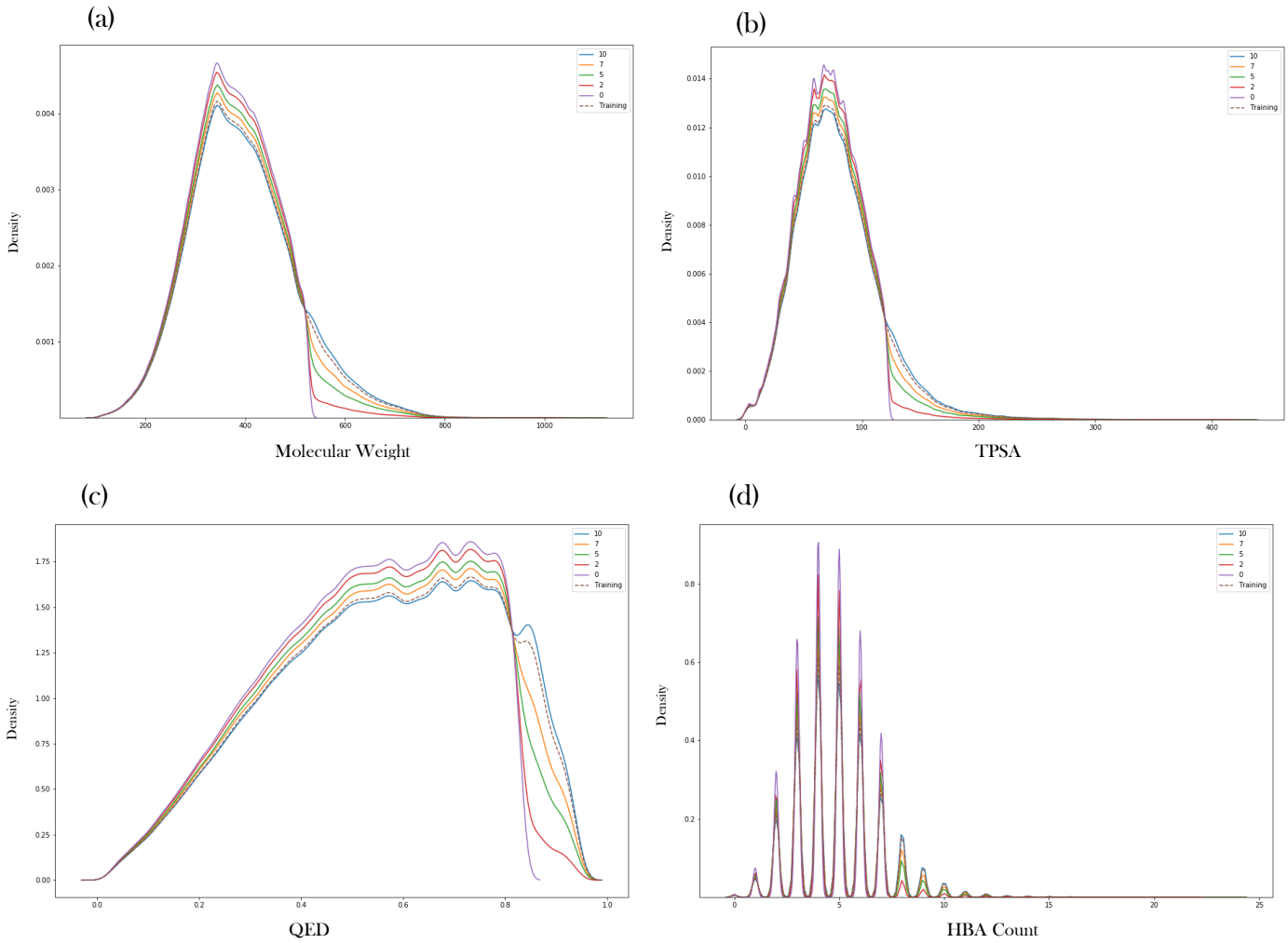
Supplementary Information Figure 3: Example molecules from cLogP optimisation. Each molecule was sampled from an agent optimised to produce molecules with cLogP between (a) -15 and -10, (b) – 10 and -5, (c) -5 and 0, (d) 5 and 10, (e) 10 and 15.

3. Effects of percentage representation

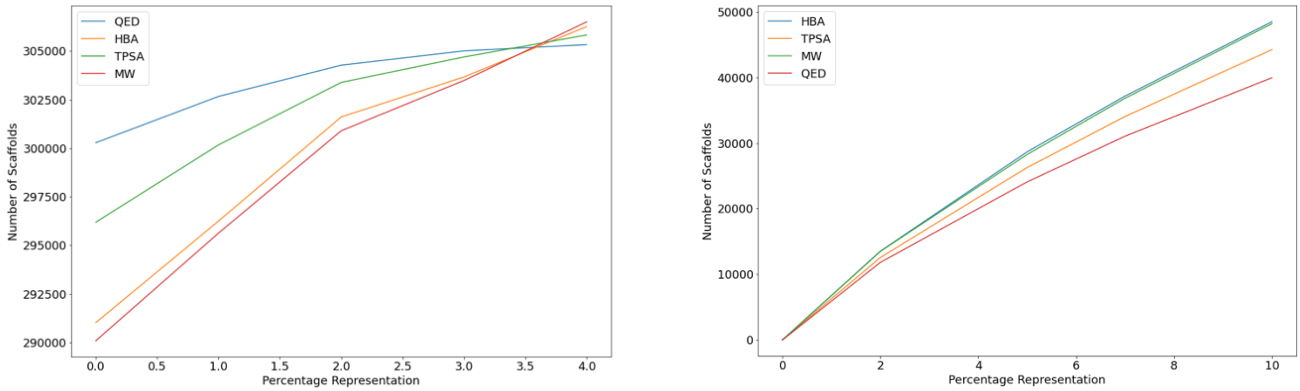
In all our experiments all training datasets were ~800,000 molecules, and contained either 0%, 2%, 5% 7% or 10% of the molecules within the reward profile. Supplementary Information Table 1 detailed the reward thresholds for each property in this experiment. Valid molecules with a property greater than the threshold received a reward of one. All other molecules returned zero reward.

Supplementary Information Table 1: Reward thresholds and dataset sizes for percentage representation experiments.

Property	Dataset Size	Reward Threshold
Hydrogen Bond Acceptors	798,471	8
Molecular Weight	799,248	527
TPSA	799,063	122
QED	797,569	0.83



Supplementary Information Figure 4: Training data distributions for four properties (a) Molecular weight, (bb) TPSA, (c) QED and (d) HBA count, and percentage representations tested.



Supplementary Information Figure 5: (a) Number of scaffolds present in each training dataset for each curated dataset across four properties. QED (blue), HBA (orange), TPSA (green), molecular weight (red). (b) Number of scaffolds in the reward range region of the same datasets.

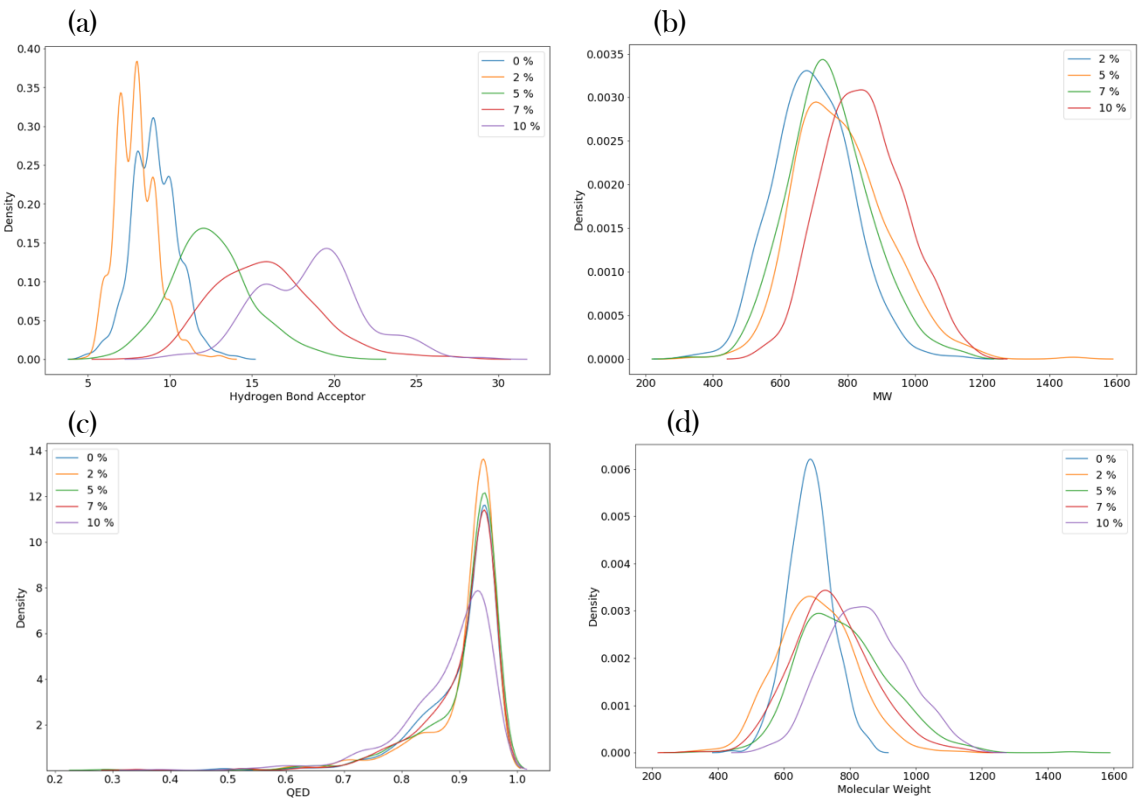
Supplementary Information Table 2: (a) Mean and (b) standard deviation of generated datasets property distributions for three properties at five different training data representations. The values are the average taken from three repetitions.

(a)

Property	**Mean of property distribution**				
	0	2	5	7	10
HBA	9.2	13.2	17.4	19.7	18.4
QED	0.900	0.906	0.908	0.903	0.891
TPSA	176	193	207	206	204
MW	624	715	748	735	787

(b)

Property	**Standard deviation of property distribution**				
	0	2	5	7	10
HBA	0.194	3.72	3.15	3.61	2.66
QED	0.00141	0.00128	0.00474	0.00125	0.0137
TPSA	3.13	3.69	5.36	3.25	4.92
MW	2.52	12.0	30.1	10.9	44.3



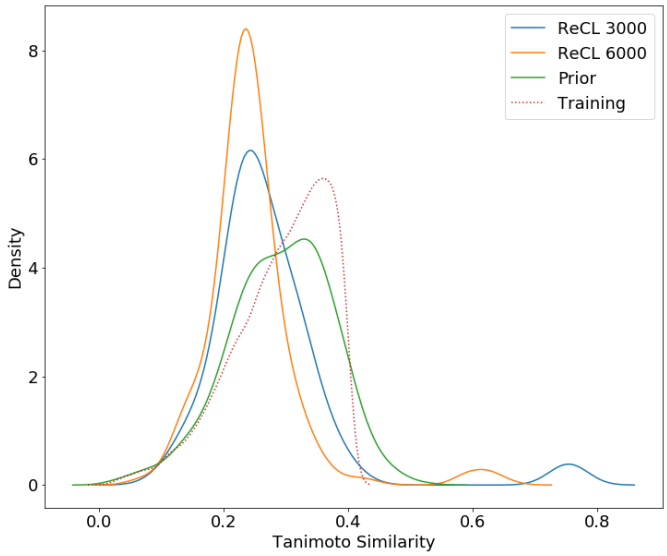
Supplementary Information Figure 6: Set of property distributions for four properties, (a) HBA, (B) molecular weight, (c) QED, (d) TPSA, that where the same optimisation regime was attempted for each using different training data. Each property was tested with 0% (blue), 2% (yellow), 5% (green), 7% (red) and 10% (purple) of the training data matching the reward profile. As the training representation increases, we see a shift in the property distribution. Suggesting that the percentage representation does affect the composition of generated sets of molecules. This figure shows one set of distribution. Quoted values in the paper were averages from three repetitions.

4. Comparison to other CL methods

In our comparison of SrIOP to a similar curriculum learning method (ReCL) we tested how well both methods can generate molecules similar to target structure without similar molecules in training data. In our results, we show that ReCL struggles with a complex target structure (Figure E6b), while SrIOP can generate similar molecules (Figure E7).

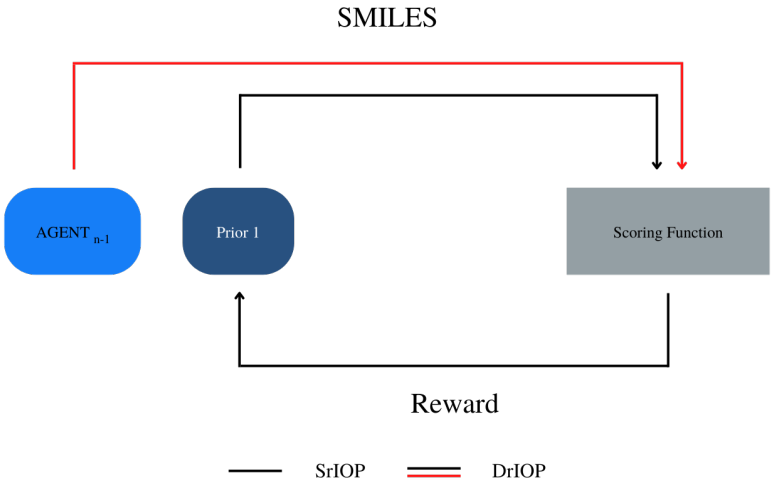
There are some key differences between our methods. In ReCL, the authors train their model across several intermediate tasks in a single training cycle. This cycle is allocated a maximum of 1500 training iterations. They only move onto the next objective once the model has fulfilled the current task and a threshold proportion of generated molecules match the desired reward profile. If the threshold is not met, the model cannot complete the overall task. In contrast, rIOP considers each intermediate task in a separate training cycle with its own reward function. Therefore, the agent parameters from the previous step are used to initiate the prior of the next

task. To ensure that any difference in performance is not a result of training iterations, we increased the total number of iterations allowed for ReCL from 1500 to 3000 and 6000 (Supplementary Information Figure 7). We found that this had no effect on the results as both models are unable to generate molecules with a tanimoto similarity greater than 0.8 to the target structure.



Supplementary Information Figure 7: Distributions of generated sets of molecules from ReCL for similarity generation task. Agent training was done for 3000 (blue) and 6000 (orange) (with) iterations to better match SrIOP training regime. Training (dotted) and prior model (green) distributions are provided for reference.

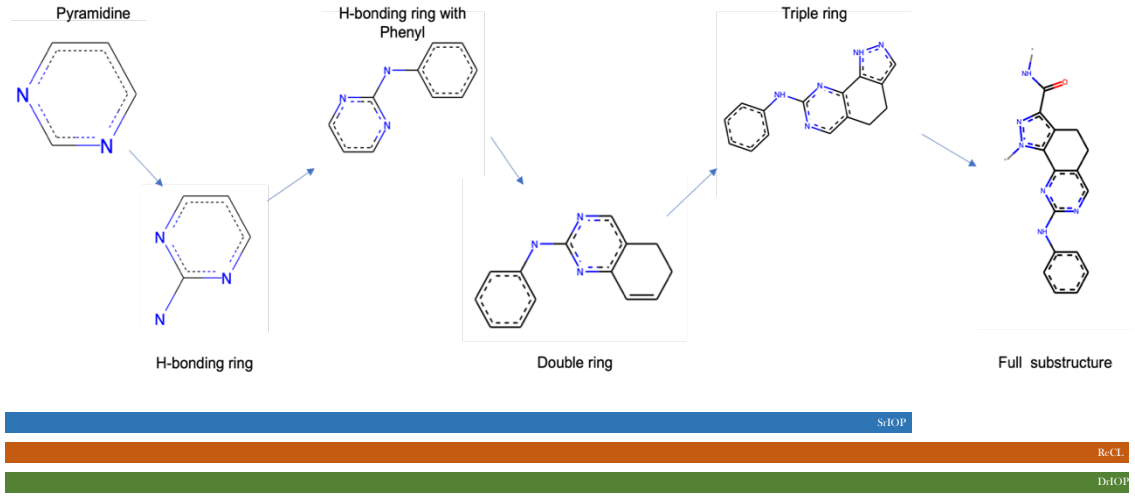
A second key distinction from the implementation by Guo et al. is the sampling of multiple models in training. DrIOP and SrIOP (Supplementary Information Figure 8). In both cases, the parameters of the previous step are used to initiate the prior of the current step. For SrIOP, only the current prior is sampled during training. However, for DrIOP the current prior and the agent from the previous step are sampled during training (but only the current prior parameters are updated). This prevents over-specialisation of the prior, which may prevent it being able to adapt to the current task.



Supplementary Information Figure 8: Diagram displaying our rIOP training regime. For SrIOP only the currently prior (which is initialised using the agent of the previous task) is sampled. For DrIOP both the current prior and the agent from the previous step are sampled.

5. Generating Complex Substructures

In their paper, Guo et al.[41], report how they use ReCL to generate a library of molecules that contain the dihydro-pyrazoloquinazoline scaffold (Supplementary Information Figure 9). They do this by training their agent to generate molecules containing a series of increasingly complex substructures that guide the agent toward the final task. To compare our methods, we carried out the same experiment. Supplementary Information Figure 9 shows each substructure and how, with our DrIOP method (blue) sampling from the two previous agents, we are also able to generate each substructure. Using DrIOP, we generated 103 molecules (from the 500



Supplementary Information Figure 9: Substructure tree used to generate complex unseen molecules. Each method attempts to generate each substructure before moving on. SrIOP (blue) can only generate molecules with the first five substructures. ReCL (orange) DrIOP (green) can generate all six.

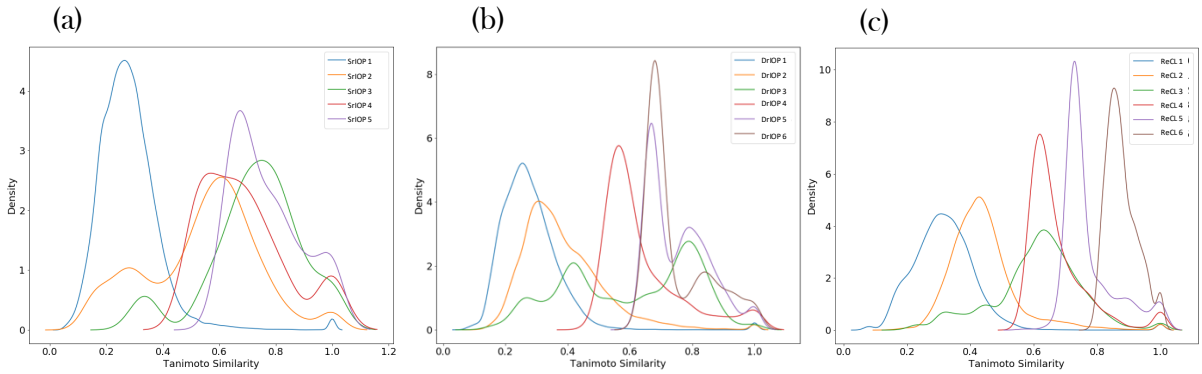
sampled) that included the final substructure across 21 distinct scaffolds. ReCL was able to generate more molecules with 451 molecules across 94 distinct scaffolds. This result suggests that DrIOP can generate molecules with increasingly complex substructures that baseline REINVENT is unable to generate.

6. Effects of Sampling from Multiple Agents

We propose two implementations of rIOP where the previous one or two agents are sampled during reinforcement learning. For simple optimisation tasks, we used only the previous agent (SrIOP). However, for more complex tasks, sampling from the previous two models (DrIOP) improved performance. For example, in Supplementary Information 5 using DrIOP, we were able to generate all six substructures; however, using SrIOP we only generated the first five.

We postulate that this is due to the specialisation of the agent at each step. For substructure generation, the success of each generation attempt is dependent on the model randomly generating a structure that maximises the reward. If, after an RL step, the current agent has learned to produce a very narrow selection of structures, the likelihood that it will randomly step out of this local minimum and produce a molecule that satisfies the next reward function is low. Therefore, over-specification at any step is harmful in the long run, and for the best performance, the model should learn to generate a diverse set of molecules that satisfy the reward function. Supplementary Information Figure 10 shows the pairwise tanimoto similarity of molecules sampled from the agents taken at each step of SrIOP (Supplementary Information Figure 10a), DrIOP (Supplementary Information Figure 10b) and ReCL (Supplementary Information Figure 10c) during a complex substructure generation experiment. The figure shows that for SrIOP and DrIOP, the first step leads to agents that produce diverse sets of molecules (blue) with mean tanimoto similarities ~0.2. However, for SrIOP we observe a significant increase in the similarity between molecules generated at steps 2 to 5. For DrIOP we see more pairs of molecules with low similarities for later steps (e.g., steps 2 and 3). Conversely, for ReCL we observe a gradual positive shift in the tanimoto similarity distributions as each new model becomes slightly more specialised. We believed that this would propagate forward and allow DrIOP to solve more complex reward functions.

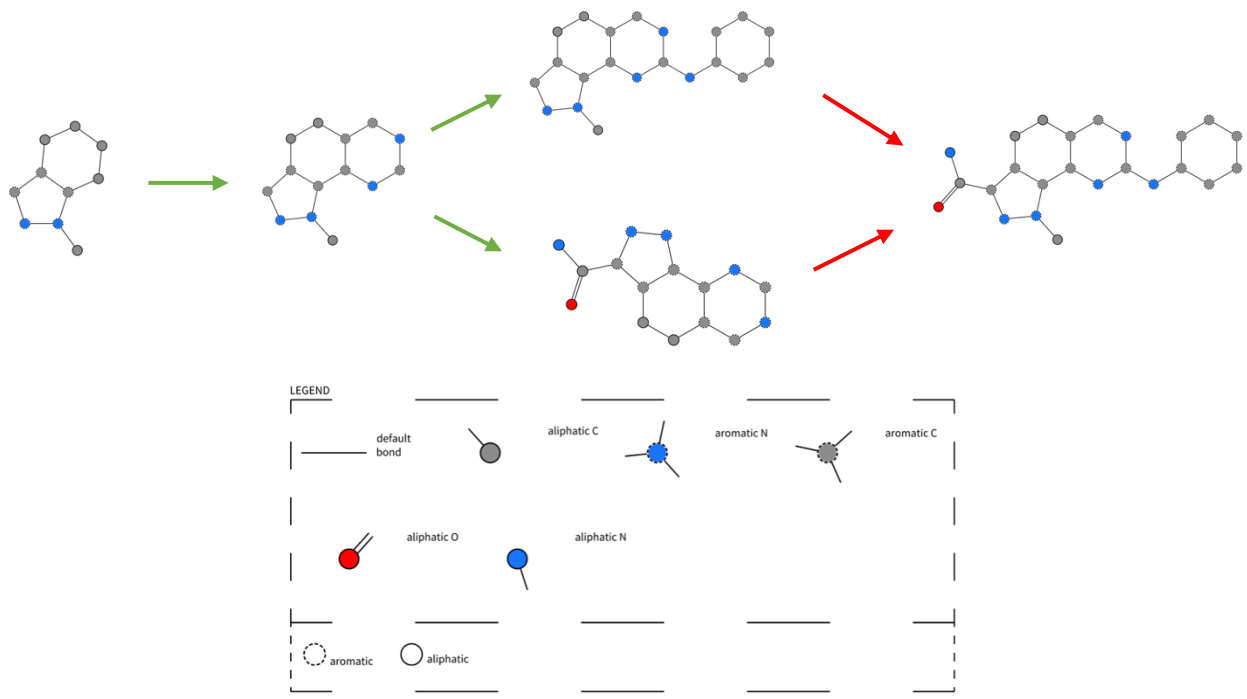
Despite their differences in performance, both implementations of rIOP are useful. The specification we observed with SrIOP allows the model to exploit a chemical property and generate a set of molecules with a very narrow property distribution. In contrast, long optimisation regimes will benefit from the reduced specificity of DrIOP.



Supplementary Information Figure 10: Comparison of model specialisation during (a) SrIOP, (b) DrIOP, and (c) ReCL. Tanimoto similarities of each pair of molecules in the generated set at each step by all methods were calculated. SrIOP shows large increase in similarity for later steps (e.g., 2 to 5) suggesting early specialisation. DrIOP and ReCL have more gradual increase in pairwise tanimoto similarities suggesting more gradual specialisation.

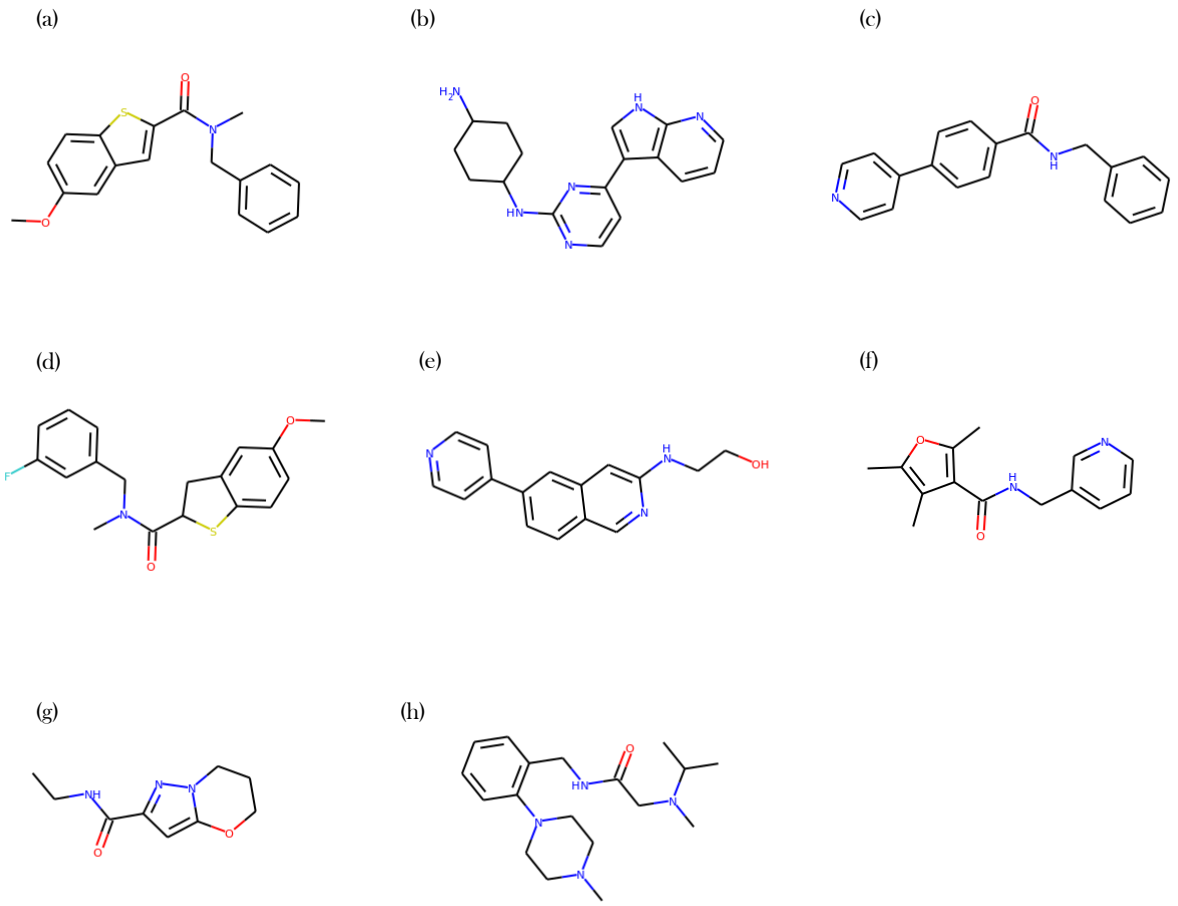
7. Limitations of SMILES

We show how the choice of SMILES directly affects the performance of molecular generators. We show that single model rIOP is unable to generate molecules that include a substructure given a series of the increasingly difficult substructures. Supplementary Information Figure 9 shows the series of SMILES used for both ReCL and rIOP. ReCL can generate all 6 substructures, whereas single model rIOP can only generate 3. We do show how DrIOP can also generate all six substructures. We suspected that altering the SMILES used would affect the performance of single model rIOP and found alternative routes (Supplementary Information Figure 11) that allowed more of the substructures to be generated. Although we did not find a route to the final structure, we believe that an exhaustive search of possible SMILES combinations would lead to successful substructure generation.

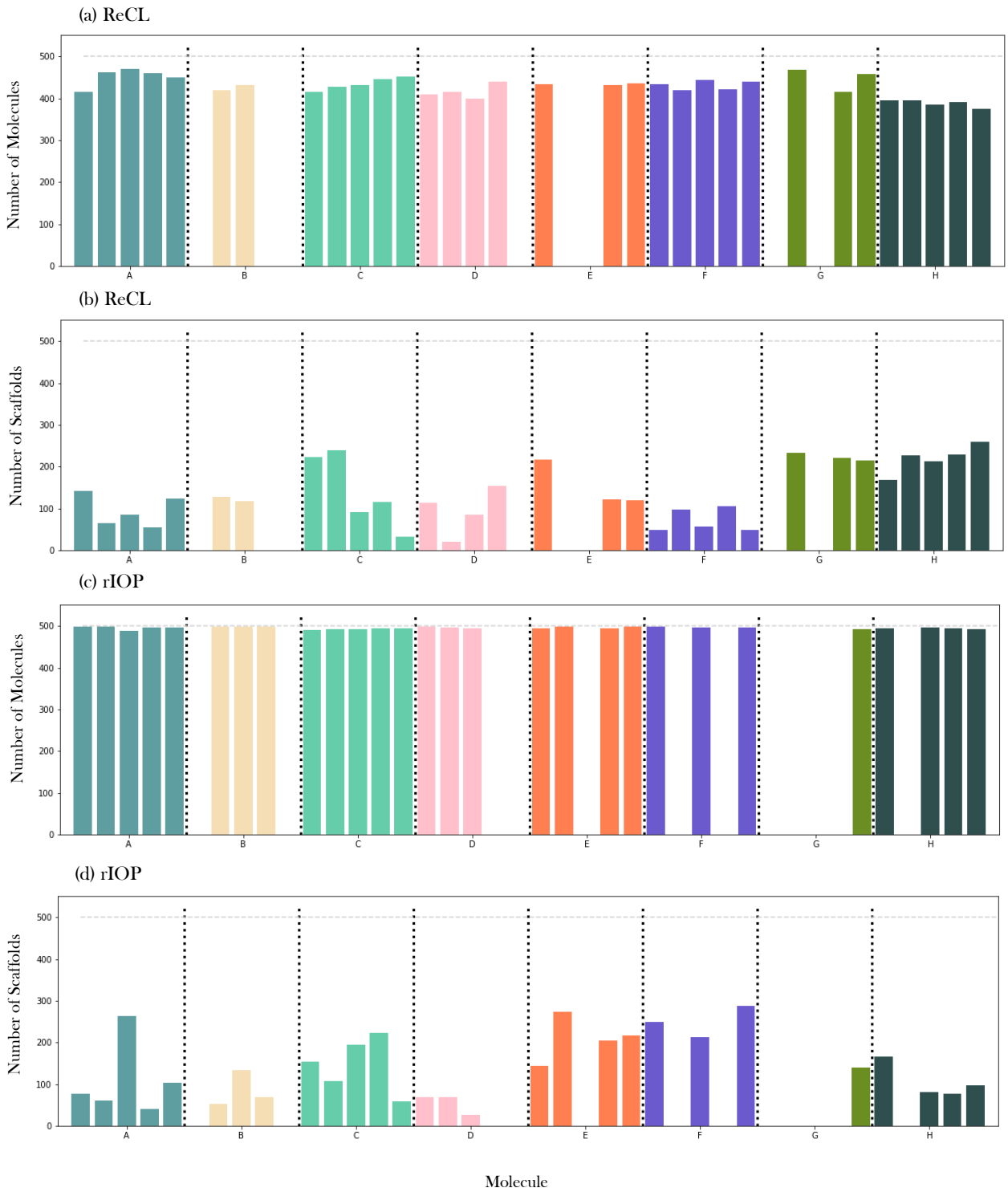


Supplementary Information Figure 11: Diagrams showing our various routes used in attempts to generate molecules that included the final substructure. Green arrows represent successful steps where the molecule following the arrow was included in generated molecules. Red arrows are unsuccessful steps.

We have shown the dependence of molecular generators on SMILES choice. To further this, we conducted an experiment on several different molecules from Chembl and curated a series of target substructures. Supplementary Information Figure 12 shows the molecules used in our experiments.

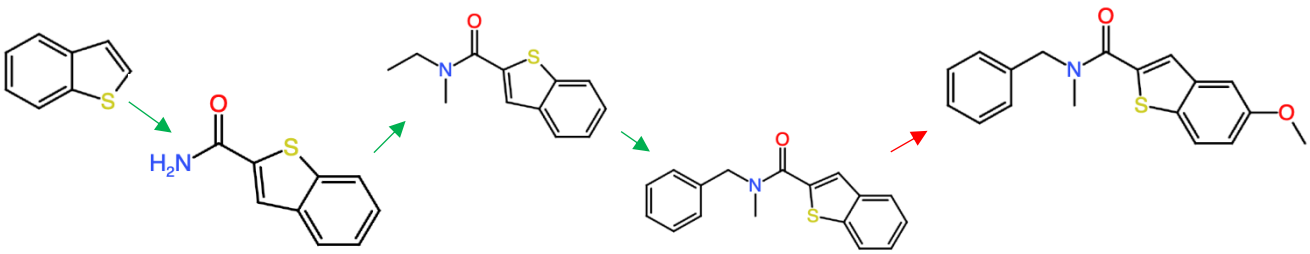


Supplementary Information Figure 12: Diagram of all molecules used to demonstrate the effects of SMILES choice on model performance.

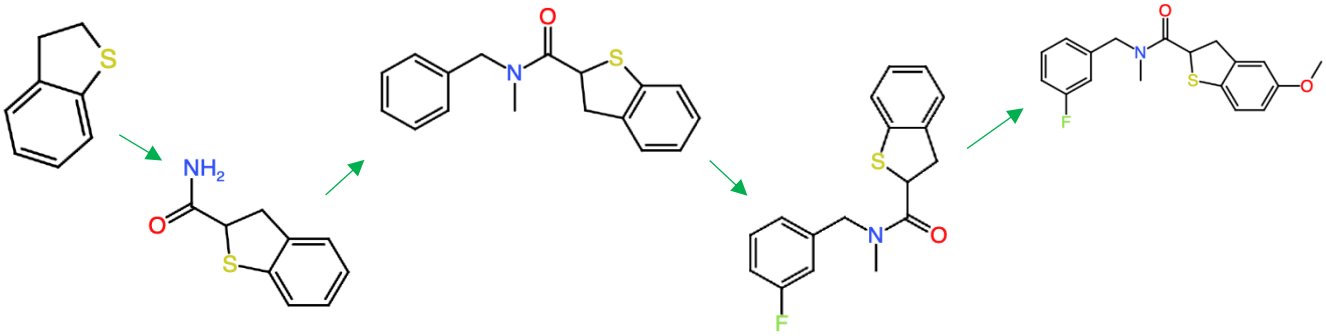


Supplementary Information Figure 13: Effect of SMILES on substructure optimisation performance using ReCL (a and b) and single model rIOP (c and d). For each molecule (A-H), 5 sets of intermediate SMILES were enumerated and used during optimisation. In (a) and (c) the coloured bars represent the total number of molecules generated by the final agent (500 molecules were sampled) that included the desired substructure (Supplementary Information Figure 12). In (b) and (d), the bars show the total number of distinct scaffolds in the same sample. The figure shows that the SMILES choice directly affects optimisation performance and diversity of generated molecules. For example, optimisation of molecule B failed (no molecules matching the desired substructure) three times using ReCL (a) and twice using rIOP (c) despite intermediate SMILES at each step across all sets representing the same structure.

(a)



(b)



Supplementary Information Figure 14: Diagram displaying the intermediate and final substructures we aimed to generate for two molecules. (a) Molecule A and (b) Molecule D from Supplementary Information Figure 12. The substructure following a red arrow was not successfully generated after optimisation. Substructures following a green arrow were generated. Despite the intermediates used for both molecules being very similar, both methods tested (ReCL and SrIOP fail on the last step for molecule A.

Supplementary Information Table 3: Number of molecules that include the target substructure and number of distinct scaffolds for each set of SMILES for (a) molecule A and (b) molecule D (Supplementary Information Figure 12).

(a)

Run	**Molecules with Substructure**		**Number of Scaffolds**	
	SrIOP		ReCL	
1	498	78	415	142
2	498	61	462	64
3	489	264	471	85
4	497	41	460	55
5	497	104	450	124

(b)

Run	**Molecules with Substructure**		**Number of Scaffolds**	
	SrIOP		ReCL	
1	499	69	410	114
2	497	68	416	21
3	494	27	400	85

4	0	0	440	155
5	0	0	0	0

Examples of all generated datasets can be found at https://github.com/m-mokaya/RIOP.git