Searching for High-Value Molecules Using Reinforcement Learning and Transformers

Anonymous Author(s)
Affiliation
Address
email

Abstract

Reinforcement learning (RL) over text representations can be effective for finding high-value policies that can search over graphs. However, RL requires careful structuring of the search space and algorithm design to be effective in this challenge. Through extensive experiments, we explore how different design choices for text grammar and algorithmic choices for training can affect an RL policy's ability to generate molecules with desired properties. We arrive at a new RL-based molecular design algorithm (ChemRLformer) and perform a thorough analysis using 25 molecule design tasks, including computationally complex protein docking simulations. From this analysis, we discover unique insights in this problem space and show that ChemRLformer achieves state-of-the-art performance while being more straightforward than prior work by demystifying which design choices are actually helpful for text-based molecule design.

1 Introduction

Molecular discovery can have a significant impact on our society, however, the vast search space makes it challenging to find high-value molecules. The potential of reinforcement learning (RL) methods to discover new, high-value molecules has resulted in a series of research work performed by RL researchers focusing on learning policies as graph neural networks (GNNs) [You et al., 2018, Zhou et al., 2019, Jin et al., 2020, Fu et al., 2022, Yang et al., 2021, Bengio et al., 2021]. In this formulation, the RL policy is trained to add atoms and bonds to a molecular graph representation. In this formulation there is a one-to-one mapping between molecules and their graph representation, making it easier to construct state and action spaces with Markovian dynamics. However, the action space in the graph formulation is vast as it consists of the product of candidate attachment positions and candidate attachment sequences. Graph-based data structures (such as adjacency matrices, trees, etc.) are a powerful representation used to describe a number of design problems, including social networks [Tan et al., 2019], transportation networks [Wang and Tang, 2021], recommendation systems [Chen et al., 2021b], and combinatorial optimization problems [Khadka et al., 2020, Miret et al., 2022] have been popular in this design space. However, GNNs are often difficult to train [Chen et al., 2022] and cannot readily take advantage of large-scale text data sets that effectively describe molecular structures and properties.

In order to take advantage of the richness of text-based representations for molecules, one can formulate the molecular search problem as the construction of tokens in a sequence that become a molecular text. The molecular texts formulated by common text-based representations, such as SMILES [Weininger, 1988] and SELFIES [Krenn et al., 2020], can then be converted into molecular graphs with cheminformatics libraries [Landrum et al., 2013] using their respective encoding and decoding rules. However, the text-based representation can be more difficult to formulate as an MDP since there is not always an exact one-to-one mapping between texts and molecules. In

Submitted to 37th Conference on Neural Information Processing Systems (NeurIPS 2023). Do not distribute.

fact, the text-to-molecule conversion can be many-to-one, where the complexity of the dynamics in the MDP given by many-to-one mappings is non-trivial. On the other hand, the action space in molecular text design can be significantly reduced given the rules of text construction imposed by a given representation. Moreover, formulating molecule discovery as sequence-generation has the potential to capitalize on recent successes in natural language modeling [Brown et al., 2020a].

In this paper, we perform a detailed empirical study of molecular discovery using text-based RL across more than 25 molecular properties relevant for drug-discovery, including docking simulation for molecular ligands [García-Ortegón et al., 2022, Huang et al., 2022] and develop our own algorithm (MoLRL) based on state-of-the-art literature as shown in Table 1. In our experiments, we evaluate two molecular text representations (SMILES, SELFIES) and the use of three neural network architectures (Multi-Layer Perceptron [Bengio et al., 2003], Recurrent Neural Network [Schmidt, 2019], Transformer [Vaswani et al., 2017]) pretrained on 5 datasets of varying quality and sizes. We create ChemRLformer that achieves the highest performance across these tasks while being much simpler than previous text-based RL algorithms [Blaschke et al., 2020a, Gao et al., 2022]. Via our detailed ablation study, we construct ChemRLformer and find that pretraining on *aligned* datasets can significantly improve performance across all molecular design tasks, even exceeding the performance of agents pretrained on 100 times larger datasets. We also show that targeted algorithmic design, such as hill-climbing in the replay buffer and regularization, further increases the performance of ChemRLformer. To the best of our knowledge, ChemRLformer is the largest analysis of text-based RL methods for molecule discovery.

Table 1: Table showing conceptual comparisons of various text based molecular optimization methods. MoLRL combines the most successful elements of prior work.

Method	Text Representation	RL	Architecture	Pretraining	Algorithmic Components
Literature Methods					
SMILES-VAE [Gómez-Bombarelli et al., 2018]	SMILES	✗	VAE	✓	Maximum Likelihood
SMILES-LSTM [Brown et al., 2019]	SMILES	✗	LSTM	✓	Maximum Likelihood
BOSS [Moss et al., 2020]	SMILES	✗	VAE	✗	Bayesian Optimization
REINVENT [Blaschke et al., 2020a]	SMILES	✓	GRU	✓	Replay buffer, KL
REINVENT 2.0 [Blaschke et al., 2020b]	SMILES	✓	GRU	✓	HC-Replay buffer, Log p, KL
STONED [Nigam et al., 2021]	SELFIES	✗	FC	✗	Genetic algorithm
Pasithea [Shen et al., 2021]	SELFIES	✗	FC	✗	Deep dreaming
ChemRLformer (Ours)	SMILES, SELFIES	✓	Transformer, FC	✓	Replay buffer, KL

2 Related Work

RL for Design and Discovery: Many methods in diverse fields leverage RL to help augment a prior design method to improve performance [Yu et al., 2018, Schaff et al., 2019]. Other methods have explicitly included the design process in the RL loop by training design problems together [Chen et al., 2021a, Ha, 2019, Luck et al., 2020, Kumar et al., 2022] with most prior work focusing on robot and agent design, not molecular design. Our molecular design work creates an autoregressive structure that grows the size of the state as the agent acts in the environment.

Molecular Discovery Using Sequence-Based Methods: Sequence-based methods treat molecular design as a sequence of tokens that get concatenated in order. Generative models for sequence-based methods span a diverse range, including variational autoencoders (VAEs) [Gómez-Bombarelli et al., 2018, Alperstein et al., 2019], recurrent neural networks (RNNs) [Gupta et al., 2018, Bjerrum and Threlfall, 2017, Grisoni et al., 2020, Flam-Shepherd et al., 2022] and transformer models[Wang et al., 2019, Fabian et al., 2020, Edwards et al., 2022a, Zeng et al., 2022, Taylor et al., 2022]. The general procedure for all the above methods is to perform self-supervised generative learning to sample molecules similar to the original dataset. MoLRL can also make use of pretrained generative models, which we then fine-tune using reinforcement learning to produce enhanced molecules.

Molecular Discovery Using Search-Based Methods: Although sequence-based molecule generation methods often provide a more structured way of learning molecular distributions, search-based methods generally have the advantage of being able to directly find molecules based on a desired property. Although a wide range of graph-based RL methods [You et al., 2018, Zhou et al., 2019, Jin et al., 2020, Fu et al., 2022, Yang et al., 2021, Bengio et al., 2021] for optimizing molecules exist, graph-based state representations introduce significant complexity to the RL problem formulation, both in the transition dynamics and action space. By contrast, text-based

methods are simpler and also relatively under-explored, motivating our focus on these methods in this work. Moreover, recent work [Cieplinski et al., 2021, Gao et al., 2022] has shown that an older text-based method REINVENT [Olivecrona et al., 2017] outperforms more complex graph-based RL methods. Some limited extensions to Olivecrona et al. [2017] have been explored, including experimenting with a newer molecular grammar designed for robust molecule generation [Gao et al., 2022]. However, there has been limited work proposing the use of language models and text-based RL for molecular discovery. Additionally, there have been limited efforts to incorporate recent advancements from the language modeling domain into these methods. For example, the a character-level LSTM network architecture used in Olivecrona et al. [2017], has not been revisited despite significant recent advances in sequence modeling [Vaswani et al., 2017, Brown et al., 2020b].

3 Background

The algorithms detailed in this paper are built on top of a foundation of reinforcement learning, text-based molecule representations, and language modeling.

Reinforcement Learning: Reinforcement learning can be used to learn policies for sequential decision-making problems. Policies are optimized based on an environment that is described as a Markov Decision Process (MDP). A discrete MDP is defined by the tuple $\langle \mathcal{S}, \mathcal{A}, \mathcal{T}, r, \gamma \rangle$ where $\mathcal{S}$ is the state space, $\mathcal{A}$ is the action space, $T : \mathcal{S} \times \mathcal{A} \times \mathcal{S}' \to [0, 1]$ is the transition function, $r : \mathcal{S} \times \mathcal{A} \to \mathcal{R}$ is the reward function and γ is the discount rate.

For actions $a_t \in \mathcal{A}$ and states $s_t \in \mathcal{S}$, the goal of reinforcement learning is to learn a policy $\pi_\theta(a_t|s_t)$ which maps states to actions, such that:

$$\pi_\theta(a_t|s_t) = \arg\max_\theta \mathbf{E}_{p(\tau|\theta)} \left[\sum_{t=0}^{T} \gamma^t r(s_t, a_t) \right] \tag{1}$$

where $p(\tau|\theta)$ is the distribution over trajectories induced by π_θ and the transition function $\mathcal{T}$.

Text representations for molecules: Molecules are most naturally described using a graph structure of atoms and bonds. However, graph-based deep learning models can be difficult to train, especially at large scale [Dwivedi et al., 2022, Geisler et al., 2023]. Recent works have proposed a variety of text representations for molecules [Weininger, 1988, Krenn et al., 2020, Heller et al., 2013, Krenn et al., 2022, Cheng et al., 2023], each having their distinct advantages and shortcomings. In this study, we focus on the two most commonly used representations: SMILES [Weininger, 1988] and SELFIES [Krenn et al., 2020]. Any text representation for molecules consists of a set of valid tokens, which may represent individual atoms or special characters that imply the presence of certain structures, as well as the encoding and decoding rules needed to convert between the text representation and the graph representation of a molecule. Valid texts under a grammar are those which respect both the vocabulary and the encoding/decoding rules for that grammar and, hence, can be converted into a graph representation of a molecule. SELFIES, which was developed in response to the tendency for SMILES-based deep learning models [Gó mez-Bombarelli et al., 2018, Jin et al., 2018] to generate invalid molecular texts, has the useful property of providing a conversion for *any* text into a graph corresponding to a molecule, provided the tokens in the text respect the SELFIES vocabulary. For example, the text representation of Benzene in SMILES is C1=CC=CC=C1 while in SELFIES one possible representation is [C][=C][C][=C][C][=C][Ring1][=Branch1].

Language modeling: Language modeling often relies on the self-supervised task of next-token prediction for model pretraining. The general framework for next-token prediction is to train a model to predict the next token in a sequence autoregressively, i.e. given the previous tokens in the sequence (left context). Many architectures to handle sequential data have been proposed: Recurrent Neural Networks (RNNs) [Hochreiter and Schmidhuber, 1997, Rumelhart and McClelland, 1987] are a class of models used in sequence modeling which use recursive connections in hidden layers to accumulate the left context for next-token prediction. Transformers are a more recent architecture that instead use a self-attention mechanism [Vaswani et al., 2017] to capture dependencies between all tokens in a sequence. For next-token prediction tasks, attention masking is used to enforce left context, meaning that representations for tokens later in the sequence are only allowed to attend to previous tokens in the sequence. In Section 4 we outline how we pretrain an autoregressive sequence model to predict sequences of known molecules.

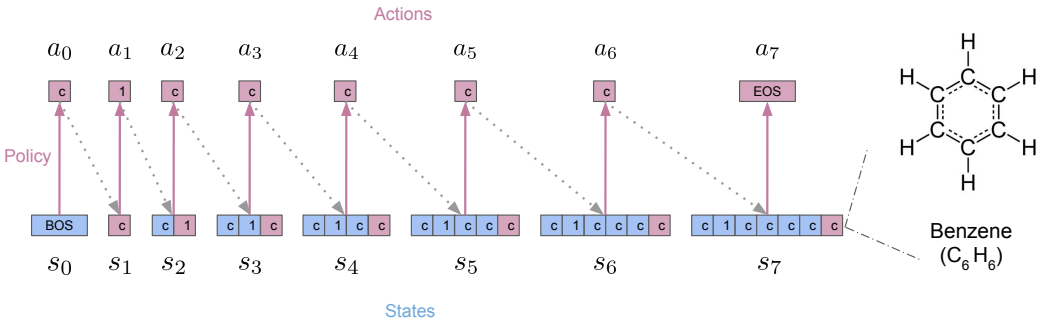


Figure 1: **Autoregressively generating a benzene molecule.**: An autoregressive model for sequence generation can be viewed as an RL policy where the actions a_t are the next tokens to append to the sequence and the state is the concatenation of all actions taken up to time $t-1$. A special end-of-sequence token can terminate the episode early at time T. The text at time s_T is then converted into a molecule based on the text-representation grammar and then scored according to a scoring function that measures the alignment of the molecule with the desired properties informed by the application. Hydrogen atoms are added at the end to complete the structure.

4 ChemRLformer Generating molecular strings via reinforcement learning

The molecular design space is complex but the benefit from finding improved options is great. In this section, we describe ChemRLformer and how combining language models and tools from RL produces a sota algorithm.

MDP for molecule generation: The vocabulary and grammar for text representations of molecules can be interpreted as an MDP as described in Section 3, where the states s_t correspond to a *variable length* text of accumulating tokens, and the actions a_t correspond to vocabulary defined by the text-representation. The transition function is a deterministic function where the action a_t taken by the agent is appended to the end of the state s_t resulting in s_{t+1} using the dynamics $s_{t+1} = [s_t, a_t] \leftarrow T(s_t, a_t)$. However, the corresponding transition function induced in the graph representation of molecules is more complex as it is determined by the encoding/decoding rules of the chosen text representation. For example, in the SMILES grammar, a random concatenation of tokens may not correspond to a valid molecule, while the SELFIES grammar is constructed such that any ordering of its tokens is encoded as a valid molecule.

Finally, the reward function $\mathcal{R}$ scores molecules according to their alignment with desired chemical properties, which can involve complex material simulations. The underlying property computation of the reward function further informs the dynamics of the MDP imposed by text representation. For example, docking scores are used to estimate the binding affinity between ligands and protein targets. We discuss reward functions for molecules in more detail in section Section 5.

Pretraining policies for molecule discovery. To advance effectively within this vast search space, we make use of datasets containing a large number of drug-like molecules in text format [Irwin et al., 2012, Sterling and Irwin, 2015b, Mendez et al., 2019]. This data is used to train an autoregressive model to predict tokens that conform to the grammar for drug-like molecules, instead of the random texts that are generated from a randomly initialized policy, thereby significantly simplifying the exploration problem. In particular, we pretrain a network p_ϕ on the self-supervised objective of next-token prediction. Although large language models can be trained with other objectives, such as corrupted text reconstruction [Edwards et al., 2022b], these models are not a good fit for our purposes since they cannot generate diverse and valid molecules without access to carefully designed prompts.

$$\min_\theta \mathbb{E}_{A\sim D}\left[\sum_{t=1}^{H} -\log p_\theta(a_t = A_t \mid A_{t-1}, \cdots A_0)\right]. \tag{2}$$

In practice a minibatch of sequences $\{A^1, \cdots, A^m\}$ are sampled from the prior dataset D to evaluate the loss function in Equation 2, and the parameters are trained using gradient descent.

4.1 RL for molecule generation

To generate a molecule, a ChemRLformer policy $\pi_\theta(a_t \mid s_t)$ is allowed to autoregressively sample tokens for a fixed number of timesteps H. The start state s_0 is always a beginning-of-sequence token [BOS], and the agent can terminate early by taking the end-of-sequence action [EOS]. Figure 1, shows how an RL policy can construct a Benzene molecule. Since we are only interested in the properties of the final molecule, there are no intermediate rewards and the goal of the RL policy is to maximize the expected scalar reward corresponding to the final constructed molecule, $r(s_T)$. Thus, assuming a discount rate $\gamma = 1$, Equation 1 can be rewritten more simply as:

$$\max_\theta \mathbb{E}_{s_T \sim \pi_\theta} [r(s_T)] \tag{3}$$

$$\text{where } s_T = [\text{BOS}][a_0][a_1] \cdots [\text{EOS}], \text{ is sampled autoregressively from the policy.}$$

Our experiments use the policy gradient algorithm [Sutton et al., 1999b] to train the RL policy because it is known to achieve state-of-the-art performance amongst RL for molecular optimization [Olivecrona et al., 2017]. Deep RL policies are able to learn the non-linear global structures of molecular texts which, as we show in section 5, enables them to generalize to novel and diverse molecules. However, training RL policies from scratch is time-consuming and can make the exploration problem infeasibly difficult. Next, we explain how we adapt recent language modelling techniques to pretrain the RL policy.

RL fine-tuning The pretrained model can directly be used to sample novel drug-like molecules. These molecules, however, are not optimized for any particular property. Note that given our definition of the state s_t as the concatenated history of all previous actions, this pretrained network is exactly analogous to the policy network in Equation 1. Hence, by initializing $\pi_\theta = p_\phi$, and $\theta = \phi$, we can fine-tune this pretrained network by optimizing Equation 1 via the policy gradient algorithm - REINFORCE [Sutton et al., 1999a]. We need only to define a reward function $r(s_T)$ which scores molecules according to their alignment with the desired properties. In the following experiments, we show that this fine-tuning is vital for ChemRLformer to sample better molecules. We also highlight the importance of pretraining and study how the size and quality of the prior data affect the downstream ability of RL to search for high-value molecules.

5 Experimental Results

Our proposed algorithm ChemRLformer uses the best combinations of choices resulting from assessing the performance across three dimensions: (1) what pretraining factors are important to improve RL for molecular discovery (Section 5.2), (2) how the use of recent text-based molecule grammars facilitates downstream RL exploration (Section 5.3); and, lastly, (3) which specific algorithmic changes are necessary to improve RL performance (Section 5.4).

5.1 Experimental Setup

Tasks. We evaluate ChemRLformer against five different **docking** targets [Alhossary et al., 2015] (fa7, parp1, 5ht1b, jak2, and braf) previously explored in the literature [Yang et al., 2021, Lee et al., 2023]. The docking scores used to estimate the binding affinity between ligands and protein targets are a complex function of the global molecular structure and have been proposed as chemically relevant benchmarks for molecule design algorithms [Cieplinski et al., 2021, Tripp et al., 2022]. In addition to the docking targets, we also evaluate on 22 pharmaceutically-relevant oracle functions [Huang et al., 2021, Gao et al., 2022, Brown et al., 2019] (**pytdc** tasks), which include tasks such as optimizing proxies of bioactivity, similarity to target molecules, and combinations of multiple physiochemical drug properties.

Evaluation metrics. We design our evaluation procedure with the final goal of identifying the best candidates to test in a wet lab. To discover such high-value candidate molecules, we use sota simulators that assign rewards to molecules by performing complex docking simulations [Alhossary et al., 2015] or using proxy models and chemical rules [Huang et al., 2021]. Previous works limit the number of molecules sampled during evaluation to around 3000 for **docking** tasks [García-Ortegón et al., 2022, Yang et al., 2021, Lee et al., 2023] and 10000 for **pytdc** tasks [Gao et al., 2022, Brown et al., 2019] due to the computational cost associated with these reward simulators. We allow up to

25000 unique oracle calls and up to 40000 total oracle calls (allowing repeats). We argue this better reflects the lower cost and availability of computing resources relative to wet-lab resources. From all the sampled molecules, the average score of the top-k ($k = 1, 10, 100$) molecules is used as a performance metric. These top groups are an estimate of the algorithm's ability to discover a group of top-quality candidates that could be given to a wet lab for thorough testing. We report **pytdc** scores on a normalized basis between zero and one by default. Next, we normalize all docking scores by dividing them by -20 in our experiments. Additionally, we report *diversity*, defined as the averaged internal distance of the top 100 molecules, and *redundancy*, defined as the total number of oracle calls that an agent makes for an already evaluated molecule.

Pretraining. We study how the quality and size of prior data affect the downstream RL performance of ChemRLformer by pretraining a GPT [Radford et al., 2018] style transformer model on five datasets of varying sizes and quality and using the pretrained model as an initialization for the RL agent's policy network. See Table 2 for the name, size, and description of all datasets used in our work. We also rank all datasets based on their quality on **docking** and **pytdc** tasks. We determine the quality of a dataset by the performance of molecules sampled from the model pretrained on that dataset. The quality of ChemRLformer's pretrained model is evaluated using the *top-100* molecules sampled by the pretrained model under the same evaluation setup in Appendix A.2. By default, these open-sourced datasets contain a large number of drug-like molecules in SMILES format. For our experiments, we also convert all datasets to the SELFIES format. Lastly, three different architectures are compared: **fully-connected (FC)**, **recurrent (RNN)** - a GRU and **transformer** - GPT style autoregressive model, and compare them on downstream RL tasks.

Table 2: **Description of molecular datasets used for pretraining:** Datasets are ranked according to procedure described in Section 5.1. Two datasets have the same rank if their average performance lies inside one standard error of the other. The datasets are drawn from a subset of the Zinc [Sterling and Irwin, 2015b, Irwin et al., 2022] and ChemBL [Gaulton et al., 2012] databases.

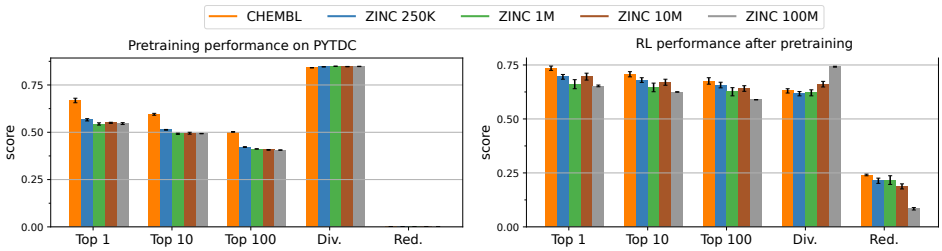
Dataset	Size	Docking Rank	Pytdc Rank	Description
CHEMBL	1.2 M	1	1	Manually curated database of bioactive molecules with drug-like properties [Gaulton et al., 2012].
ZINC 250K	250 K	2	2	ZINC database molecules curated for their pharmaceutical relevance and popularity [Gao et al., 2022].
ZINC 1M	1 M	3	3	Random molecules from $\approx$ 1.5 billion molecules from the ZINC database [Sterling and Irwin, 2015a]. ZINC 1M $\subset$ ZINC 10M $\subset$ ZINC100M.
ZINC 10M	10 M	3	4	
ZINC 100M	100 M	3	4	

All of our experiments on **pytdc** tasks are run across 5 random seeds. Since docking simulations are expensive and time consuming, we run all **docking** experiments across 3 random seeds. Experiments with different seeds use the same pretrained model which is only pretrained once for every dataset. Additional details about the task rewards, evaluation metrics, and the pretraining datasets and models are discussed in Appendix A.1, A.2, and A.3 respectively.

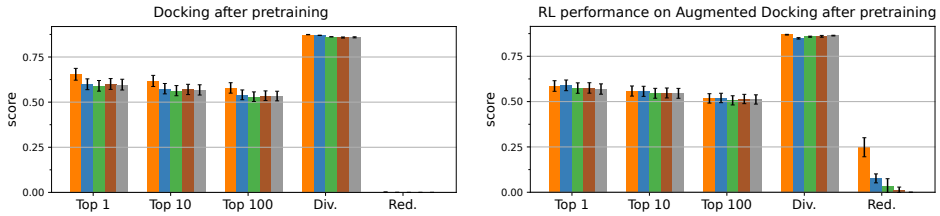
5.2 How does prior data affect the final performance of ChemRLformer?

In this section, we pretrain the REINFORCE policy on datasets of varying size and quality from Table 2. Our datasets vary from small (250K) to very large (100M) sizes. Due to the parallelizability of training on larger datasets, we use the transformer policy architecture for all experiments in this section. In natural language processing (NLP), pretraining transformer models on large and diverse unlabelled datasets have been found to perform well on downstream tasks using few-shot labeled data [Brown et al., 2020b]. Yet, our results in Figure 2b indicate that the quality of the prior dataset matters more than its size. Figure 2, shows that the distribution of the *ChEMBL* dataset is more aligned with both the **pytdc** and the **docking** tasks. As a result, the RL agent pretrained on the *ChEMBL* dataset outperforms all other agents, including the ones trained on 100 times more data.

Results may seem surprising from an NLP perspective, but they make sense when viewed from an RL perspective. Pretraining using next token prediction Equation (2), is analogous to behavior cloning in this context, where the performance depends largely on the quality of the offline dataset [Ross et al., 2011, Ho and Ermon, 2016]. These results suggest that ChemRLformer might



(a) Performance on SMILES-based molecular design with pertaining (left) and with pretrianing and RL (right).



(b) performance on SMILES-based molecular docking with pertaining (right) and with pretrainig and RL (right). Section 5.2 describes augmented docking setting with additional experiments shown in Appendix A.3.

Figure 2: On the left pretrained performance on SMILES-based ChemRLformer. Higher-quality datasets, such as ChemBL lead to higher-performance for both **pytdc** and augmented docking. On the right is the performance after RL training. RL has a substantial benefit for **pytdc** tasks, while for docking tasks an *augmented docking* score is used to avoid reward hacking, see Figure 4 for details.

benefit from better pretraining objectives, that go beyond simple imitation learning, when trained on large and diverse offline datasets [Kumar et al., 2023, Farebrother et al., 2023].

5.3 Text representations and architectures for ChemRLformer

Starting with a REINFORCE agent, we isolate the effect of various text representations for molecules and policy network architectures on performance. All experiments in this section use **ZINC-250k** dataset for pretraining. Similar results obtained for other datasets are shared in the following sections. Whenever we show normalized results across different experiments, we add the individual plots in Appendix C.1.

Text representations. In Figure 3 we compare ChemRLformer agents using different architectures and tasks across environments that base their dynamics on SELFIES and SMILES. The results show normalized scores across all architectures. Consistent with prior work [Gao et al., 2022] we find that SMILES-based polices generally outperforms SELFIES-based policies. On all **pytdc** tasks and architectures, ChemRLformer agents based on SMILES consistently achieve better rewards when compared to SELFIES-based agents across all reward metrics. Although more subtle, we observe a similar theme in the **docking** tasks where SMILES achieves higher rewards than SELFIES on all top-K metrics. Another consistent theme in the results is that even though the diversity of top-100 molecules obtained by SELFIES is higher, the redundancy of SELFIES agents is higher as well. This means that SELFIES-based ChemRLformer agents explore a much smaller region of the molecular space. These results suggest that the rules which allow SELFIES strings to always be converted into a valid molecule can actually be detrimental to the agent's exploration and search for high-value molecules, more details in Appendix C.1.

Architectures. The results in Figure 4 show that the **transformer** and **RNN** have similar performance on all tasks. On the **pytdc** tasks, **FC** achieves worse performance than other architectures specially made to handle strings, as expected. However, on **docking** tasks, **FC** obtains unusually high rewards. We find that this method performs a type of *reward function hacking* [Amodei et al., 2016, Skalse et al., 2022, Everitt, 2019] by exploiting a corner case of the docking-based reward function which provides high rewards for long strings of Carbon and Nitrogen atoms together. To evade the reward hacking of docking scores, we constructed an *augmented docking* score function with commonly used oracles (QED and SA scores) based on

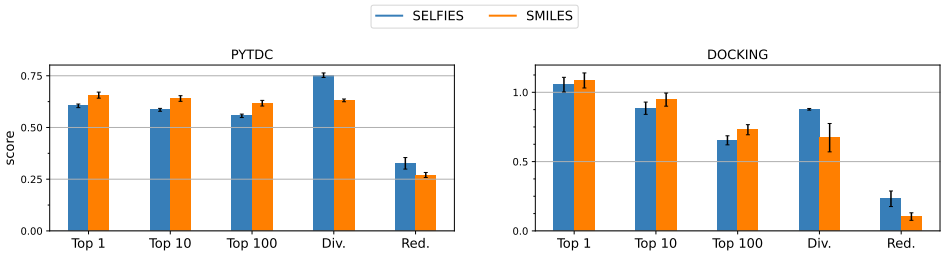


Figure 3: **Comparison between SELFIES and SMILES:** The SELFIES representation makes it relatively difficult for ChemRLformer agents to explore effectively leading to generally lower performance on pytdc and docking while scoring higher on diversity. Scores are reported for the transformer model and are averaged across all reward functions.

previous work [Lee et al., 2023] (See Appendix A.3 for more details). This finding shows that the REINFORCE agent can search the space well and, in this case, can be used to expose issues with the current design of reward functions.

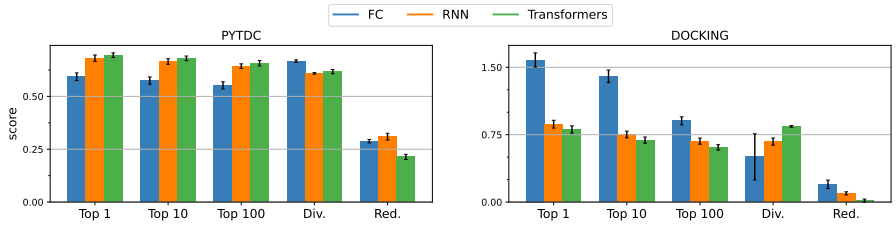


Figure 4: **Comparison of different policy architectures:** No single architecture clearly outperforms for molecular ChemRLformer. Although FC does better on the docking tasks, our analysis shows that it learns to exploit the docking function as opposed to designing high-value molecules. More details about ways to tackle this issue are given in Appendix C.3. Additional experiments for comparing transformers and RNNs are shown in Appendix C.5. These experiments use the smiles text representation.

5.4 Revisiting RL Algorithm Design Choices for ChemRLformer.

Previous experiments identify key design and algorithmic choices for efficient text-based RL. But recent text-based RL algorithms [Olivecrona et al., 2017, Bjerrum et al., 2023, Thomas et al., 2022] also employ many additional algorithmic components like replay buffers, hill climbing, KL regularisation towards the pretrained policy, and likelihood penalties. We perform an ablation study across these components to understand which ones are beneficial for the performance of ChemRLformer. All experiments in this section are performed on **pytdc** tasks, using an RNN architecture as it is most commonly used in text-based RL. See Appendix C.4 for experiments with other architectures and reward functions.

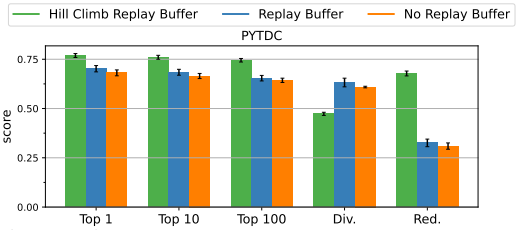


Figure 5: Hill climbing buffer lead to 13% improvement in Top-100 rewards.

Replay buffers and hill climbing. In off-policy deep RL, a replay buffer is generally used to store and reuse previous trajectories for training. Although text-based RL algorithms are trained on-policy, prior work has proposed using a replay buffer to improve performance [Mnih et al., 2013]. Standard replay buffers throw away the oldest trajectories as newer ones arrive. But many text-based RL algorithms propose to use hill-climb replay buffers, that randomly sample a batch of molecules from the highest scoring molecules seen so far and add them to the current mini batch. In Figure 5, we see that using the hill-climb buffer results in a significant performance boost for ChemRLformer, whereas using a standard buffer does not contribute much. Notably, the use of a hill-climb replay buffer reduces diversity

and increases redundancy quite substantially. The following two experiments involve combining regularisation terms with the RL objective in Equation 3. The coefficients on these extra terms can largely affect the final performance. To make a fair comparison, we perform hyper-parameter tuning over six different values for every new regularisation term with details provided in Appendix B.

Should the policy to stay close to the pretrained model? Pretrained models carry information on how to build valid drug-like molecules. To ensure that ChemRLformer agents do not stray far away from the space of valid drug-like molecules during exploration, Olivecrona et al. [2017], Gao et al. [2022] constrain the KL divergence between the policy and the pretrained model by adding a KL penalty to the policy gradient loss function in Equation (3). Prior works show that adding this penalty helps the agent achieve better sample efficiency [Gao et al., 2022]. Yet, our results in Figure 6 suggest that, when you increase the number of oracle calls in simulation, adding this penalty does not yield any additional benefit while substantially increasing the GPU memory requirement, especially when using larger models. Since invalid molecules correspond to zero rewards, the ChemRLformer agent is able to learn to avoid invalid structures on its own merit.

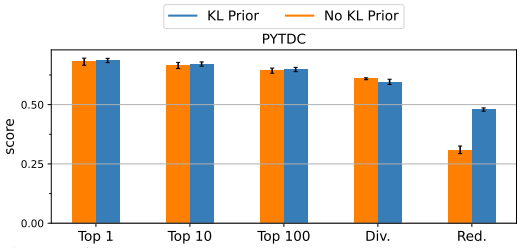


Figure 6: Our experiments show little difference in performance for multiple KL regularization terms.

Regularizing the policy's likelihood for exploration. RL agents classically face an exploration-exploitation dilemma, which can lead to agents getting stuck in sub-optimal local maxima when not well balanced. ChemRLformer agents are not immune to this dilemma. Upon encountering good, but sub-optimal molecules, an agent may adjust its policy to increase the likelihood of sampling these sub-optimal molecules and, without sufficient exploration, fail to discover higher-value regions of policy space. This can be particularly detrimental during the initial learning stages.

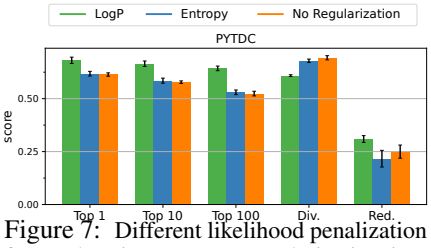


Figure 7: Different likelihood penalization for exploration. Log P regularization is a better choice for efficient exploration for ChemRLformer.

To combat this issue, entropy regularisation, which adds a $\log \pi(s)$ term to the RL loss, has been proposed [Haarnoja et al., 2018]. This encourages the RL policy to explore states with lower likelihood values. Similarly [Olivecrona et al., 2017] adds a Log p regularizer, which penalizes higher likelihood values by adding a $-1/\log \pi(s)$ term to the RL loss. In Figure 7, our results show that although an entropy regularizer leads to lesser redundancy, the Log p regularizer boosts performance significantly by exploring more efficiently. The Log p regularizer only penalizes the agent for being extremely certain (likelihood $\xrightarrow{\text{tends to}} 1$) about its actions, and is mostly agnostic for lower likelihood values. This penalty is a much better choice for ChemRLformer as it only activates when stuck in a local optimum of molecular space.

6 Conclusion and Future Work

We present ChemRLformer that resulted from our empirical study of multiple algorithmic components of text-based molecular design. For future practitioners, our method suggests the following philosophy: (1) Using SMILES is a better choice than SELFIES. (2) When collecting data for pretraining, the quality of molecules matter much more than the number of molecules. (3) Both transformer and RNN architectures achieve similar performance across all tasks using current datasets. (4) Incorporating components such as a hill-climb buffer and Log P regularization yields substantial performance improvements. Conversely, introducing KL regularization or opting for more intricate actor-critic algorithms may result in diminished performance, at the cost of more hyperparameters and memory resources. While our analysis addresses many questions, it also shows that RL agents were able to *hack* the reward functions suggesting that there is space to improve on the metrics used for molecule quality.

References

Amr Alhossary, Stephanus Daniel Handoko, Yuguang Mu, and Chee-Keong Kwoh. Fast, accurate, and reliable molecular docking with QuickVina 2. *Bioinformatics*, 31(13):2214–2216, 02 2015. ISSN 1367-4803. doi: 10.1093/bioinformatics/btv082. URL https://doi.org/10.1093/bioinformatics/btv082.

Zaccary Alperstein, Artem Cherkasov, and Jason Tyler Rolfe. All smiles variational autoencoder. *arXiv preprint arXiv:1905.13343*, 2019.

Dario Amodei, Chris Olah, Jacob Steinhardt, Paul Christiano, John Schulman, and Dan Mané. Concrete problems in ai safety. *arXiv preprint arXiv:1606.06565*, 2016.

Emmanuel Bengio, Moksh Jain, Maksym Korablyov, Doina Precup, and Yoshua Bengio. Flow network based generative models for non-iterative diverse candidate generation, 2021. URL https://arxiv.org/abs/2106.04399.

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

Mostapha Benhenda. Chemgan challenge for drug discovery: can ai reproduce natural chemical diversity?, 2017.

Esben Jannik Bjerrum and Richard Threlfall. Molecular generation with recurrent neural networks (rnns). *arXiv preprint arXiv:1705.04612*, 2017.

Esben Jannik Bjerrum, Christian Margreitter, Thomas Blaschke, and Raquel López-Ríos de Castro. Faster and more diverse de novo molecular optimization with double-loop reinforcement learning using augmented smiles, 2023.

Thomas Blaschke, Josep Arús-Pous, Hongming Chen, Christian Margreitter, Christian Tyrchan, Ola Engkvist, Kostas Papadopoulos, and Atanas Patronov. Reinvent 2.0: an ai tool for de novo drug design. *Journal of chemical information and modeling*, 60(12):5918–5922, 2020a.

Thomas Blaschke, Josep Arús-Pous, Hongming Chen, Christian Margreitter, Christian Tyrchan, Ola Engkvist, Kostas Papadopoulos, and Atanas Patronov. Reinvent 2.0: An ai tool for de novo drug design. *Journal of Chemical Information and Modeling*, 60, 10 2020b. doi: 10.1021/acs.jcim.0c00915.

Nathan Brown, Marco Fiscato, Marwin H.S. Segler, and Alain C. Vaucher. GuacaMol: Benchmarking models for de novo molecular design. *Journal of Chemical Information and Modeling*, 59(3):1096–1108, mar 2019. doi: 10.1021/acs.jcim.8b00839. URL https://doi.org/10.1021%2Facs.jcim.8b00839.

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel Ziegler, Jeffrey Wu, Clemens Winter, Chris Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. Language Models are Few-Shot Learners. In H Larochelle, M Ranzato, R Hadsell, M F Balcan, and H Lin, editors, *Advances in Neural Information Processing Systems*, volume 33, pages 1877–1901. Curran Associates, Inc., 2020a. URL https://proceedings.neurips.cc/paper_files/paper/2020/file/1457c0d6bfcb4967418bfb8ac142f64a-Paper.pdf.

Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. Language models are few-shot learners, 2020b.

Tianjian Chen, Zhanpeng He, and Matei Ciocarlie. Hardware as policy: Mechanical and computational Co-Optimization using deep reinforcement learning. In Jens Kober, Fabio Ramos, and Claire Tomlin, editors, *Proceedings of the 2020 Conference on Robot Learning*, volume 155 of *Proceedings of Machine Learning Research*, pages 1158–1173. PMLR, 2021a.

Tianlong Chen, Kaixiong Zhou, Keyu Duan, Wenqing Zheng, Peihao Wang, Xia Hu, and Zhangyang Wang. Bag of tricks for training deeper graph neural networks: A comprehensive benchmark study. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2022.

Xiaocong Chen, Lina Yao, Julian McAuley, Guanglin Zhou, and Xianzhi Wang. A survey of deep reinforcement learning in recommender systems: A systematic review and future directions. *arXiv preprint arXiv:2109.03540*, 2021b.

Austin H Cheng, Andy Cai, Santiago Miret, Gustavo Malkomes, Mariano Phielipp, and Alán Aspuru-Guzik. Group selfies: a robust fragment-based molecular string representation. *Digital Discovery*, 2023.

Tobiasz Cieplinski, Tomasz Danel, Sabina Podlewska, and Stanislaw Jastrzebski. We should at least be able to design molecules that dock well, 2021.

Vijay Prakash Dwivedi, Chaitanya K. Joshi, Anh Tuan Luu, Thomas Laurent, Yoshua Bengio, and Xavier Bresson. Benchmarking graph neural networks, 2022.

Carl Edwards, Tuan Lai, Kevin Ros, Garrett Honke, Kyunghyun Cho, and Heng Ji. Translation between molecules and natural language. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pages 375–413, Abu Dhabi, United Arab Emirates, December 2022a. Association for Computational Linguistics. URL `https://aclanthology.org/2022.emnlp-main.26`.

Carl Edwards, Tuan Lai, Kevin Ros, Garrett Honke, Kyunghyun Cho, and Heng Ji. Translation between molecules and natural language, 2022b.

Tom Everitt. *Towards safe artificial general intelligence*. PhD thesis, The Australian National University (Australia), 2019.

Benedek Fabian, Thomas Edlich, Héléna Gaspar, Marwin Segler, Joshua Meyers, Marco Fiscato, and Mohamed Ahmed. Molecular representation learning with language models and domain-relevant auxiliary tasks. *arXiv preprint arXiv:2011.13230*, 2020.

Jesse Farebrother, Joshua Greaves, Rishabh Agarwal, Charline Le Lan, Ross Goroshin, Pablo Samuel Castro, and Marc G. Bellemare. Proto-value networks: Scaling representation learning with auxiliary tasks, 2023.

Daniel Flam-Shepherd, Kevin Zhu, and Alán Aspuru-Guzik. Language models can learn complex molecular distributions. *Nature Communications*, 13(1):3293, 2022.

Tianfan Fu, Cao Xiao, Lucas M. Glass, and Jimeng Sun. Moler: Incorporate molecule-level reward to enhance deep generative model for molecule optimization. *IEEE Transactions on Knowledge and Data Engineering*, 34(11):5459–5471, 2022. doi: 10.1109/TKDE.2021.3052150.

Wenhao Gao, Tianfan Fu, Jimeng Sun, and Connor W. Coley. Sample efficiency matters: A benchmark for practical molecular optimization, 2022.

Miguel García-Ortegón, Gregor NC Simm, Austin J Tripp, José Miguel Hernández-Lobato, Andreas Bender, and Sergio Bacallado. Dockstring: easy molecular docking yields better benchmarks for ligand design. *Journal of chemical information and modeling*, 62(15):3486–3502, 2022.

Anna Gaulton, Louisa J. Bellis, A. Patricia Bento, Jon Chambers, Mark Davies, Anne Hersey, Yvonne Light, Shaun McGlinchey, David Michalovich, Bissan Al-Lazikani, and John P. Overington. Chembl: A large-scale bioactivity database for drug discovery. *Nucleic Acids Research*, 40(D1):D1100–D1107, January 2012. ISSN 0305-1048. doi: 10.1093/nar/gkr777.

Simon Geisler, Tobias Schmidt, Hakan Şirin, Daniel Zügner, Aleksandar Bojchevski, and Stephan Günnemann. Robustness of graph neural networks at scale, 2023.

Rafael Gó mez-Bombarelli, Jennifer N. Wei, David Duvenaud, José Miguel Hernández-Lobato, Benjamín Sánchez-Lengeling, Dennis Sheberla, Jorge Aguilera-Iparraguirre, Timothy D. Hirzel, Ryan P. Adams, and Alán Aspuru-Guzik. Automatic chemical design using a data-driven continuous representation of molecules. *ACS Central Science*, 4(2):268–276, jan 2018. doi: 10.1021/acscentsci.7b00572. URL https://doi.org/10.1021%2Facscentsci.7b00572.

Rafael Gómez-Bombarelli, Jennifer N Wei, David Duvenaud, José Miguel Hernández-Lobato, Benjamín Sánchez-Lengeling, Dennis Sheberla, Jorge Aguilera-Iparraguirre, Timothy D Hirzel, Ryan P Adams, and Alán Aspuru-Guzik. Automatic chemical design using a data-driven continuous representation of molecules. *ACS central science*, 4(2):268–276, 2018.

Francesca Grisoni, Michael Moret, Robin Lingwood, and Gisbert Schneider. Bidirectional molecule generation with recurrent neural networks. *Journal of chemical information and modeling*, 60(3): 1175–1183, 2020.

Anvita Gupta, Alex T Müller, Berend JH Huisman, Jens A Fuchs, Petra Schneider, and Gisbert Schneider. Generative recurrent networks for de novo drug design. *Molecular informatics*, 37 (1-2):1700111, 2018.

David Ha. Reinforcement learning for improving agent design. *Artif. Life*, 25(4):352–365, November 2019.

Tuomas Haarnoja, Aurick Zhou, Pieter Abbeel, and Sergey Levine. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor, 2018.

Stephen Heller, Alan McNaught, Stephen Stein, Dmitrii Tchekhovskoi, and Igor Pletnev. Inchi-the worldwide chemical structure identifier standard. *Journal of cheminformatics*, 5(1):1–9, 2013.

Jonathan Ho and Stefano Ermon. Generative adversarial imitation learning, 2016.

Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural Comput.*, 9(8): 1735–1780, nov 1997. ISSN 0899-7667. doi: 10.1162/neco.1997.9.8.1735. URL https://doi.org/10.1162/neco.1997.9.8.1735.

Kexin Huang, Tianfan Fu, Wenhao Gao, Yue Zhao, Yusuf Roohani, Jure Leskovec, Connor W. Coley, Cao Xiao, Jimeng Sun, and Marinka Zitnik. Therapeutics data commons: Machine learning datasets and tasks for drug discovery and development, 2021.

Kexin Huang, Tianfan Fu, Wenhao Gao, Yue Zhao, Yusuf Roohani, Jure Leskovec, Connor W Coley, Cao Xiao, Jimeng Sun, and Marinka Zitnik. Artificial intelligence foundation for therapeutic science. *Nature Chemical Biology*, 18(10):1033–1036, 2022.

John J Irwin, Teague Sterling, Michael M Mysinger, Erin S Bolstad, and Ryan G Coleman. Zinc: a free tool to discover chemistry for biology. *Journal of chemical information and modeling*, 52 (7):1757–1768, 2012.

Ross Irwin, Spyridon Dimitriadis, Jiazhen He, and Esben Jannik Bjerrum. Chemformer: a pre-trained transformer for computational chemistry. *Machine Learning: Science and Technology*, 3 (1):015022, 2022.

Wengong Jin, Regina Barzilay, and Tommi Jaakkola. Junction tree variational autoencoder for molecular graph generation, 2018. URL https://arxiv.org/abs/1802.04364.

Wengong Jin, Regina Barzilay, and Tommi Jaakkola. Multi-objective molecule generation using interpretable substructures, 2020.

Jared Kaplan, Sam McCandlish, Tom Henighan, Tom B. Brown, Benjamin Chess, Rewon Child, Scott Gray, Alec Radford, Jeffrey Wu, and Dario Amodei. Scaling laws for neural language models, 2020.

Shauharda Khadka, Estelle Aflalo, Mattias Marder, Avrech Ben-David, Santiago Miret, Shie Mannor, Tamir Hazan, Hanlin Tang, and Somdeb Majumdar. Optimizing memory placement using evolutionary graph reinforcement learning. *arXiv preprint arXiv:2007.07298*, 2020.

Mario Krenn, Florian Häse, AkshatKumar Nigam, Pascal Friederich, and Alan Aspuru-Guzik. Self-referencing embedded strings (SELFIES): A 100% robust molecular string representation. *Machine Learning: Science and Technology*, 1(4):045024, oct 2020. doi: 10.1088/2632-2153/aba947. URL https://doi.org/10.1088%2F2632-2153%2Faba947.

Mario Krenn, Qianxiang Ai, Senja Barthel, Nessa Carson, Angelo Frei, Nathan C Frey, Pascal Friederich, Théophile Gaudin, Alberto Alexander Gayle, Kevin Maik Jablonka, et al. Selfies and the future of molecular string representations. *Patterns*, 3(10):100588, 2022.

Aviral Kumar, Amir Yazdanbakhsh, Milad Hashemi, Kevin Swersky, and Sergey Levine. Data-driven offline optimization for architecting hardware accelerators, 2022.

Aviral Kumar, Rishabh Agarwal, Xinyang Geng, George Tucker, and Sergey Levine. Offline q-learning on diverse multi-task data both scales and generalizes, 2023.

Greg Landrum et al. Rdkit: A software suite for cheminformatics, computational chemistry, and predictive modeling. *Greg Landrum*, 8, 2013.

Seul Lee, Jaehyeong Jo, and Sung Ju Hwang. Exploring chemical space with score-based out-of-distribution generation, 2023.

Ilya Loshchilov and Frank Hutter. Sgdr: Stochastic gradient descent with warm restarts, 2017.

Kevin Sebastian Luck, Heni Ben Amor, and Roberto Calandra. Data-efficient Co-Adaptation of morphology and behaviour with deep reinforcement learning. In Leslie Pack Kaelbling, Danica Kragic, and Komei Sugiura, editors, *Proceedings of the Conference on Robot Learning*, volume 100 of *Proceedings of Machine Learning Research*, pages 854–869. PMLR, 2020.

David Mendez, Anna Gaulton, A Patrícia Bento, Jon Chambers, Marleen De Veij, Eloy Félix, María Paula Magariños, Juan F Mosquera, Prudence Mutowo, Michał Nowotka, et al. Chembl: towards direct deposition of bioassay data. *Nucleic acids research*, 47(D1):D930–D940, 2019.

Santiago Miret, Vui Seng Chua, Mattias Marder, Mariano Phiellip, Nilesh Jain, and Somdeb Majumdar. Neuroevolution-enhanced multi-objective optimization for mixed-precision quantization. In *Proceedings of the Genetic and Evolutionary Computation Conference*, pages 1057–1065, 2022.

Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou, Daan Wierstra, and Martin Riedmiller. Playing atari with deep reinforcement learning, 2013.

Henry B. Moss, Daniel Beck, Javier Gonzalez, David S. Leslie, and Paul Rayson. Boss: Bayesian optimization over string spaces, 2020.

AkshatKumar Nigam, Robert Pollice, and Alan Aspuru-Guzik. Janus: Parallel tempered genetic algorithm guided by deep neural networks for inverse molecular design, 2021.

Marcus Olivecrona, Thomas Blaschke, Ola Engkvist, and Hongming Chen. Molecular de novo design through deep reinforcement learning, 2017.

Emilio Parisotto, H. Francis Song, Jack W. Rae, Razvan Pascanu, Caglar Gulcehre, Siddhant M. Jayakumar, Max Jaderberg, Raphael Lopez Kaufman, Aidan Clark, Seb Noury, Matthew M. Botvinick, Nicolas Heess, and Raia Hadsell. Stabilizing transformers for reinforcement learning, 2019.

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Köpf, Edward Yang, Zach DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. Pytorch: An imperative style, high-performance deep learning library, 2019.

Alec Radford, Karthik Narasimhan, Tim Salimans, and Ilya Sutskever. Improving language understanding by generative pre-training. 2018.

Stephane Ross, Geoffrey J. Gordon, and J. Andrew Bagnell. A reduction of imitation learning and structured prediction to no-regret online learning, 2011.

David E. Rumelhart and James L. McClelland. *Learning Internal Representations by Error Propagation*, pages 318–362. MIT Press, 1987.

Charles Schaff, David Yunis, Ayan Chakrabarti, and Matthew R Walter. Jointly learning to construct and control agents using deep reinforcement learning. In *2019 International Conference on Robotics and Automation (ICRA)*, pages 9798–9805. ieeexplore.ieee.org, May 2019.

Robin M Schmidt. Recurrent neural networks (rnns): A gentle introduction and overview. *arXiv preprint arXiv:1912.05911*, 2019.

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms, 2017.

Cynthia Shen, Mario Krenn, Sagi Eppel, and Alá n Aspuru-Guzik. Deep molecular dreaming: inverse machine learning for de-novo molecular design and interpretability with surjective representations. *Machine Learning: Science and Technology*, 2(3):03LT02, jul 2021. doi: 10.1088/2632-2153/ac09d6. URL https://doi.org/10.1088%2F2632-2153%2Fac09d6.

Joar Skalse, Nikolaus HR Howe, Dmitrii Krasheninnikov, and David Krueger. Defining and characterizing reward hacking. *arXiv preprint arXiv:2209.13085*, 2022.

Teague Sterling and John Irwin. Zinc 15 - ligand discovery for everyone. *Journal of chemical information and modeling*, 55, 10 2015a. doi: 10.1021/acs.jcim.5b00559.

Teague Sterling and John J Irwin. Zinc 15–ligand discovery for everyone. *Journal of chemical information and modeling*, 55(11):2324–2337, 2015b.

Richard S Sutton, David McAllester, Satinder Singh, and Yishay Mansour. Policy gradient methods for reinforcement learning with function approximation. In S. Solla, T. Leen, and K. Müller, editors, *Advances in Neural Information Processing Systems*, volume 12. MIT Press, 1999a. URL https://proceedings.neurips.cc/paper_files/paper/1999/file/464d828b85b0bed98e80ade0a5c43b0f-Paper.pdf.

Richard S. Sutton, David McAllester, Satinder Singh, and Yishay Mansour. Policy gradient methods for reinforcement learning with function approximation. In *Proceedings of the 12th International Conference on Neural Information Processing Systems*, NIPS'99, page 1057–1063, Cambridge, MA, USA, 1999b. MIT Press.

Qiaoyu Tan, Ninghao Liu, and Xia Hu. Deep representation learning for social network analysis. *Frontiers in big Data*, 2:2, 2019.

Ross Taylor, Marcin Kardas, Guillem Cucurull, Thomas Scialom, Anthony Hartshorn, Elvis Saravia, Andrew Poulton, Viktor Kerkez, and Robert Stojnic. Galactica: A large language model for science. *arXiv preprint arXiv:2211.09085*, 2022.

Morgan Thomas, Noel O'Boyle, Andreas Bender, and Chris Graaf. Augmented hill-climb increases reinforcement learning efficiency for language-based de novo molecule generation. *Journal of Cheminformatics*, 14, 10 2022. doi: 10.1186/s13321-022-00646-z.

Austin Tripp, Wenlin Chen, and José Miguel Hernández-Lobato. An evaluation framework for the objective functions of de novo drug design benchmarks. In *ICLR2022 Machine Learning for Drug Discovery*, 2022. URL https://openreview.net/forum?id=W1tcNQNG1S.

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

Qi Wang and Chunlei Tang. Deep reinforcement learning for transportation network combinatorial optimization: A survey. *Knowledge-Based Systems*, 233:107526, 2021.

Sheng Wang, Yuzhi Guo, Yuhong Wang, Hongmao Sun, and Junzhou Huang. Smiles-bert: large scale unsupervised pre-training for molecular property prediction. In *Proceedings of the 10th ACM international conference on bioinformatics, computational biology and health informatics*, pages 429–436, 2019.

David Weininger. Smiles, a chemical language and information system. 1. introduction to methodology and encoding rules. *J. Chem. Inf. Comput. Sci.*, 28(1):31–36, feb 1988. ISSN 0095-2338. doi: 10.1021/ci00057a005. URL https://doi.org/10.1021/ci00057a005.

Soojung Yang, Doyeong Hwang, Seul Lee, Seongok Ryu, and Sung Ju Hwang. Hit and lead discovery with explorative rl and fragment-based molecule generation, 2021.

Denis Yarats, Rob Fergus, Alessandro Lazaric, and Lerrel Pinto. Mastering visual continuous control: Improved data-augmented reinforcement learning, 2021.

Jiaxuan You, Bowen Liu, Rex Ying, Vijay Pande, and Jure Leskovec. Graph convolutional policy network for goal-directed molecular graph generation, 2018. URL https://arxiv.org/abs/1806.02473.

Wenhao Yu, C. Karen Liu, and Greg Turk. Policy transfer with strategy optimization, 2018.

Zhenni Zeng, Yuan Yao, Zhiyuan Liu, and Maosong Sun. A deep-learning system bridging molecule structure and biomedical text with comprehension comparable to human professionals. *Nature communications*, 13(862), 2022.

Zhenpeng Zhou, Steven Kearnes, Li Li, Richard N. Zare, and Patrick Riley. Optimization of molecules via deep reinforcement learning. *Scientific Reports*, 9(1), jul 2019. doi: 10.1038/s41598-019-47148-x. URL https://doi.org/10.1038%2Fs41598-019-47148-x.

Outline of Appendices. In Appendix A we provide details about the experimental setup. In Appendix B we describe our hyperparameter tuning strategy. In Appendix C we include additional results from our experiments.

A Experimental setup

In this section, we provide additional details about the tasks, evaluation metrics and pretraining models and data used in our work.

A.1 Tasks

Pydtc tasks. These tasks are a set of 21 pharmaceutically-relevant oracle functions, which have been commonly used in prior work [Brown et al., 2019, Gao et al., 2022, Huang et al., 2021] for evaluating performance across molecular discovery algorithms:

- QED: A quantitative estimate of drug-likeness calculated using a set of rules.
- DRD2, GSK3β, and JNK3: Classical machine learning models (SVMs and random forests) that provide an estimate of properties like target affinity or susceptibility towards a disorder.
- Celecoxib, Troglitazone, and Thiothixene rediscovery: An estimate of smiles text similarity, based on tanimoto metric, towards a target molecule.
- Albuterol and Mestranol similarity: Generate molecules similar to a target molecule.
- Isomers_c7h8n2o2 and isomers_c9h10n2o2pf2cl: Generate molecules corresponding to a target molecular formula.
- Median1 and Median2: Generate molecules that are maximally similar to several target molecules.
- Osimertinib_mpo, fexofenadine_mpo, ranolazine_mpo, perindopril_mpo, amlodipine_mpo, sitagliptin_mpo, zaleplon_mpo: Generate molecules that maximize multiple properties of a targeted drug.
- valsartan_smarts: Generate molecules that contain a certain SMARTS pattern and certain physicochemical properties.

Most of these tasks are from the GuacaMol benchmark [Brown et al., 2019]. All oracles are calculated using the Python API provided by Therapeutics Data Commons [Huang et al., 2021] and more details for these tasks can be found on their website.

Docking tasks. We used QuickVina 2 [Alhossary et al., 2015] for calculating docking scores using the same default configuration parameters as prior works [Yang et al., 2021, Lee et al., 2023]. For example, we used exhaustiveness $= 1$, and modes $= 10$. We choose 5 different protein targets to calculate docking scores: fa7 (FA7), parp1 (PARP-1), 5ht1b (5-HT1B), jak2 (JAK-2), and braf (BRAF). These targets were chosen by [Yang et al., 2021, Lee et al., 2023] because the docking simulators for these targets work fairly well when compared to the ground truth. In our experiments in Section 5.3, we found that text-based RL algorithms were easily able to produce chemically trivial molecules that have very high docking scores. To understand the complexity of computing docking scores, we report the time taken to dock 1000 molecules in parallel using 12 CPUs table 3. We also provide the time taken to run the RL algorithm on these 1000 molecules after their docking scores are available.

Table 3: **Time complexity of docking score evaluation** : More than half the running time is spent evaluating the docking scores.

Number of molecules	Docking time	RL update time
1000	130 seconds	74 seconds

Augmented docking tasks. In our results for the standard **docking** tasks (Figure 3 and Figure 4), we found that using the simulated docking scores as rewards did not lead to chemically relevant molecules. Text-based RL algorithms were able to exploit their state and action spaces to design chemically trivial molecules that have very high docking scores. To tackle this issue of undesirable reward hacking, we tried a reward function based on prior works [García-Ortegón et al., 2022, Lee et al., 2023] that combine objectives for drug-like, and synthesizable molecules with docking scores. We call tasks corresponding to this new reward function as **augmented docking** tasks. Concretely, we chose the same reward function from [Lee et al., 2023]

$$r(s) = -\text{DS}(s)/20 \times \text{QED}(s) \times (10 - \text{SA}(s))/9, \tag{4}$$

Where DS is the docking score, QED and SA are quantitative estimates of drug likeness and synthesizablity respectively.

A.2 Evaluation metrics

Most of the metrics we use are described in detail in Section 5.1. Here, we provide additional details about the diversity metric. We calculate the diversity of the top 100 molecules sampled by the algorithm, where higher diversity is considered better given that it increases the chances for success in further wet lab experiments. In our experiments, we use the diversity evaluator from TDC [Huang et al., 2021], which defines the diversity of a set of molecules as the average pairwise Tanimoto similarity between Morgan fingerprints of the molecules. See Section 2 of [Benhenda, 2017] for exact details of how Tanimoto similarity is calculated.

A.3 Pretraining

In this section, we provide more details about the pretraining datasets and models used in our experiments.

Pretraining datasets. The ZINC 250k dataset contains approximately 250k molecules from the ZINC database [Irwin et al., 2012], chosen for their pharmaceutical relevance, moderate size, and popularity [Gao et al., 2022]. The CHEMBL dataset [Mendez et al., 2019] consists of approximately 2M manually curated drug-like molecules. The other 3 datasets consist of randomly selected subsets of the ZINC-15 dataset [Sterling and Irwin, 2015b] that obey some chemically imposed mild constraints [Irwin et al., 2022]. We test three subsets of different sizes: (1) ZINC 1M (2) ZINC 10M, and (3) ZINC 100M, to test the impact of scaling the size of pre-training data. These datasets and data-subsets, including their vocabularies, will be shared in an easily accessible format upon acceptance.

Removing outliers and unusual non drug-like compounds helps to keep the vocabulary small and improves the quality of the generative model [Blaschke et al., 2020a]. To achieve this, we filter all datasets by removing molecules which contain 1) less than 10 or more than 50 heavy atoms and 2) molecules *other than* Carbon, Nitrogen, Oxygen, Fluorine, Silicon, Chlorine and Bromine. We also canonicalize and sanitize all molecules using RDKIT [Landrum et al., 2013]. For experiments that apply SELFIES, we convert all datasets to SELFIES using the Python API provided by [Krenn et al., 2020] (Version: 2.1.1).

Apart from the experiments shown in the main paper, Appendix C.2 contains additional experiments comparing text-based RL agents across different pretraining datasets.

Pretraining models. In Table 4 we provide details about the pretraining modes which we use in our experiments. Upon acceptance, we will open-source our code and release the pretrained weights to support reproducible research.

We select network sizes that have been commonly used in RL [Blaschke et al., 2020a, Yarats et al., 2021]. Although conducting a study of scaling the model size [Kaplan et al., 2020] is out of the scope of our work, we believe that it is a promising direction for future.

Since the fully connected model can only take fixed length inputs, we always input a molecular text padded to a certain maximum length (we used length 100 in our experiments). This padding is done

Table 4: **Description of model architectures used for pretraining**

Model	Number of Parameters	Description
FC	1.07×10^7	FC is a fully connected neural network with 3 hidden layers of 1024 size each.
RNN	4.17×10^6	RNN is a recurrent network which consists of 3 GRU layers of hidden sizes 512 each.
TRANSFORMER	4.78×10^6	GPT [Brown et al., 2020b] style transformer with 6 layers, 16 heads and 256 embedding dimensions.

using a special token [PAD] to convey that corresponding tokens should not be considered while deciding the value of the text.

Pretraining experimental details. We pretrain FC, RNN and transformer architectures on the ZINC 250K dataset and pretrain a transformer on all other datasets. All models are pretrained using the PyTorch [Paszke et al., 2019] framework. All models used an initial learning rate of $1e-3$, with a cosine learning rate schedule [Loshchilov and Hutter, 2017]. FC and RNNs used a batch size of 128 and were trained for 10 epochs. All transformers were trained for 5 epochs, with the largest batch size that we could fit in the memory of a single NVIDIA RTX A6000 GPU, for example, a batch size of 2048 for pretraining the transformer on ZINC 100M dataset. We made sure that all models were trained until convergence. On the ZINC 250K SMILES dataset, the FC, the RNN and the transformer model achieved a validation loss of 29.417, 22.507, and 22.923 respectively.

A.4 RL finetuning

The pretrained model is further trained using the policy gradient algorithm, REINFORCE [Sutton et al., 1999b]. Given the reward function $r(s_H)$ corresponding to the text s_T, this algorithm optimizes the loss function

$$\min_{\theta} J(\theta) = -\left[\sum_{t=1}^{H} \log p_\theta(a_t = A_t \mid A_{t-1}, \cdots A_0) r(s_H = [A_0, \cdots A_H])\right], \tag{5}$$

where A_t is the token sampled by the agent at time-step t.

B Hyperparameter tuning

We conduct a common hyperparameter tuning strategy for all experiments. Specifically, we conduct hyperparameter tuning for

- Learning rate for different architectures Figure 4 and text grammars Figure 3.
- Coefficients for different likelihood regularizations Figure 7.
- Coefficients for KL regularization loss term Figure 6.

We select three tasks from the **pytdc** tasks, i.e., troglitazone_rediscovery, sitagliptin_mpo, and median2 for hyperparameter tuning. For each hyperparameter, we select a set of 5 evenly spaced realistic values and run 5 random seeds of RL experiments per hyperparameter value. We select the hyperparameter value that achieves the best average score of the top-100 molecules as the final value for running all the experiments. We report the hyperparamters used for the policy gradient training in table 5.

C Results

C.1 Text representations and architectures for RL

Here, we present additional results from subsection 5.3. Figure 8 shows that SMILES are a better molecular grammar when compared to SELFIES across all architectures, for the text based RL

Table 5: **Hyperparamters**

Name	Value
Maximum number of unique molecules	25000
Learning rate	5.00×10^{-4} RNN and FC 1.00×10^{-4} Transformer
Batch size	64
Log p coefficient	5
KL coefficient	1.00×10^{-3}

algorithms that we consider. Figure 9 compares various architectures, while keeping the molecular grammar fixed to SELFIES. The results in Figure 9 reflect our findings in Figure 4 that no single architecture clearly outperforms for molecular text-based RL. It also shows the reward hacking behavior of the **docking** tasks by the FC based RL agent.

(a) PYTDC.

(b) DOCKING.

Figure 8: Comparison between SELFIES and SMILES across different architectures. These figures are the individual plots corresponding to the normalised plot show in Figure 3.

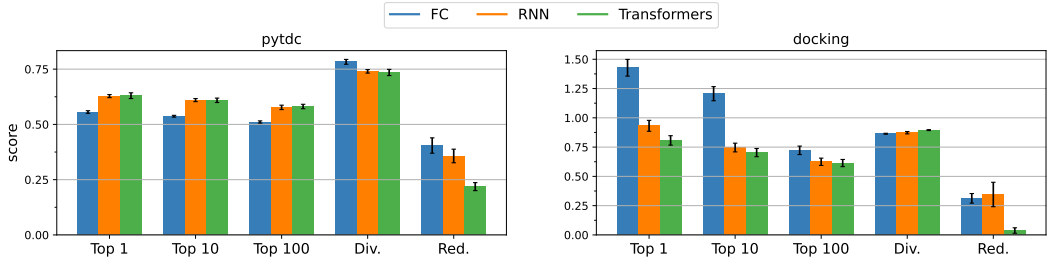


Figure 9: **Comparison of different policy architectures (SELFIES):** No single architecture clearly outperforms for molecular text-based RL. Although FC does better on the docking tasks, our analysis shows that it learns to exploit the docking function as opposed to designing high-value molecules.

The reason for lower value molecules for SELFIES environments can be explained by the SELFIES grammar that induces a flat optimization landscape. Many SELFIES strings can correspond to the

same molecule, and in fact, once an *invalid* action is taken, any subsequent sequence of tokens will be ignored on the resulting molecule. This makes exploration of new molecules difficult [Krenn et al., 2020, Gao et al., 2022]. On the other hand, the benefit of SELFIES over SMILES in eliminating invalid molecule generation is mitigated by our pretraining process, which initializes SMILES-based policies with a strong bias toward generating valid molecules. Overall, we find that SMILES-based policies, when combined with pretraining, are more effective at exploring and finding high-value molecules.

C.2 Pretraining for RL

Figure 2 (right) shows the top docking scores obtained by RL agents pre-trained on different datasets when trained with on the **augmented docking** tasks. In Figure 10, we show the actual augmented rewards obtained by the RL agent. These results suggest that the augmented docking score is a complex reward function as the RL agent is achieved minimal improvement over the prior agent. To verify this hypothesis, we increased the molecule budget of the RL agent by 10 times. We indeed see that RL agents corresponding to all prior-datasets exhibit considerable improvement. Text-based RL algorithms learn to search more efficiently when provided with more compute.

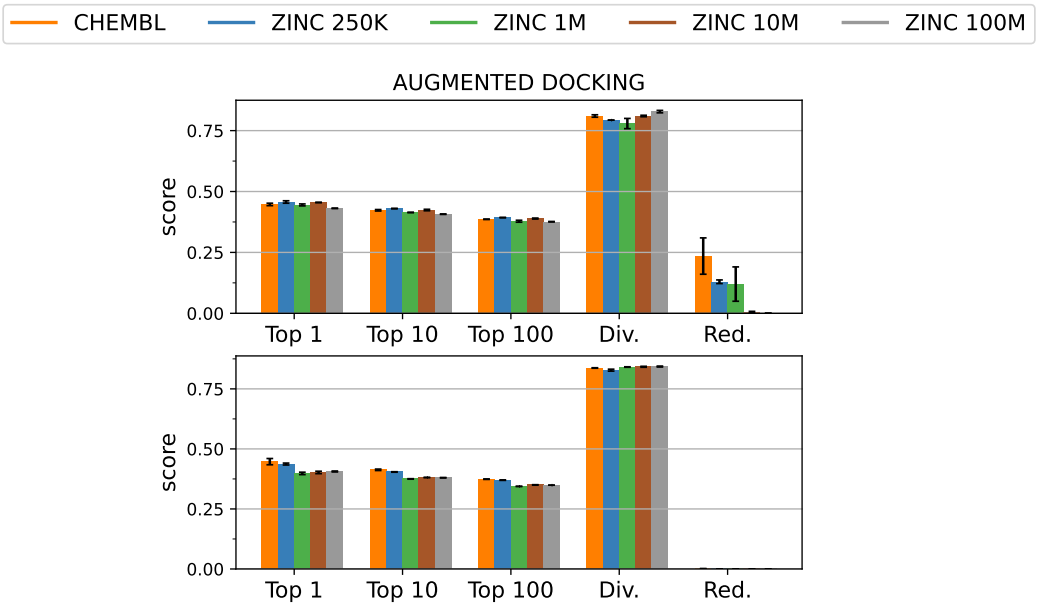


Figure 10: This figure shows the augmented rewards obtained by the RL agents (Top) and data quality (Bottom) of different datasets. See subsection A.1 for how the augmented reward is calculated.

C.3 Reward Hacking

Figure 4 and Figure 9 show that text-based RL agents that are trained using fully connected neural networks are able to obtain unusually high rewards. This is probably because it is easier for FC agents to find actions that exploit the local structure of the reward function as RNNs and Transformers are inductively biased to find global solutions. This highlights an undesirable type of reward function hacking by the FC agent which provides high rewards for molecules with long strings of Carbon and Nitrogen atoms together. Similar to prior work [Lee et al., 2023], we augment the docking scores with objectives for drug-like and synthesizable molecules. See Appendix A for details of this task and Figure 2 and Figure 10 for results corresponding to this task. Our initial results on this task (Figure 2 and Figure 10) suggested that the augmented reward function was more aligned towards chemically relevant molecules. We also noticed that the RL agents were not able to improve a lot over the prior baselines for this task. To verify whether the low performance of RL agents was because of less training data or the augmented reward function was indeed a more realistic and robust reward function, we repeated the experiments in Figure 10 with a ten times

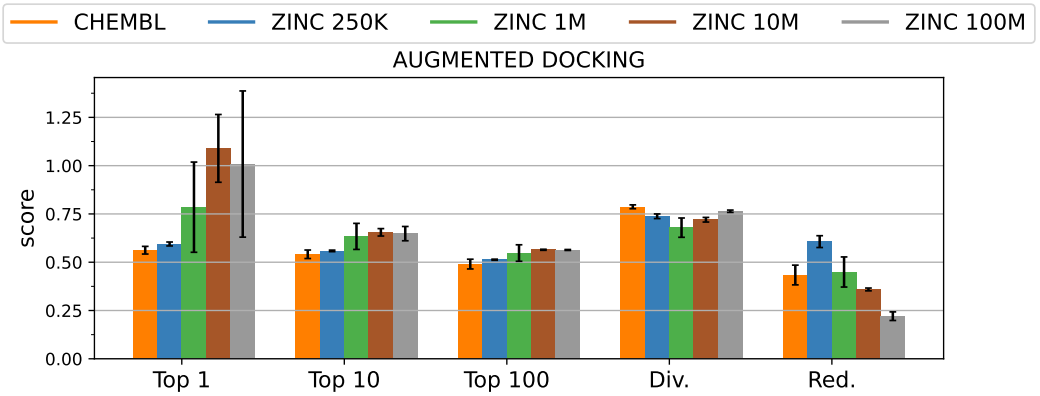


Figure 11: This figure shows the augmented rewards obtained by the RL agents trained for 10 times more molecules.

higher training budget. Given more data, all RL agents showed considerable improvements over the priors. This experiment also revealed that the agents pre-trained on ZINC 1M, ZINC 10M and ZINC 100M, were able to exploit the reward function to generate unrealistic yet highly rewarding molecules. These molecules have unusually low docking scores (less than -20). Our results highlight the need for an aligned and a more robust reward function to generate molecules for docking protein targets.

C.4 Additional results for the importance of algorithmic choices for text-based RL.

Section 5.4 compares various algorithmic components like replay buffers, hill climbing, KL regularisation towards the pretrained policy, and likelihood penalties and show results for **PYTDC** tasks. In this section, we repeat all the experiments from Section 5.4 on **augmented docking** tasks as well and reach the same conclusions. In Figure 14 we see that using the hill-climb buffer results in a significant performance boost, whereas using a standard buffer does not contribute much. Figure 15 shows that Log P regularization is a better choice for efficient exploration when it comes to text-based RL algorithms. In Figure 16 show that penalising the policy to move away from the pretrained policy does not improve performance.

C.5 Instability of transformers for online RL.

Many works [Parisotto et al., 2019] have pointed out the instability of training transformers using online reinforcement learning. To understand this in the context of text based RL, we compare a transformer and an RNN based agent on the augmented docking task. To probe whether pronounced effects of this instability are seen, we train both agents for 10 times more molecules (250K molecules). In figure Figure 12, we see that both agents perform comparably across all docking targets.

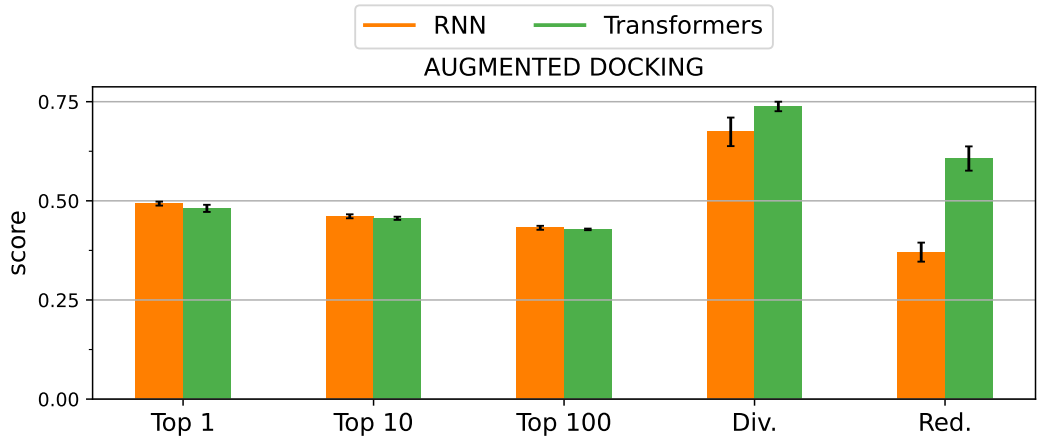


Figure 12: A transformer and an RNN based RL agent trained for 10 times more molecules on augmented docking scores.

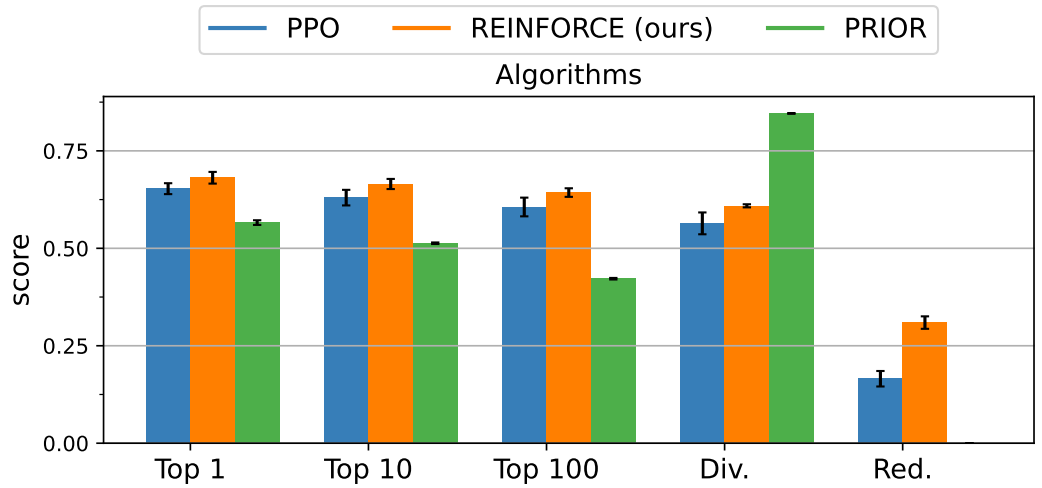


Figure 13: **Comparison with another RL algorithm PPO.** In control tasks complex algorithms like PPO [Schulman et al., 2017] are known to outperform the vanilla policy gradient algorithm. But on the molecular optimization tasks of PyTDC, our results indicate that vanilla policy gradient algorithms are more stable than actor critic algorithms like PPO and achieve higher performance. This resonates with the findings of previous work in molecular optimization [Cieplinski et al., 2021, Gao et al., 2022].

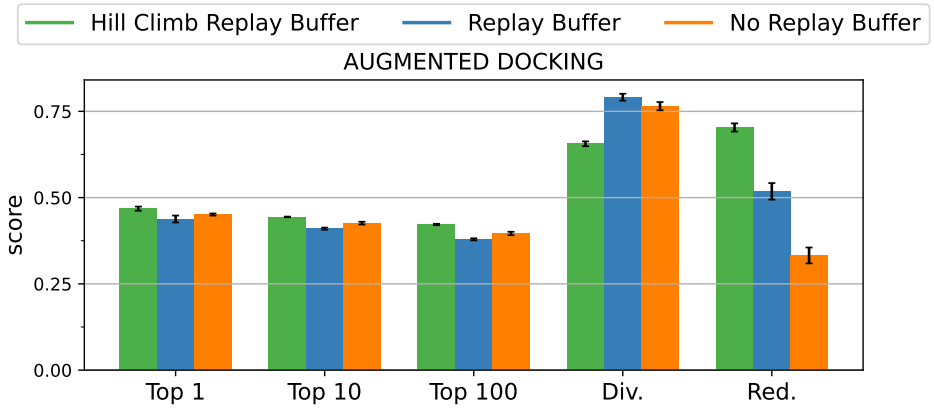


Figure 14: **Do replay buffers help?**

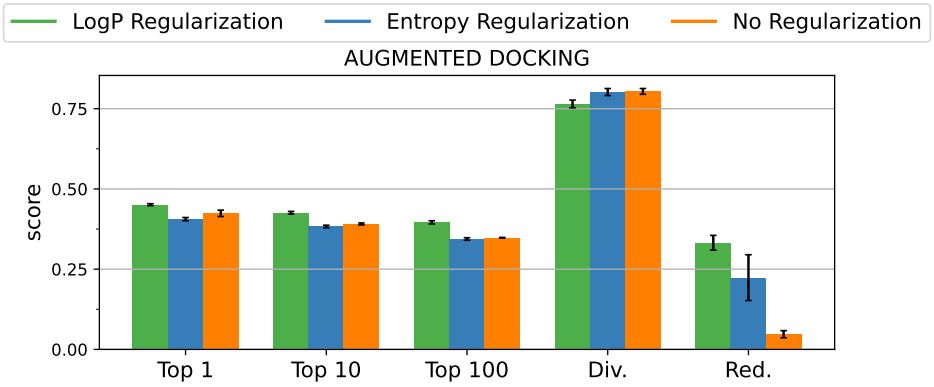


Figure 15: **Comparison of different likelihood penalization for efficient exploration**

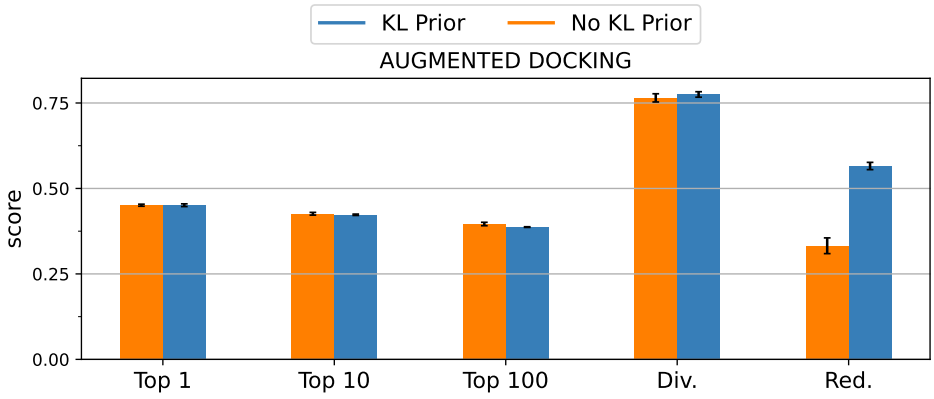


Figure 16: **Is KL regularisation with a prior necessary?**