Reinforcement Learning for Molecular Design Guided by Quantum Mechanics

Gregor N. C. Simm [*] [1] **Robert Pinsler** [*] [1] **José Miguel Hernández-Lobato** [1]

Abstract

Automating molecular design using deep reinforcement learning (RL) holds the promise of accelerating the discovery of new chemical compounds. Existing approaches work with molecular graphs and thus ignore the location of atoms in space, which restricts them to 1) generating single organic molecules and 2) heuristic reward functions. To address this, we present a novel RL formulation for molecular design in Cartesian coordinates, thereby extending the class of molecules that can be built. Our reward function is directly based on fundamental physical properties such as the energy, which we approximate via fast quantum-chemical methods. To enable progress towards de-novo molecular design, we introduce MolGym, an RL environment comprising several challenging molecular design tasks along with baselines. In our experiments, we show that our agent can efficiently learn to solve these tasks from scratch by working in a translation and rotation invariant state-action space.

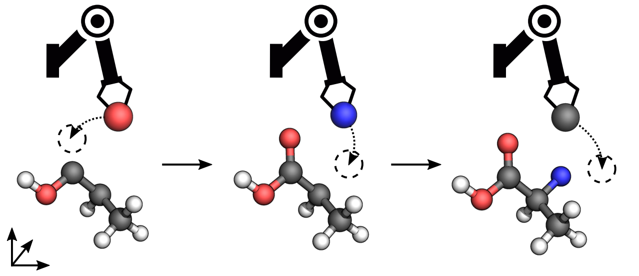


Figure 1. Visualization of the molecular design process presented in this work. The RL agent (depicted by a robot arm) sequentially places atoms onto a canvas. By working directly in Cartesian coordinates, the agent learns to build structures from a very general class of molecules. Learning is guided by a reward that encodes fundamental physical properties. Bonds are only for illustration.

1. Introduction

Finding new chemical compounds with desired properties is a challenging task with important applications such as *de novo* drug design and materials discovery (Schneider et al., 2019). The diversity of synthetically feasible chemicals that can be considered as potential drug-like molecules was estimated to be between 10^{30} and 10^{60} (Polishchuk et al., 2013), making exhaustive search hopeless.

Recent applications of machine learning have accelerated the search for new molecules with specific desired properties. Generative models such as variational autoencoders (VAEs) (Gómez-Bombarelli et al., 2018), recurrent neural networks (RNNs) (Segler et al., 2018), and generative adversarial networks (GANs) (De Cao & Kipf, 2018) have been successfully applied to propose potential drug candidates. Despite recent advances in generating valid structures, proposing truly novel molecules beyond the training data distribution remains a challenging task. This issue is exacerbated for many classes of molecules (e.g. transition metals), where such a representative dataset is not even available.

An alternative strategy is to employ RL, in which an agent builds new molecules in a step-wise fashion (e.g., Olivecrona et al. (2017), Guimaraes et al. (2018), Zhou et al. (2019), Zhavoronkov et al. (2019)). Training an RL agent only requires samples from a reward function, alleviating the need for an existing dataset of molecules. However, the choice of state representation in current models still severely limits the class of molecules that can be generated. In particular, molecules are commonly described by graphs, where atoms and bonds are represented by nodes and edges, respectively. Since a graph is a simplified model of the physical representation of molecules in the real world, one is limited to the generation of single organic molecules. Other types of molecules cannot be appropriately described as this representation lacks important three-dimensional (3D) information, i.e. the relative position of atoms in space. For example, systems consisting of multiple molecules cannot be generated for this reason. Furthermore, it prohibits the

[*]Equal contribution [1]Department of Engineering, University of Cambridge, Cambridge, UK. Correspondence to: Gregor N. C. Simm <gncs2@cam.ac.uk>.

Proceedings of the 37th International Conference on Machine Learning, Online, PMLR 119, 2020. Copyright 2020 by the author(s).

use of reward functions based on fundamental physical laws; instead, one has to resort to heuristic physicochemical parameters, e.g. the Wildman-Crippen partition coefficient (Wildman & Crippen, 1999). Lastly, it is not possible to impose geometric constraints on the design process, e.g. those given by the binding pocket of a protein which the generated molecule is supposed to target.

In this work, we introduce a novel RL formulation for molecular design in which an agent places atoms from a given bag of atoms onto a 3D canvas (see Fig. 1). As the reward function is based on fundamental physical properties such as energy, this formulation is not restricted to the generation of molecules of a particular type. We thus encourage the agent to implicitly learn the laws of atomic interaction from scratch to build molecules that go beyond what can be represented with graph-based RL methods. To enable progress towards designing such molecules, we introduce a new RL environment called MolGym. It comprises a suite of tasks in which both single molecules and molecule clusters need to be constructed. For all of these tasks, we provide baselines using quantum-chemical calculations. Finally, we propose a novel policy network architecture that can efficiently learn to solve these tasks by working in a translation and rotation invariant state-action space.

In summary, our contributions are as follows:

- we propose a novel RL formulation for general molecular design in Cartesian coordinates (Section 2.2);
- we design a reward function based on the electronic energy, which we approximate via fast quantum-chemical calculations (Section 2.3);
- we present a translation and rotation invariant policy network architecture for molecular design (Section 3);
- we introduce MolGym, an RL environment comprising several molecular design tasks along with baselines based on quantum-chemical calculations (Section 5.1);
- we perform experiments to evaluate the performance of our proposed policy network using standard policy gradient methods (Section 5.2).

2. Reinforcement Learning for Molecular Design Guided by Quantum Mechanics

In this section, we provide a brief introduction to RL and present our novel RL formulation for molecular design in Cartesian coordinates.

2.1. Background: Reinforcement Learning

In the standard RL framework, an agent interacts with the environment in order to maximize some reward. We consider a fully observable environment with deterministic dynamics. Such an environment is formally described by a Markov decision process (MDP) $\mathcal{M} = (\mathcal{S}, \mathcal{A}, \mathcal{T}, \mu_0, \gamma, T, r)$ with state space $\mathcal{S}$, action space $\mathcal{A}$, transition function $\mathcal{T} : \mathcal{S} \times \mathcal{A} \mapsto S$, initial state distribution μ_0, discount factor $\gamma \in (0, 1]$, time horizon T and reward function $r : \mathcal{S} \times \mathcal{A} \mapsto \mathbb{R}$. The value function $V^\pi(s_t)$ is defined as the expected discounted return when starting from state s_t and following policy π thereafter, i.e. $V^\pi(s_t) = \mathbb{E}_\pi[\sum_{t'=t}^{T} \gamma^{t'} r(s_{t'}, a_{t'})|s_t]$. The goal is to learn a stochastic policy $\pi(a_t|s_t)$ that maximizes the expected discounted return $J(\theta) = \mathbb{E}_{s_0 \sim \mu_0}[V^\pi(s_0)]$.

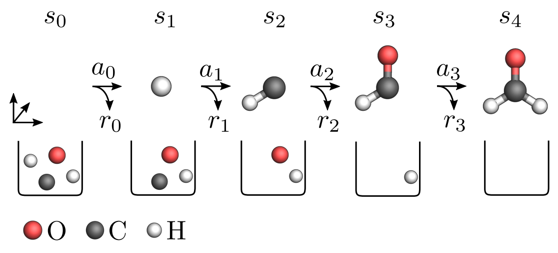


Figure 2. Rollout of an episode with bag $\beta_0 = \mathrm{CH_2O}$. The agent constructs a molecule by sequentially placing atoms from the bag onto the 3D canvas until the bag is empty.

Policy Gradient Algorithms Policy gradient methods are well-suited for RL in continuous action spaces. These methods learn a parametrized policy π_θ by performing gradient ascent in order to maximize $J(\theta)$. More recent algorithms (Schulman et al., 2015; 2017) improve the stability during learning by constraining the policy updates. For example, proximal policy optimization (PPO) (Schulman et al., 2017) employs a clipped surrogate objective. Denoting the probability ratio between the updated and the old policy as $r_t(\theta) = \frac{\pi_\theta(a_t|s_t)}{\pi_{\theta_{\text{old}}}(a_t|s_t)}$, the clipped objective J^{CL} is given by

$$J^{\text{CL}}(\theta) = \mathbb{E}\left[\min(r_t(\theta)\hat{A}_t, \text{clip}(r_t(\theta), 1-\epsilon, 1+\epsilon)\hat{A}_t)\right],$$

where $\hat{A}_t$ is an estimator of the advantage function, and ϵ is a hyperparameter that controls the interval beyond which $r(\theta)$ gets clipped. To further reduce the variance of the gradient estimator, actor-critic approaches (Konda & Tsitsiklis, 2000) are often employed. The idea is to use the value function (i.e. the critic) to assist learning the policy (i.e. the actor). If the actor and critic share parameters, the objective becomes

$$J^{\text{AC}}(\theta) = \mathbb{E}\left[J^{\text{CL}}(\theta) - c_1 J^{\text{V}} + c_2 \mathbb{H}[\pi_\theta|s_t]\right],$$

where c_1, c_2 are coefficients, $J^{\text{V}} = (V^\pi_\phi(s_t) - V^{\text{target}})^2$ is a squared-error loss, and $\mathbb{H}$ is an entropy regularization term to encourage sufficient exploration.

2.2. Setup

We design molecules by sequentially drawing atoms from a given bag and placing them onto a 3D *canvas*. This task can be formulated as a sequential decision-making problem in an MDP with deterministic transition dynamics, where

- state $s_t = (\mathcal{C}_t, \beta_t)$ contains the canvas $\mathcal{C}_t = \mathcal{C}_0 \cup \{(e_i, x_i)\}_{i=0}^{t-1}$, i.e. a set of atoms with chemical element $e_i \in \{\mathrm{H}, \mathrm{C}, \mathrm{N}, \mathrm{O}, \dots\}$ and position $x_i \in \mathcal{R}^3$ placed until time $t-1$, as well as a bag $\beta_t = \{(e, m(e))\}$ of atoms still to be placed; $\mathcal{C}_0$ can either be empty, $\mathcal{C}_0 = \emptyset$, or contain a set of atoms, i.e. $\mathcal{C}_0 = \{(e_i, x_i)\}$ for some $i \in \mathbb{Z}^-$; $m(e)$ is the multiplicity of the element e;
- action $a_t = (e_t, x_t)$ contains the element $e_t \in \beta_t$ and position $x_t \in \mathbb{R}^3$ of the next atom to be placed;
- deterministic transition function $\mathcal{T}(s_t, a_t)$ places an atom through action a_t in state s_t, returning the next state $s_{t+1} = (\mathcal{C}_{t+1}, \beta_{t+1})$ with $\beta_{t+1} = \beta_t \backslash e_t$;
- reward function $r(s_t, a_t)$ quantifies how applying action a_t in state s_t alters properties of the molecule, e.g. the stability of the molecule as measured in terms of its quantum-chemical energy.

Fig. 2 depicts the rollout of an episode. The initial state $(\mathcal{C}_0, \beta_0) \sim \mu_0(s_0)$ of the episode comprises the initial content $\mathcal{C}_0$ of the canvas and a bag of atoms β_0 to be placed, e.g. $\mathcal{C}_0 = \emptyset$, and $\beta_0 = \mathrm{CH_2O}$[1] sampled uniformly from a given set of bags. The agent then sequentially draws atoms from the bag without replacement and places them onto the canvas until the bag is empty.

2.3. Reward Function

One advantage of designing molecules in Cartesian coordinates is that we can evaluate states in terms of quantum-mechanical properties, such as the energy or dipole moment. In this paper, we focus on designing *stable* molecules, i.e. molecules with low energy $E \in \mathbb{R}$; however, linear combinations of multiple desirable properties are possible as well (see Section 5.1 for an example). We define the reward $r(s_t, a_t) = -\Delta_E(s_t, a_t)$ as the negative difference in energy between the resulting molecule described by $\mathcal{C}_{t+1}$ and the sum of energies of the current molecule $\mathcal{C}_t$ and a new atom of element e_t, i.e.

$$\Delta_E(s_t, a_t) = E(\mathcal{C}_{t+1}) - [E(\mathcal{C}_t) + E(e_t)], \qquad (1)$$

where $E(e) \coloneqq E(\{(e, [0,0,0]^\mathsf{T}\})$. Intuitively, the agent is rewarded for placing atoms so that the energy of the resulting molecules is low. Importantly, with this formulation the undiscounted return for building a molecule is independent of the order in which atoms are placed. If the reward only consisted of $E(C_{t+1})$, one would double-count interatomic interactions. As a result, the formulation in Eq. (1) prevents the agent from learning to greedily choose atoms of high atomic number first, as they have low intrinsic energy.

[1]Short hand for $\{(\mathrm{C}, 2), (\mathrm{H}, 2), (\mathrm{O}, 1)\}$.

Quantum-chemical methods, such as the ones based on density functional theory (DFT), can be employed to compute the energy E for a given $\mathcal{C}$. Since such methods are computationally demanding in general, we instead choose to evaluate the energy using the semi-empirical Parametrized Method 6 (PM6) (Stewart, 2007) as implemented in the software package SPARROW (Husch et al., 2018; Bosia et al., 2019); see the Appendix for details. PM6 is significantly faster than more accurate methods based on DFT and sufficiently accurate for the scope of this study. For example, the energy E of systems containing 10 atoms can be computed within hundreds of milliseconds with PM6; with DFT, this would take minutes. We note that more accurate methods can be used as well if the computational budget is available.

3. Policy

Building molecules in Cartesian coordinates allows to 1) extend molecular design through deep RL to a much broader class of molecules compared to graph-based approaches, and 2) employ reward functions based on fundamental physical properties such as the energy. However, working directly in Cartesian coordinates introduces several additional challenges for policy learning.

Firstly, it would be highly inefficient to naively learn to place atoms directly in Cartesian coordinates since molecular properties are invariant under symmetry operations such as translation and rotation. For instance, the energy of a molecule—and thus the reward—does not change if the molecule gets rotated, yet an agent that is not taking this into account would need to learn these solutions separately. Therefore, we require an agent that is *covariant* to translation and rotation, i.e., if the canvas is rotated or translated, the position x_t of the atom to be placed should be rotated and translated as well. To achieve this, our agent first models the atom's position in *internal* coordinates which are *invariant* under translation and rotation. Then, by mapping from internal to Cartesian coordinates, we obtain a position x_t that features the required *covariance*. The agent's internal representations for states and actions are introduced in Sections 3.1 and 3.2, respectively.

Secondly, the action space contains both discrete (i.e. element e_t) and continuous actions (i.e. position x_t). This is in contrast to most RL algorithms, which assume that the action space is either discrete or continuous. Due to the continuous actions, policy exploration becomes much more challenging compared to graph-based approaches. Further, not all discrete actions are valid in every state, e.g. the element e_t has to be contained in the bag β_t. These issues are addressed in Section 3.2, where we propose a novel actor-critic neural network architecture for efficiently constructing molecules in Cartesian coordinates.

3.1. State Representation

Given that our agent models the position of the atom to be placed in internal coordinates, we require a representation for each atom on the canvas $\mathcal{C}$ that is invariant under translation and rotation of the canvas.[2] To achieve this, we employ SchNet (Schütt et al., 2017; 2018b), a deep learning architecture consisting of continuous-filter convolutional layers that works directly on atoms placed in Cartesian coordinates. SchNet($\mathcal{C}$) produces an embedding of each atom in $\mathcal{C}$ that captures information about its local atomic environment. As shown in Fig. 4 (left), we combine this embedding $\tilde{\mathcal{C}}$ with a latent representation $\tilde{\beta}$ of the bag, yielding a *state embedding* $\tilde{s}$, i.e.

$$\tilde{s} = [\tilde{\mathcal{C}}, \tilde{\beta}], \qquad \tilde{\mathcal{C}} = \text{SchNet}(\mathcal{C}), \quad \tilde{\beta} = \text{MLP}_\beta(\beta), \tag{2}$$

where MLP_β is a multi-layer perceptron (MLP).

3.2. Actor

Action Representation We model the position of the atom to be placed in *internal* coordinates—a commonly used representation for molecular structures in computational chemistry—relative to previously placed atoms. If the canvas is initially empty, $\mathcal{C}_0 = \emptyset$, the agent selects an element e_0 from β_0 and places it at the origin, i.e. $a_0 = (e_0, [0,0,0]^\mathsf{T})$. Once the canvas $\mathcal{C}_t$ contains at least one atom, the agent first decides on a focal atom, $f \in \{1, \ldots, n(\mathcal{C}_t)\}$, where $n(\mathcal{C}_t)$ denotes the number of atoms in $\mathcal{C}_t$. This focal atom represents a local reference point close to which the next atom is going to be placed (see Fig. 3). The agent then models the position $x \in \mathbb{R}^3$ with respect to f in internal coordinates (d, α, ψ), where

- $d \in \mathbb{R}$ is the Euclidean distance between x and the position x_f of the focal atom;
- $\alpha \in [0, \pi]$ is the angle between the two lines defined by (x, x_f) and (x, x_{n1}), where x_{n1} is the position of the atom closest to f; if less than two atoms are on the canvas, α is undefined/unused.
- $\psi \in [-\pi, \pi]$ is the dihedral angle between two intersecting planes spanned by (x, x_f, x_{n1}) and (x_f, x_{n1}, x_{n2}), where x_{n2} is the atom that is the second[3] closest to the focal atom; if less than three atoms are on the canvas, ψ is undefined/unused.

As shown in Fig. 3 (right), these internal coordinates can then be mapped back to Cartesian coordinates x.

Model This action representation suggests a natural generative process: first choose next to which focal atom the new atom is placed, then select its element, and finally decide where to place the atom relative to the focal atom. Therefore, we assume that the policy factorizes as

$$\pi_\theta(\psi, \alpha, d, e, f|s) = p(\psi, \alpha, d|e, f, s) \times p(e|f, s)p(f|s). \tag{3}$$

We model the distributions over f and e as categorical, $\text{Cat}(h)$, where $h_f \in \mathbb{R}^{n(\mathcal{C})}$ and $h_e \in \mathbb{R}^{E_\text{max}}$ are the logits predicted by separate MLPs, and E_max is the largest atomic number that can be selected. Further, $p(\psi, \alpha, d|e, f, s)$ is factored into a product of univariate Gaussian distributions $\mathcal{N}(\mu, \sigma^2)$, where the means μ_d, μ_α and μ_ψ are given by an MLP and the standard deviations σ_d, σ_α and σ_ψ are global parameters. Formally,

$$h_f = \text{MLP}_f(\tilde{s}), \tag{4}$$

$$h_e = \text{MLP}_e(\tilde{s}_f), \tag{5}$$

$$\mu_d, \mu_\alpha, \mu_\psi = \text{MLP}_\text{cont}(\tilde{s}_f, \mathbb{1}(e)), \tag{6}$$

where $\tilde{s}_f = [\tilde{\mathcal{C}}_f, \tilde{\beta}]$ is the state embedding of the focal atom $f \sim \text{Cat}(f; h_f)$, $\mathbb{1}(e)$ is a one-hot vector representation of element $e \sim \text{Cat}(e; h_e)$, and $d \sim \mathcal{N}(d; \mu_d, \sigma_d^2)$, $\alpha \sim \mathcal{N}(\alpha; \mu_\alpha, \sigma_\alpha^2)$, and $\psi \sim \mathcal{N}(\psi; \mu_\psi, \sigma_\psi^2)$ are sampled from their respective distributions. The model is shown in Fig. 4.

Maintaining Valid Actions As the agent places atoms onto the canvas during a rollout, the number of possible focal atoms f increases and the number of elements e to choose from decreases. To guarantee that the agent only chooses valid actions, i.e. $f \in \{1, \ldots, n(\mathcal{C})\}$ and $e \in \beta$, we mask out invalid focal atoms and elements by setting their probabilities to zero and re-normalizing the categorical distributions. Neither the agent nor the environment makes use of ad-hoc concepts like valence or bond connectivity—any atom on the canvas can potentially be chosen.

Learning the Dihedral Angle The sign of the dihedral angle ψ depends on the two nearest neighbors of the focal

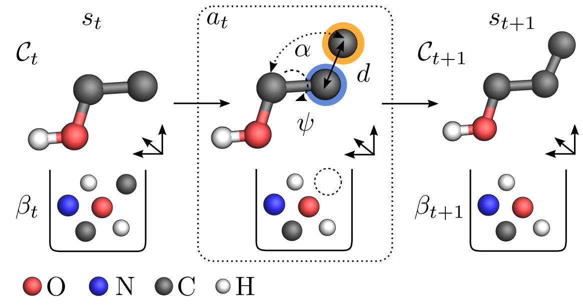


Figure 3. Construction of a molecule using an action-space representation that is invariant under translation and rotation. **Left:** Current state s_t with canvas $\mathcal{C}_t$ and remaining bag β_t. **Center:** Action a_t adds an atom from the bag (highlighted in orange) relative to the focus f (highlighted in blue). The relative coordinates (d, α, ψ) uniquely determine its absolute position. **Right:** Resulting state s_{t+1} after applying action a_t in state s_t.

[2] We omit the time index when it is clear from the context.

[3] In the unlikely event that two atoms are exactly equally far from the focal atom, a random order for x_{n1} and x_{n2} is chosen.

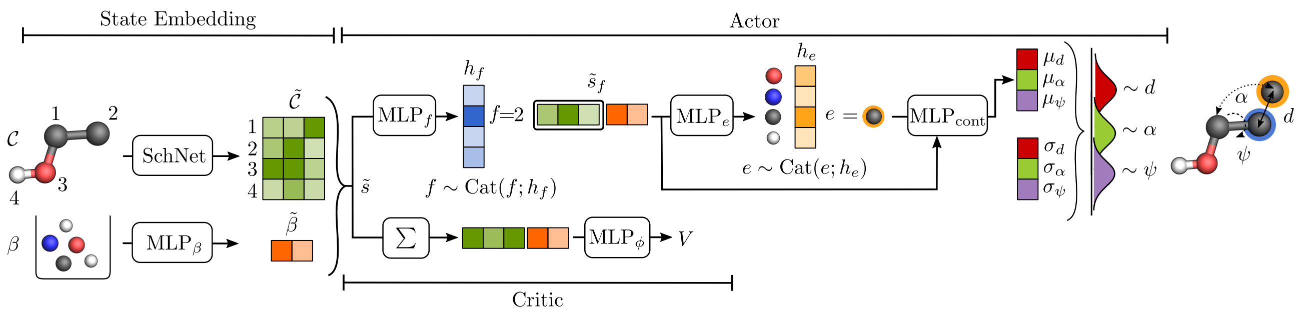


Figure 4. Illustration of the state embedding, actor and critic network. The canvas $\mathcal{C}$ and the bag of atoms β are fed to the state embedding network to obtain a translation and rotation invariant state representation $\tilde{s}$. The actor network then selects 1) a focal atom f, 2) an element e, and 3) internal coordinates (d, α, ψ). The critic takes the bag and the sum across all atoms on the canvas to compute a value V.

atom and is difficult to learn, especially if the two atoms are nearly equally close to the focal atom. In practice, we therefore learn the absolute value $|\psi| \in [0, \pi]$ instead of ψ, as well as the sign $\kappa \in \{+1, -1\}$, such that $\psi = \kappa|\psi|$. To estimate κ, we exploit the fact that the transition dynamics are deterministic. We generate embeddings of both possible next states (for $\kappa = +1$ and $\kappa = -1$) and select the embedding of the atom just added, which we denote by $\tilde{s}_+$ and $\tilde{s}_-$. We then choose $\kappa = +1$ over $\kappa = -1$ with probability

$$p_+ = \frac{\exp(u_+)}{\exp(u_+) + \exp(u_-)}, \tag{7}$$

such that $p(\kappa|\,|\psi|, \alpha, d, e, f, s) = \text{Ber}(\kappa; p_+)$, where $u_\pm = \text{MLP}_\kappa(\tilde{s}_{\pm,})$; we further motivate this choice in the Appendix. Thus, the policy is given by $\pi_\theta(\kappa|\psi|, \alpha, d, e, f|s) = p(\kappa|\,|\psi|, \alpha, d, e, f, s)p(|\psi|, \alpha, d|e, f, s)p(e|f, s)p(f|s)$.

3.3. Critic

The critic needs to compute a value for the entire state s. Since the canvas is growing as more atoms are taken from the bag and placed onto the canvas, a pooling operation is required. Here, we compute the sum over all atomic embeddings $\tilde{C}_i$. Thus, the critic is given by

$$V_\phi(s) = \text{MLP}_\phi\left(\sum_{i=1}^{n(\mathcal{C})} \tilde{C}_i, \tilde{\beta}\right), \tag{8}$$

where MLP$_\phi$ is an MLP that computes value V (see Fig. 4).

3.4. Optimization

We employ PPO (Schulman et al., 2017) to learn the parameters (θ, ϕ) of the actor π_θ and critic V_ϕ, respectively. While most RL algorithms can only deal with either continuous or discrete action spaces and thus require additional modifications to handle both (Masson et al., 2016; Wei et al., 2018; Xiong et al., 2018), PPO can be applied directly as is. To help maintain sufficient exploration throughout learning, we include an entropy regularization term over the policy. However, note that the entropies of the continuous and categorical distributions often have different magnitudes; further, in this setting the entropies over the categorical distributions vary significantly throughout a rollout: as the agent places more atoms, the support of the distribution over valid focal atoms f increases and the support of the distribution over valid elements e decreases. To mitigate this issue, we only apply entropy regularization to the categorical distributions, which we find to be sufficient in practice.

4. Related Work

Deep Generative Models A prevalent strategy for molecular design based on machine learning is to employ deep generative models. These approaches first learn a latent representation of the molecules and then perform a search in latent space (e.g., through gradient descent) to discover new molecules with sought chemical properties. For example, Gómez-Bombarelli et al. (2018); Kusner et al. (2017); Blaschke et al. (2018); Lim et al. (2018); Dai et al. (2018) utilized VAEs to perform search or optimization in a latent space to find new molecules. Segler et al. (2018) used RNNs to design molecular libraries. The aforementioned approaches generate SMILES strings, a linear string notation, to describe molecules (Weininger, 1988). Further, there exist a plethora of generative models that work with graph representations of molecules (e.g., Jin et al. (2017); Bradshaw et al. (2019a); Li et al. (2018a;b); Liu et al. (2018); De Cao & Kipf (2018); Bradshaw et al. (2019b)). In these methods, atoms and bonds are represented by nodes and edges, respectively. Brown et al. (2019) developed a benchmark suite for graph-based generative models, showing that generative models outperform classical approaches for molecular design. While the generated molecules are shown to be valid (De Cao & Kipf, 2018; Liu et al., 2018) and synthesizable (Bradshaw et al., 2019b), the generative model is restricted to a (small) region of chemical space for which the graph representation is valid, e.g. single organic molecules.

3D Point Cloud Generation Another downside of string- and graph-based approaches is their neglect of information encoded in the interatomic distances. To this end, Gebauer et al. (2018; 2019) proposed a generative neural network for sequentially placing atoms in Cartesian coordinates. While their model respects local symmetries by construction, atoms are placed on a 3D grid. Further, similar to aforementioned approaches, this model depends on a dataset to exist that covers the particular class of molecules for which one seeks to generate new molecules.

Reinforcement Learning Olivecrona et al. (2017), Guimaraes et al. (2018), Putin et al. (2018), Neil et al. (2018) and Popova et al. (2018) presented RL approaches based on string representations of molecules. They successfully generated molecules with given desirable properties but, similar to other generative models using SMILES strings, struggled with chemical validity. You et al. (2018) proposed a graph convolutional policy network based on graph representations of molecules, where the reward function is based on empirical properties such as the drug-likeliness. While this approach was able to consistently produce valid molecules, its performance still depends on a dataset required for pre-training. Considering the large diversity of chemical structures, the generation of a dataset that covers the whole chemical space is hopeless. To address this limitation, Zhou et al. (2019) proposed an agent that learned to generate molecules from scratch using a Deep Q-Network (DQN) (Mnih et al., 2015). However, such graph-based RL approaches are still restricted to the generation of single organic molecules for which this representation was originally designed. Further, graph representations prohibit the use of reward functions based on fundamental physical laws, and one has to resort to heuristics instead. Finally, geometric constraints cannot be imposed on the design process. Jørgensen et al. (2019) introduced an atomistic structure learning algorithm, called *ALSA*, that utilizes a convolutional neural network to build 2D structures and planar compounds atom by atom.

5. Experiments

We perform experiments to evaluate the performance of the policy introduced in Section 3. While prior work has focused on building molecules using molecular graph representations, we are interested in designing molecules in Cartesian coordinates. To this end, we introduce a new RL environment called MolGym in Section 5.1. It comprises a set of molecular design tasks, for which we provide baselines using quantum-chemical calculations. See the Appendix for details on how the baselines are determined. [4]

We use MolGym to answer the following questions: 1) can our agent learn to construct single molecules in Cartesian coordinates from scratch, 2) does our approach allow building molecules across multiple bags simultaneously, 3) are we able to scale to larger molecules, and 4) can our agent construct systems comprising multiple molecules?

5.1. Tasks

We propose three different tasks for molecular design in Cartesian coordinates, which are instances of the MDP formulation introduced in Section 2.2: *single-bag*, *multi-bag*, and *solvation*. More formally, the tasks are as follows:

Single-bag Given a bag, learn to design stable molecules. This task assesses an agent's ability to build single stable molecules. The reward function is given by $r(s_t, a_t) = -\Delta_E(s_t, a_t)$, see Eq. (1). If the reward is below a threshold of -0.6, the molecule is deemed invalid and the episode terminates prematurely with the reward clipped at -0.6.[5]

Multi-bag Given multiple bags with one of them being randomly selected before each episode, learn to design stable molecules. This task focuses on the agent's capabilities to learn to build different molecules of different composition and size at the same time. The same reward function as in the *single-bag* task is used. Offline performance is evaluated in terms of the average return across bags. Similarly, the baseline is given by the average optimal return over all bags.

Solvation The task is to learn to place water molecules around an existing molecule (i.e. $\mathcal{C}_0$ is non-empty). This task assesses an agent's ability to distinguish intra- and inter-molecular interactions, i.e. the atomic interactions within a molecule and those between molecules. These interactions are paramount for the accurate description of chemistry in the liquid phase. In this task, we deviate from the protocol used in the previous experiments as follows. Initially, the agent is provided with an H_2O bag. Once the bag is empty, the environment will refill it and the episode continues. The episode terminates once $n \in \mathbb{N}^+$ bags of H_2O have been placed on the canvas. By refilling the H_2O bag $n-1$ times instead of providing a single $H_{2n}O_n$ bag, the agent is guided towards building H_2O molecules. [6] The reward function is augmented with a penalty term for placing atoms far away from the center, i.e. $r(s_t, a_t) = -\Delta_E - \rho\|x\|_2$, where ρ is a hyper-parameter. This corresponds to a soft constraint on the radius at which the atoms should be placed. This is a task a graph-based RL approach could not solve.

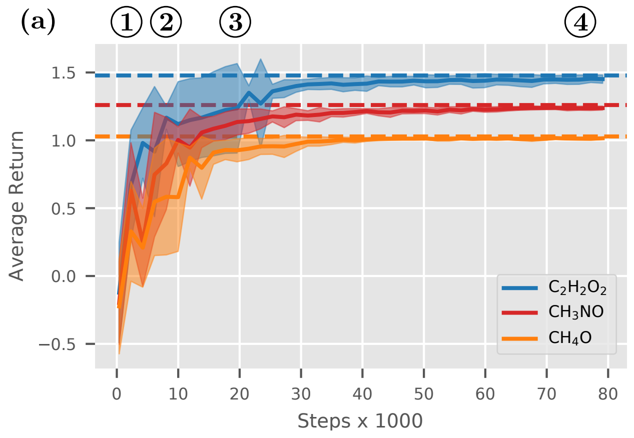
5.2. Results

In this section, we use the tasks specified in Section 5.1 to evaluate our proposed policy. We further assess the chemical

[4]Source code of the agent and environment is available at `https://github.com/gncs/molgym`.

[5]Δ_E is on the order of magnitude of -0.1 Hartree, resulting in a reward of around 0.25 for a well placed atom.

[6]A comparison of the two protocols is given in the Appendix.



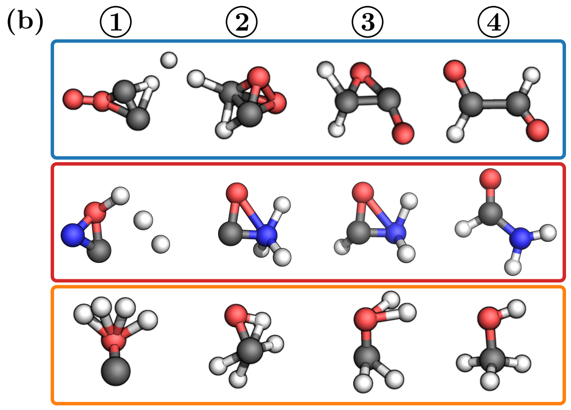


Figure 5. **(a)** Average offline performance on the *single-bag* task for bags CH_3NO, CH_4O and $C_2H_2O_2$ across 10 seeds. Dashed lines denote optimal returns for each bag, respectively. Error bars show two standard deviations. **(b)** Generated molecular structures at different terminal states over time, showing the agent's learning progress.

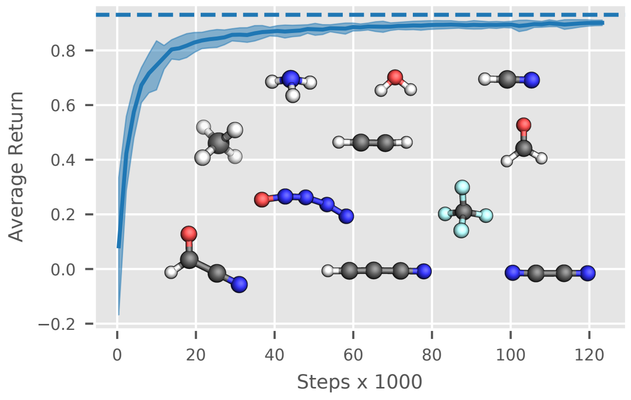


Figure 6. Average offline performance on the *multi-bag* task, using 11 bags consisting of up to five atoms across 10 seeds. The dashed line denotes the optimal average return. Error bars show two standard deviations. The molecular structures shown are the terminal states at the end of training from one seed.

validity, diversity and stability of the generated structures. Experiments were run on a 16-core Intel Xeon Skylake 6142 CPU with 2.6GHz and 96GB RAM. Details on the model architecture and hyperparameters are in the Appendix.

Learning to Construct Single Molecules In this toy experiment, we train the agent on the *single-bag* task for the bags CH_3NO, CH_4O and $C_2H_2O_2$, respectively. Fig. 5 shows that the agent was able to learn the rules of chemical bonding and interatomic distances from scratch. While on average the agent reaches 90% of the optimal return after only 12 000 steps, the snapshots in Fig. 5 (b) highlight that the last 10% determine chemical validity. As shown in Fig. 5 (b), the model first learns the atomic distances d, followed by the angles α and the dihedral angles ψ.

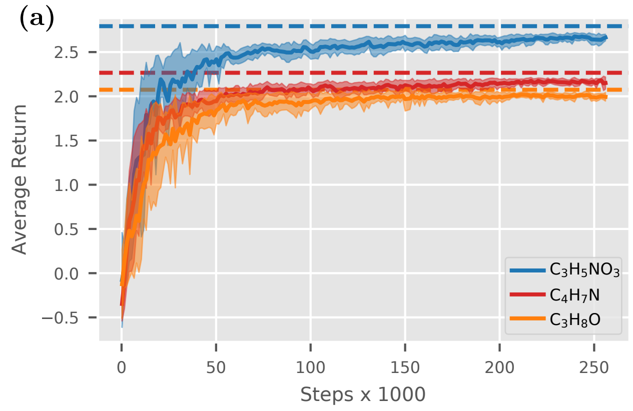
Learning across Multiple Bags We train the agent on the *multi-bag* task using all formulas contained in the QM9 dataset (Ruddigkeit et al., 2012; Ramakrishnan et al., 2014) with up to 5 atoms, resulting in 11 bags (see Table 1). Despite their small size, the molecules feature a diverse set of bonds (single, double, and triple) and geometries (linear, trigonal planar, and tetrahedral). From the performance and from visual inspection of the generated molecular structures shown in Fig. 6, it can be seen that a single policy is able to build different molecular structures across multiple bags. For example, it learned that a carbon atom can have varying number and type of neighboring atoms leading to specific bond distance, angles, and dihedral angles.

Table 1. QM9 bags used in the experiments.

Experiment	QM9 Bags Used
Single-bag	$C_2H_2O_2, CH_3NO, CH_4O$
Multi-bag	$H_2O, CHN, C_2N_2, H_3N, C_2H_2, CH_2O, C_2HNO, N_4O, C_3HN, CH_4, CF_4$
Single-bag (large)	$C_3H_5NO_3, C_4H_7N, C_3H_8O$

Scaling to Larger Molecules To study our agent's ability to construct large molecules we let it solve the *single-bag* task with the bags $C_3H_5NO_3$, C_3H_8O, and C_4H_7N. Results are shown in Fig. 7. After 154 000 steps, the agent achieved an average return of 2.60 on $C_3H_5NO_3$ (maximum across seeds at 2.72, optimum at 2.79), 2.17 on C_4H_7N (2.21, 2.27), and 1.98 on C_3H_8O (2.04, 2.07). While the agent did not always find the most stable configurations, it was able to explore a diverse set of chemically valid structures (including bimolecular structures, see Appendix).

Constructing Molecular Clusters We task the agent to place 5 water molecules around a formaldehyde molecule, i.e. $\mathcal{C}_0 = CH_2O$ and $n = 5$. The distance penalty parameter



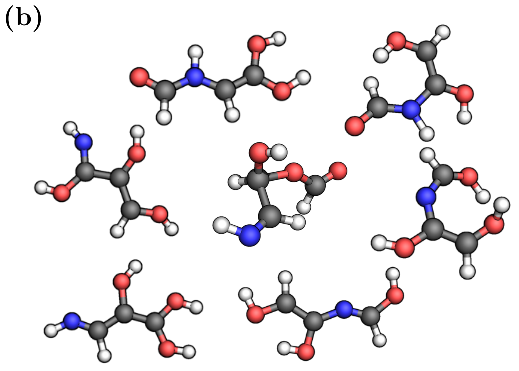


Figure 7. **(a)** Average offline performance on the *single-bag* task for bags $C_3H_5NO_3$, C_3H_8O and C_4H_7N across 10 seeds. Dashed lines denote optimal return for each bag, respectively. Error bars show two standard deviations. **(b)** Selection of molecular structures generated by trained models for the bag $C_3H_5NO_3$. For the bags C_3H_8O and C_4H_7N, see the Appendix.

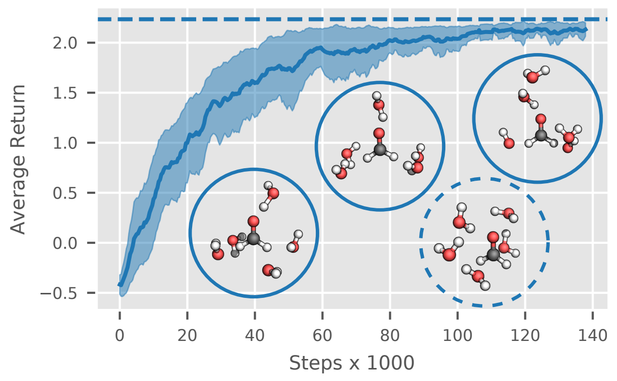


Figure 8. Average offline performance on the *solvation* task with 5 H_2O molecules across 10 seeds. Error bars show two standard errors. The plot is smoothed across five evaluations for better readability. The dashed line denotes the optimal return. A selection of molecular clusters generated by trained models are shown in solid circles; for comparison, a stable configuration obtained through structure optimization is depicted in a dashed circle.

ρ is set to 0.01.[7] From Fig. 8, we observe that the agent is able to learn to construct H_2O molecules and place them in the vicinity of the solute. A good placement also allows for hydrogen bonds to be formed between water molecules themselves and between water molecules and the solute (see Fig. 8, dashed circle). In most cases, our agent arranges H_2O molecules such that these bonds can be formed (see Fig. 8, solid circles). The lack of hydrogen bonds in some structures could be attributed to the approximate nature of the quantum-chemical method used in the reward function. Overall, this experiment showcases that our agent is able to learn both intra- and intermolecular interactions, going beyond what graph-based agents can learn.

[7]Further experiments on the *solvation* task are in the Appendix.

Table 2. Assessment of generated structures in different experiments by chemical validity, RMSD (in Å), and diversity.

Task	Experiment	Validity	RMSD	Diversity
Single-bag	$C_2H_2O_2$	0.90	0.32	3
	CH_3NO	0.70	0.20	3
	CH_4O	0.80	0.11	1
Multi-bag	-	0.78	0.05	22
Single-bag (large)	$C_3H_5NO_3$	0.70	0.39	40
	C_4H_7N	0.80	0.29	20
	C_3H_8O	0.90	0.47	4
	$C_7H_8N_2O_2$	0.60	0.61	61
Solvation	Formaldehyde	0.80	1.03	1
	Acetonitrile	0.90	1.06	1
	Ethanol	0.90	0.92	1

Quality Assessment of Generated Molecules In the spirit of the *GuacaMol* benchmark (Brown et al., 2019), we assess the molecular structures generated by the agent with respect to chemical validity, diversity and structural stability for each experiment. To enable a comparison with existing approaches, we additionally ran experiments with the bag $C_7H_8N_2O_2$, the stoichiometry of which is taken from the *GuacaMol* benchmark (Brown et al., 2019).

The results are shown in Table 2. To determine the validity and stability of the generated structures, we first took the terminal states of the last iteration for a particular experiment. Structures are considered valid if they can be successfully parsed by RDKit (Landrum, 2019). However, those consisting of multiple molecules were not considered valid (except in the *solvation* task). The validity reported in Table 2 is the ratio of valid molecules over 10 seeds.

All valid generated structures underwent a structure optimization using the PM6 method (see Appendix for more details). Then, the RMSD (in Å) between the original and the optimized structure were computed. In Table 2, the median RMSD over all generated structures is given per experiment. In the approach by Gebauer et al. (2019), an average RMSD of ≈ 0.25 Å is reported. Due to significant differences in approach, application, and training procedure we forego a direct comparison of the methods.

Further, two molecules are considered identical if the SMILES strings generated by RDKit are the same. The diversity reported in Table 2 is the total number of unique and valid structures generated through training over 10 seeds.

6. Discussion

This work is a first step towards general molecular design through RL in Cartesian coordinates. One limitation of the current formulation is that we need to provide bags for which we know good solutions exist when placed completely. While being able to provide such prior knowledge can be beneficial, we are currently restricted to designing molecules of known formulas. A possible solution is to provide bags that are larger than necessary, e.g. generated randomly or according to some fixed budget for each element, and enable the agent to stop before the bag is empty.

Compared to graph-based approaches, constructing molecules by sequentially placing atoms in Cartesian coordinates greatly increases the flexibility in terms of the type of molecular structures that can be built. However, it also makes the exploration problem more challenging: whereas in graph-based approaches a molecule can be expanded by adding a node and an edge, here, the agent has to learn to precisely position an atom in Cartesian coordinates from scratch. As a result, the molecules we generate are still considerably smaller. Several approaches exist to mitigate the exploration problem and improve scalability, including: 1) *hierarchical RL*, where molecular fragments or entire molecules are used as high-level actions; 2) *imitation learning*, in which known molecules are converted into expert trajectories; and 3) *curriculum learning*, where the complexity of the molecules to be built increases over time.

7. Conclusion

We have presented a novel RL formulation for molecular design in Cartesian coordinates, in which the reward function is based on quantum-mechanical properties such as the energy. We further proposed an actor-critic neural network architecture based on a translation and rotation invariant state-action representation. Finally, we demonstrated that our model can efficiently solve a range of molecular design tasks from our MolGym RL environment from scratch.

In future work, we plan to increase the scalability of our approach and enable the agent to stop before a given bag is empty. Moreover, we are interested in combining the reward with other properties such as drug-likeliness and applying our approach to other classes of molecules, e.g. transition-metal catalysts.

Acknowledgements

We would like to thank the anonymous reviewers for their valuable feedback. We further thank Austin Tripp and Vincent Stimper for useful discussions and feedback. GNCS acknowledges funding through an Early Postdoc.Mobility fellowship by the Swiss National Science Foundation (P2EZP2_181616). RP receives funding from iCASE grant #1950384 with support from Nokia.

References

Blaschke, T., Olivecrona, M., Engkvist, O., Bajorath, J., and Chen, H. Application of Generative Autoencoder in De Novo Molecular Design. *Mol. Inf.*, 37(1-2):1700123, 2018.

Bosia, F., Husch, T., Vaucher, A. C., and Reiher, M. qcscine/sparrow: Release 1.0.0, 2019. URL https://doi.org/10.5281/zenodo.3244106.

Bradshaw, J., Kusner, M. J., Paige, B., Segler, M. H. S., and Hernández-Lobato, J. M. A generative model for electron paths. In *International Conference on Learning Representations*, 2019a.

Bradshaw, J., Paige, B., Kusner, M. J., Segler, M., and Hernández-Lobato, J. M. A Model to Search for Synthesizable Molecules. In *Advances in Neural Information Processing Systems*, pp. 7935–7947, 2019b.

Brown, N., Fiscato, M., Segler, M. H., and Vaucher, A. C. GuacaMol: Benchmarking Models for de Novo Molecular Design. *J. Chem. Inf. Model.*, 59(3):1096–1108, 2019. doi: 10.1021/acs.jcim.8b00839.

Dai, H., Tian, Y., Dai, B., Skiena, S., and Song, L. Syntax-directed variational autoencoder for structured data. In *International Conference on Learning Representations*, 2018.

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

Gebauer, N. W. A., Gastegger, M., and Schütt, K. T. Generating equilibrium molecules with deep neural networks. *arXiv preprint arXiv:1810.11347*, 2018.

Gebauer, N. W. A., Gastegger, M., and Schütt, K. T. Symmetry-adapted generation of 3d point sets for the targeted discovery of molecules. In *Advances in Neural Information Processing Systems*, pp. 7564–7576, 2019.

Gómez-Bombarelli, R., Wei, J. N., Duvenaud, D., Hernández-Lobato, J. M., Sánchez-Lengeling, B., Sheberla, D., Aguilera-Iparraguirre, J., Hirzel, T. D., Adams, R. P., and Aspuru-Guzik, A. Automatic Chemical Design Using a Data-Driven Continuous Representation of Molecules. *ACS Cent. Sci.*, 4(2):268–276, 2018.

Guimaraes, G. L., Sanchez-Lengeling, B., Outeiral, C., Farias, P. L. C., and Aspuru-Guzik, A. Objective-Reinforced Generative Adversarial Networks (ORGAN) for Sequence Generation Models. *arXiv preprint arXiv:1705.10843*, 2018.

Husch, T. and Reiher, M. Comprehensive Analysis of the Neglect of Diatomic Differential Overlap Approximation. *J. Chem. Theory Comput.*, 14(10):5169–5179, 2018.

Husch, T., Vaucher, A. C., and Reiher, M. Semiempirical molecular orbital models based on the neglect of diatomic differential overlap approximation. *Int. J. Quantum Chem.*, 118(24):e25799, 2018.

Jin, W., Coley, C., Barzilay, R., and Jaakkola, T. Predicting Organic Reaction Outcomes with Weisfeiler-Lehman Network. In *Advances in Neural Information Processing Systems*, pp. 2607–2616, 2017.

Jørgensen, M. S., Mortensen, H. L., Meldgaard, S. A., Kolsbjerg, E. L., Jacobsen, T. L., Sørensen, K. H., and Hammer, B. Atomistic structure learning. *J. Chem. Phys.*, 151 (5):054111, 2019. doi: 10.1063/1.5108871.

Konda, V. R. and Tsitsiklis, J. N. Actor-critic algorithms. In *Advances in Neural Information Processing Systems*, pp. 1008–1014, 2000.

Kusner, M. J., Paige, B., and Hernández-Lobato, J. M. Grammar Variational Autoencoder. In Precup, D. and Teh, Y. W. (eds.), *International Conference on Machine Learning*, volume 70, pp. 1945–1954. PMLR, 2017.

Landrum, G. RDKit 2019.09.3. http://www.rdkit.org/, 2019. (Accessed: 22. January 2019).

Li, Y., Vinyals, O., Dyer, C., Pascanu, R., and Battaglia, P. Learning Deep Generative Models of Graphs. *arXiv preprint arXiv:1803.03324*, 2018a.

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

Lim, J., Ryu, S., Kim, J. W., and Kim, W. Y. Molecular generative model based on conditional variational autoencoder for de novo molecular design. *J. Cheminf.*, 10:31, 2018.

Liu, Q., Allamanis, M., Brockschmidt, M., and Gaunt, A. Constrained Graph Variational Autoencoders for Molecule Design. In *Advances in Neural Information Processing Systems*, pp. 7795–7804, 2018.

Masson, W., Ranchod, P., and Konidaris, G. Reinforcement learning with parameterized actions. In *Thirtieth AAAI Conference on Artificial Intelligence*, 2016.

Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A. A., Veness, J., Bellemare, M. G., Graves, A., Riedmiller, M., Fidjeland, A. K., Ostrovski, G., Petersen, S., Beattie, C., Sadik, A., Antonoglou, I., King, H., Kumaran, D., Wierstra, D., Legg, S., and Hassabis, D. Human-level control through deep reinforcement learning. *Nature*, 518(7540): 529–533, 2015.

Neil, D., Segler, M., Guasch, L., Ahmed, M., Plumbley, D., Sellwood, M., and Brown, N. Exploring Deep Recurrent Models with Reinforcement Learning for Molecule Design. *OpenReview*, 2018. URL `https://openreview.net/forum?id=HkcTe-bR-`.

Olivecrona, M., Blaschke, T., Engkvist, O., and Chen, H. Molecular de-novo design through deep reinforcement learning. *J. Cheminf.*, 9(1):48, 2017.

Polishchuk, P. G., Madzhidov, T. I., and Varnek, A. Estimation of the size of drug-like chemical space based on GDB-17 data. *J. Comput.-Aided Mol. Des.*, 27(8): 675–679, 2013.

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

Putin, E., Asadulaev, A., Ivanenkov, Y., Aladinskiy, V., Sanchez-Lengeling, B., Aspuru-Guzik, A., and Zhavoronkov, A. Reinforced Adversarial Neural Computer for de Novo Molecular Design. *J. Chem. Inf. Model.*, 58 (6):1194–1204, 2018.

Ramakrishnan, R., Dral, P. O., Rupp, M., and von Lilienfeld, O. A. Quantum chemistry structures and properties of 134 kilo molecules. *Sci. Data*, 1:140022, 2014.

Ruddigkeit, L., van Deursen, R., Blum, L. C., and Reymond, J.-L. Enumeration of 166 Billion Organic Small Molecules in the Chemical Universe Database GDB-17. *J. Chem. Inf. Model.*, 52(11):2864–2875, 2012.

Schneider, P., Walters, W. P., Plowright, A. T., Sieroka, N., Listgarten, J., Goodnow, R. A., Fisher, J., Jansen, J. M., Duca, J. S., Rush, T. S., Zentgraf, M., Hill, J. E., Krutoholow, E., Kohler, M., Blaney, J., Funatsu, K., Luebkemann, C., and Schneider, G. Rethinking drug design in the artificial intelligence era. *Nat. Rev. Drug Discovery*, pp. 1–12, 2019.

Schulman, J., Levine, S., Abbeel, P., Jordan, M., and Moritz, P. Trust region policy optimization. In *International Conference on Machine Learning*, pp. 1889–1897, 2015.

Schulman, J., Wolski, F., Dhariwal, P., Radford, A., and Klimov, O. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.

Schütt, K., Kindermans, P.-J., Sauceda Felix, H. E., Chmiela, S., Tkatchenko, A., and Müller, K.-R. SchNet: A continuous-filter convolutional neural network for modeling quantum interactions. In *Advances in Neural Information Processing Systems*, pp. 991–1001, 2017.

Schütt, K. T., Kessel, P., Gastegger, M., Nicoli, K. A., Tkatchenko, A., and Müller, K.-R. SchNetPack: A Deep Learning Toolbox For Atomistic Systems. *J. Chem. Theory Comput.*, 2018a.

Schütt, K. T., Sauceda, H. E., Kindermans, P. J., Tkatchenko, A., and Müller, K. R. SchNet–A deep learning architecture for molecules and materials. *J. Chem. Phys.*, 148 (24), 2018b.

Segler, M. H. S., Kogej, T., Tyrchan, C., and Waller, M. P. Generating Focused Molecule Libraries for Drug Discovery with Recurrent Neural Networks. *ACS Cent. Sci.*, 4 (1):120–131, 2018.

Stewart, J. J. P. Optimization of parameters for semiempirical methods V: Modification of NDDO approximations and application to 70 elements. *J. Mol. Model.*, 13(12): 1173–1213, 2007.

Wei, E., Wicke, D., and Luke, S. Hierarchical approaches for reinforcement learning in parameterized action space. In *2018 AAAI Spring Symposium Series*, 2018.

Weininger, D. SMILES, a chemical language and information system. 1. Introduction to methodology and encoding rules. *J. Chem. Inf. Comput. Sci.*, 28(1):31–36, 1988.

Wildman, S. A. and Crippen, G. M. Prediction of Physicochemical Parameters by Atomic Contributions. *J. Chem. Inf. Comput. Sci.*, 39(5):868–873, 1999.

Xiong, J., Wang, Q., Yang, Z., Sun, P., Han, L., Zheng, Y., Fu, H., Zhang, T., Liu, J., and Liu, H. Parametrized deep q-networks learning: Reinforcement learning with discrete-continuous hybrid action space. *arXiv preprint arXiv:1810.06394*, 2018.

You, J., Liu, B., Ying, Z., Pande, V., and Leskovec, J. Graph Convolutional Policy Network for Goal-Directed Molecular Graph Generation. In *Advances in Neural Information Processing Systems*, pp. 6410–6421, 2018.

Zhavoronkov, A., Ivanenkov, Y. A., Aliper, A., Veselov, M. S., Aladinskiy, V. A., Aladinskaya, A. V., Terentiev, V. A., Polykovskiy, D. A., Kuznetsov, M. D., Asadulaev, A., Volkov, Y., Zholus, A., Shayakhmetov, R. R., Zhebrak, A., Minaeva, L. I., Zagribelnyy, B. A., Lee, L. H., Soll, R., Madge, D., Xing, L., Guo, T., and Aspuru-Guzik, A. Deep learning enables rapid identification of potent DDR1 kinase inhibitors. *Nat. Biotechnol.*, 37(9):1038–1040, 2019.

Zhou, Z., Kearnes, S., Li, L., Zare, R. N., and Riley, P. Optimization of Molecules via Deep Reinforcement Learning. *Sci. Rep.*, 9(1):1–10, 2019.