Scientific Research Publishing

Computational Molecular Bioscience, 2021, 11, 69-83
https://www.scirp.org/journal/cmb
ISSN Online: 2165-3453
ISSN Print: 2165-3445

Reinforcement Learning of Molecule Optimization with Bayesian Neural Networks

Wei Hu

Department of Computer Science, Houghton College, Houghton, NY, USA
Email: wei.hu@houghton.edu

How to cite this paper: Hu, W. (2021) Reinforcement Learning of Molecule Optimization with Bayesian Neural Networks. *Computational Molecular Bioscience*, **11**, 69-83.
https://doi.org/10.4236/cmb.2021.114005

Received: October 10, 2021
Accepted: December 24, 2021
Published: December 27, 2021

Abstract

Creating new molecules with desired properties is a fundamental and challenging problem in chemistry. Reinforcement learning (RL) has shown its utility in this area where the target chemical property values can serve as a reward signal. At each step of making a new molecule, the RL agent learns selecting an action from a list of many chemically valid actions for a given molecule, implying a great uncertainty associated with its learning. In a traditional implementation of deep RL algorithms, deterministic neural networks are typically employed, thus allowing the agent to choose one action from one sampled action at each step. In this paper, we proposed a new strategy of applying Bayesian neural networks to RL to reduce uncertainty so that the agent can choose one action from a pool of sampled actions at each step, and investigated its benefits in molecule design. Our experiments suggested the Bayesian approach could create molecules of desirable chemical quality while maintained their diversity, a very difficult goal to achieve in machine learning of molecules. We further exploited their diversity by using them to train a generative model to yield more novel drug-like molecules, which were absent in the training molecules as we know novelty is essential for drug candidate molecules. In conclusion, Bayesian approach could offer a balance between exploitation and exploration in RL, and a balance between optimization and diversity in molecule design.

Keywords

Molecule Design, Bayesian Neural Networks, Reinforcement Learning

1. Introduction

Optimizing molecules for specific chemical properties is of great importance in chemistry. Traditionally, this task has been carried out by human experts with domain knowledge and intuition, which are expensive in terms of cost and time with a very loss rate of success. With its recent advances, deep learning has shown it can increase the scale and accelerate speed of this process. However, this approach heavily relies on large datasets of molecules, which can be hard to acquire in real applications. Furthermore, the models trained on a set of molecules tend to make biased predictions towards the training molecules, thereby limiting the scope of their learning and generalization. As a result, computational techniques for the generation of new molecules from scratch have attracted much attention recently [1]-[7].

The generation of a molecule is a sequential decision process, and the target chemical property values can be used as a reward single. Therefore, reinforcement learning (RL) is a natural choice for this kind of artificial intelligence task. The first issue of applying RL to molecule learning is how to represent a molecule so that computers can process it. The well-known SMILES string representations could produce invalid molecules while the graph representations incline to require pre-defined dataset of molecules, which could make the trained models to generate molecules not very different from the training ones. MolDQN, a deep RL learning algorithm, was introduced to overcome these limitations, which operates directly on molecular graphs to generate 100% chemically valid molecules [8]. Actions in MolDQN such as atom/bond addition and removal are ensured to be chemically valid with help of the domain knowledge in chemistry. One clear advantage of MolDQN and other similar strategies is that they do not require a training set of molecules, thus eliminating the possible model biased caused by the training set [8].

Deep RL algorithms generally use deterministic networks, which represent their weights as single values or point estimates. As a result, these networks tend to perform well in regions with lots of data, but fail to indicate uncertainty in regions with little or no data, giving overconfident decisions. Bayesian neural networks, on the other hand, represent weights as probability distributions, and thus able to express the uncertainty of weight estimation and predictions. They can be trained with variational inference that learns the parameters of weight distributions instead of the weights directly. Bayes back propagation [9] provides a principled way to train Bayesian networks. It takes the gradients on Monte Carlo samples of the loss function to make the typical intractable learning tractable, offering the advantage of allowing more complex prior and variational approximation distributions.

The challenge of balancing the exploitation and exploration is perpetually present in RL, and the trade-off is hard to calculate, due to the lack of a complete knowledge of the environment. On one hand, the agent needs to explore new states so that it would not miss out any good states for bigger rewards. On the other hand, the agent needs to keep visiting the previously discovered good states so that it can learn from experiences gained so far. The uncertainty from the learning process and the environment makes RL particularly difficult and permeates every aspect of RL algorithm design.

Another challenge of machine learning of molecules is the trade-off between optimality and diversity. The former refers to having molecules with desired chemical properties, and the latter refers to having many different molecules, which could then lead to novel drug molecules. Generally, machine learning models can only have one focus, either on optimality (depth) or on diversity (breadth).

This study explored the potential of utilizing Bayesian networks to replace the traditional deterministic networks in deep RL and to assess this idea in the context of molecular generation. Our efforts were to reach the goal of ideal machine learning of molecules: creating molecules with optimized properties and as much diversity as possible. Our current work could also be viewed as a continuation of our earlier works in [10] [11], where multi-agent reinforcement learning and invertible neural networks were applied to molecule design, respectively.

2. Methods

The aim of this work was to use Bayesian neural networks in deep RL and applied this approach to generate new and diverse molecules. Then these newly created molecules were used to train a variational autoencoder to produce more novel molecules. This section explained the techniques employed in this study.

2.1. Bayesian Neural Networks

Deterministic neural networks, when viewed as probabilistic models, have their weights as single values $p(y \mid x, w)$ while Bayesian neural networks have their weights as probability distributions $p(w \mid D)$, where x is the input to a network, y an output of the network, w are the weights, and $D = \{(x_i, y_i)\}$ is the training dataset. The single values of w in the traditional networks can be learned from D by maximum likelihood estimation.

The learning of Bayesian networks involves calculation of the posterior distribution of the weights given the training data. In general, exact Bayesian inference on the weights of a neural network $p(w \mid D)$ is intractable so a simplified variational approximation $q(w \mid \theta)$ is frequently employed. The parameters θ can be found by minimizing the Kullback-Leibler (*KL*) divergence between $q(w \mid \theta)$ and the true posterior $p(w \mid D)$:

$$\begin{aligned} KL\left[q(w \mid \theta) \,\|\, p(w \mid D)\right] &= \int q(w \mid \theta) \log \frac{q(w \mid \theta)}{p(w \mid D)} \mathrm{d}w \\ &= \int q(w \mid \theta) \log \frac{q(w \mid \theta)\, p(D)}{p(W)\, p(D \mid w)} \mathrm{d}w \end{aligned} \tag{1}$$

The last step in (1) is using Bayes formula, and because we want to estimate θ so the terms involving $p(D)$ can be ignored. Thus, we obtain the following intractable loss function:

$$F(D, \theta) = KL\left[q(w \mid \theta) \,\|\, p(w)\right] - E_{q(w \mid \theta)}\left[\log p(D \mid w)\right] \tag{2}$$

where $p(w)$ is the prior distribution of *w*. The first part of the loss is prior dependent or regularizing $q(w|\theta)$ towards $p(w)$ by squeezing $q(w|\theta)$ under $p(w)$ and the second is data dependent or the likelihood of observing D given *w*. The existence of the first term prevents the second term overfitting the model. Bayes by Backprop [9] was introduced to find θ through Monte Carlo sampling to evaluate the expectations:

$$F(D,\theta) \approx \sum_{i=1}^{n} \log q(w^i \mid \theta) - \log p(w^i) - \log p(D \mid w^i) \tag{3}$$

where w^i denotes the$i$th Monte Carlo sample drawn fromthe variational posterior $q(w|\theta)$. As a result, the task of minimizing the loss in (1) can be accomplished via the Monte Carlo gradients on the tractable loss in (2). The work in [9] shows that the derivative of an expectation can be expressed as the expectation of a derivative, under certain conditions. The Monte Carlo technique in (2) removes the need to use a closed form for the first or the second term in (1), thereby reduces any potential bias from a particular selection of the closed form. A common implementation of $q(w|\theta)$ uses a Gaussian distribution with $\theta = (\mu, \sigma)$ where μ is the mean and σ the standard deviation. These probability distributions describe the uncertainty in weights and can be used to estimate uncertainty in predictions.

Since deterministic networks can only output point estimates, Bayesian networks can be viewed as an assemble of infinitely many deterministic networks.

2.2. Reinforcement Learning of Molecule Design

Machine learning is increasingly being applied to the targeted generation of molecules and guided exploration of chemical space. Reinforcement learning is a subfield of machine learning in which an agent learns how to act in an environment so as to maximize its cumulative rewards. Therefore, RL agents are natural fitfor creating molecules with desirable properties as their rewards. MolDQN [8] formulated optimizing molecules as a Markov decision process (MDP). In this MDP, the agent took actions on the molecule to transform it sequentially. Atoms in a molecule are connected with one another by chemical bonds, therefore, three categories of actions were employed: atom addition, bond addition, and bond removal. For the validity of each molecule, "no modification" was also included as an action, which permitted the molecule to remain unchanged at the current step. By only allowing chemically valid actions, all molecules generated by the agent were ensured to be valid. Furthermore, it could learn from scratch without any training molecules. The goal of the agent was to learn a Q function $Q(s,a)$ which represents the long-term rewards for taking action *a* in the state *s*. From the learned values of $Q(s,a)$, the agent could decide what actions to take in a given state, or using a more precise term in RL: the agent could infer a policy from this Q function, here policy is a mapping from states to actions.

In this study, we replaced the deterministic networks in MolDQN with Bayesian networks. The purpose of using a Bayesian approach was to aid exploration and reduce uncertainty in RL, which could then yield diverse molecules of desirable properties when applied to molecule design. In the remaining part of this paper, DQN and BDQN denote MolDQN and Bayesian MolDQN, respectively. Our work used the Pytorch implementation of MolDQN from https://github.com/aksub99/MolDQN-pytorch.

2.3. Variational Autoencoders (VAEs) for Molecule Generation

VAEs are commonly used to construct generative models by learning the underlying distribution of the data [12], which were used to create novel molecules in this study. Assume $p(x)$ is a distribution of data, and $p(z)$ is a distribution of latent representation of the data. It is beneficial for the latent distribution to be probabilistic, since this introduces noise that helps VAEs to learn more robust representations. The objective in generative learning is to learn $p(x)$ from which we can sample x from, and we know $p(x)=\int p(x|z)p(z)\mathrm{d}z$. Therefore, learning $p(x)$ is equivalent to learning $p(x|z)$ and $p(z)$, where $p(z)$ is a prior for z and $p(x|z)$ is the likelihood of observing x given z. $p(x|z)$ is also called decoder as we can sample z from $p(z)$, and then sample x from $p(x|z)$. The encoder $p(z|x)$ is the posterior given the prior $p(z)$ and the likelihood $p(x|z)$ through Bayes formula. A similar derivation like the one in (1) can lead to the following equation:

$$\log p(x) - KL\left[q(z|x) \,\|\, p(z|x)\right] = E_{q(z|x)}\left[\log p(x|z)\right] - KL\left[q(z|x) \,\|\, p(z)\right] \quad (4)$$

Because KL divergence is always non-negative, we can maximize $\log p(x)$ in (4) by minimizing the loss in (5). By doing so, we actually maximize the lower bound of $\log p(x)$.

$$Loss = KL\left[q(z|x) \,\|\, p(z)\right] - E_{q(z|x)}\left[\log p(x|z)\right] \quad (5)$$

where the first term on the right-hand side is forcing $q(z|x)$ getting close to $p(z)$ so it can serve as regularization to avoid overfitting and the second term measures the quality of reconstruction by the encoder $q(z|x)$ and decoder $p(x|z)$. This loss function suggests that generative models can give us insight into our data. Notice the similarity between the loss functions in (2) and (4). However, the loss in (4) aims to learn the distribution $q(z|x)$ over the latent variable z for each x, and the loss in (2) is to learn the distribution over the weights in the networks. Every x corresponds to a different $q(z|x)$ while in (2) there is only one single distribution over weights. To conclude, VAEs are latent variable models in which we can draw z from aprior $p(z)$ and passed it into a decoder $p(x|z)$. Our work used the VAE implemented in [13], where the encoder $q(z|x)$ and decoder $p(x|z)$ are represented as neural networks.

3. Results

The chemical property studied in this work is QED, which stands for quantitative estimation of drug-likeness and is a measurement based a collection of molecular properties. Its range is (0, 1) and a value closer to 1 means more drug-likeness. Its calculation was carried out by software RDKit. Each molecule was represented by a Morgan fingerprint with radius of 3 and length of 200. Each element of the fingerprint indicates the presence or absence of a particular molecular feature. The experiments reported in this section assumed one episode was made of maximum of 20 steps in the design of a molecule, and one action was taken from one possible action at each step by the DQN agent and one action from multiple possible actions from BDQN. The reward for each action was the QED value for the molecule being generated by the agent. One episode generated one molecule. The atoms that could be added onto the molecule were "C", "O", "N", and the starting molecule was none.

We showed in this section the results of applying Bayesian approach to RL in the generation and optimization of molecules. The molecules generated by BDQN had similar QED values with those from DQN, but more diverse. Furthermore, because of their diversity, these newly created molecules were employed to train a generative model to yield more novel molecules.

3.1. Needs for Better Action Selection

The inspiring motivation for us to use Bayesian networks was that there were varied number of valid actions at each step of generating a molecule by a DQN agent. This uncertainty, therefore, called for better action selection. In the first 500 episodes of one run of DQN, the number of valid actions in each step ranged from 1 to 121 (**Figure 1**).

3.2. Comparison of DQN and BDQN

This section assessed the performance of DQN and BDQN regarding the optimization and diversity of the molecules they created. We ran DQN and BDQN for three experiments with each of 10,000 episodes and the learning curves of

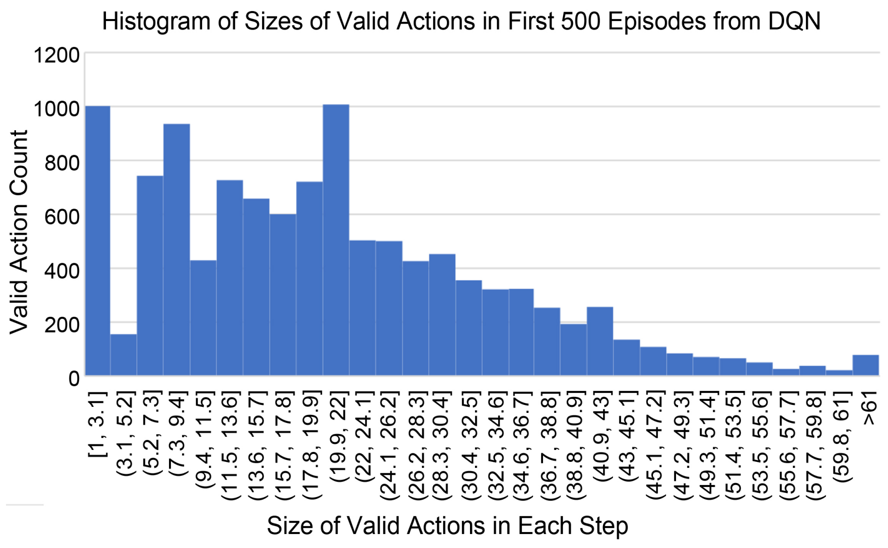

Figure 1. Histogram of size of valid actions in each step in the first 500 episodes from one run of DQN.

averaged rewards were plotted using a simple moving average of size 50, and the shaded areas around the curves were standard deviations of their rewards (Figure 2). The DQN agent obviously collected more rewards than the BDQN agent, therefore, DQN was better in optimization. However, generating diverse molecules is also of great interest in chemistry.

In order to reveal the diversity of the molecules, this time we calculated the unique molecules produced by DQN and BDQN agents respectively (Figure 3).

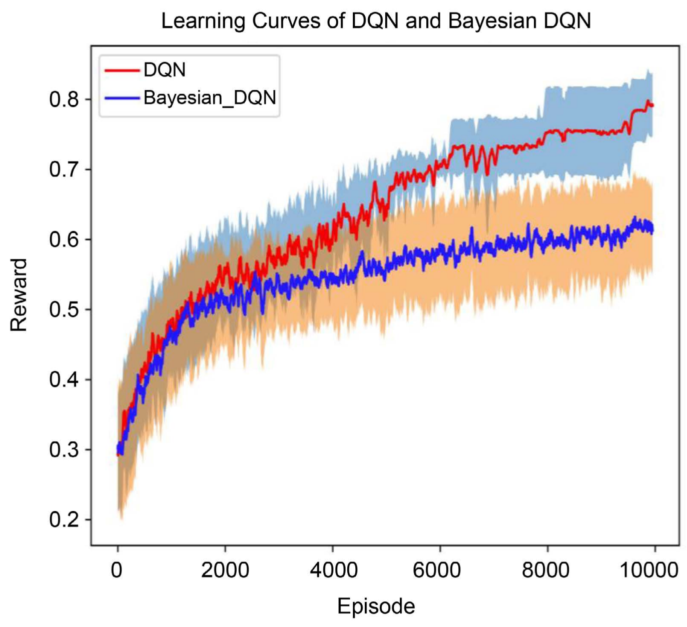

Figure 2. Learning curves of DQN and BDQN where the curves are the average of three rewards and the shaded areas around the curves are the standard deviations of the rewards.

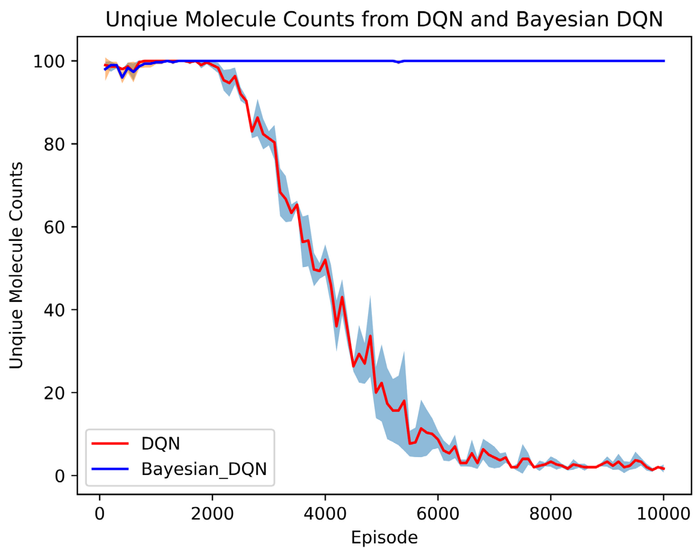

Figure 3. Unique molecule counts from DQN and BDQN where the curves are the average of three counts and the shaded areas around the curves are the standard deviations of the counts.

Out of the 10,000 episodes in each of the three experiments, we counted the unique molecules in a segment of 100 episodes. Recall that each episode generated one molecule. The average counts plotted as curves and their standard deviations were the shaded areas around the curves (Figure 3). It seemed that as the learning took place DQN dropped quickly to a single molecule in each segment of 100 episodes after 8000 episodes, whereas BDQN maintained steady 100 molecules in each segment to the end of 10,000 episodes. What we observed in Figure 3 was mode collapse, where DQN only generated a narrow range of different molecules.

3.3. Advantage of Offering Multiple Possible Actions by BDQN

Bayesian neural networks output multiple predictions instead of a single one from deterministic neural networks. We allowed the BDQN agent to provide several possible actions and then took one of the most votes. Our notation BDQN_1 meant offering one possible action, BDQN_5 meant 5 actions, and BDQN_15 meant 15 actions. Our purpose was to elaborate on the benefits of choosing one action from a collection of possible actions during decision making in the face of uncertainty. We ran three experiments of 10,000 episodes for each, and the average of three rewards were calculated and plotted as curves and the standard deviation of the rewards were the shaded areas (Figure 4). The curves in Figure 4 implied that having multiple actions to choose from can gain more rewards. At the same time, it is of interest to see how the diversity of the molecules was kept as the number of possible actions were increased (from 1 to 5, and then to 15 actions) (Figure 5). As observed from Figure 5, the diversity

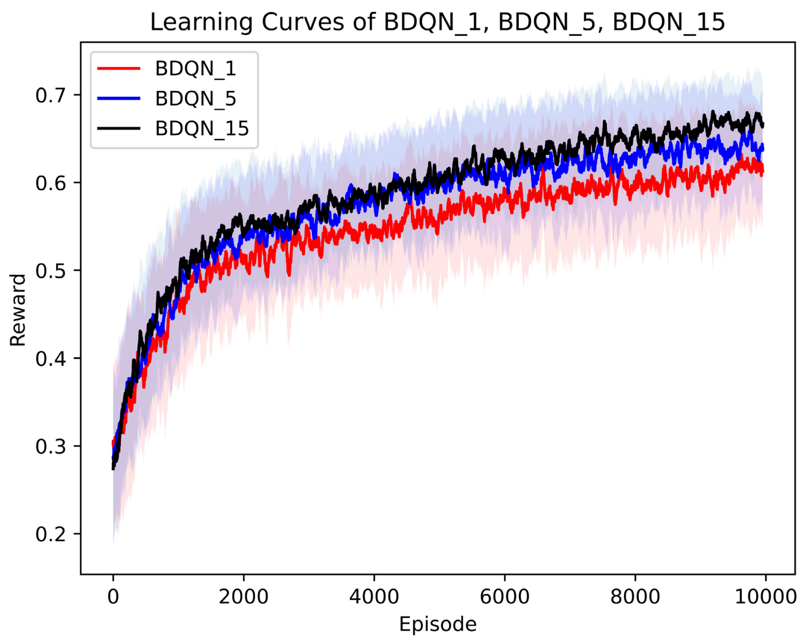

Figure 4. Learning curves of BDQN_1, BDQN_5, and BDQN_15 where the curves are the average of three rewards and the shaded areas around the curves are the standard deviations of the rewards.

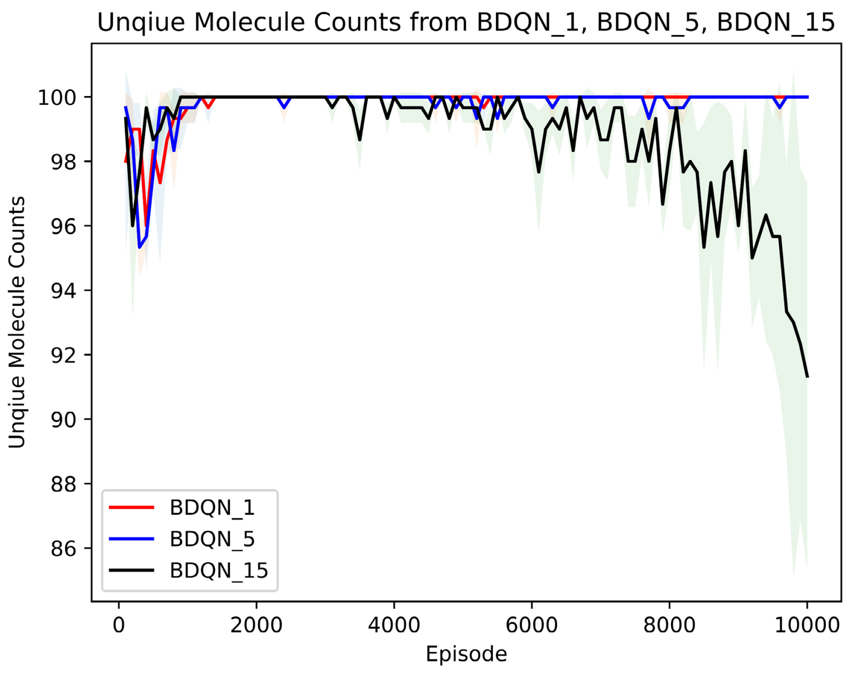

Figure 5. Unique molecule counts of BDQN_1, BDQN_5, and BDQN_15 where the curves are the average of three counts and the shaded areas around the curves are the standard deviations of the counts.

was deteriorated in the case of BDQN_15, while BDQN_5 and BDQN_1 were still very robust. The information in Figure 4 and Figure 5 demonstrated a real play of the dilemma between optimization and diversity: BDQN_15 increased the QED values of molecules but at the expense of their diversity. Therefore, our Bayesian approach provided a means to choose a trade-off between the two.

3.4. Illustration of Learning Process of BDQN_15

This section highlighted the learning process of BDQN_15 as reported in Figure 4 within one episode as well as between two episodes. We first recorded the molecules that were produced after the actions taken at step 19 in episode 526. From 44 valid actions at this step, the actions sampled from the Q function were [1, 1, 1, 1, 22, 18, 1, 22, 1, 23, 1, 22, 1, 1, 26] (index starts from 0 not 1), action 1 had the most votes and therefore it was chosen by the agent. This action resulted in a molecule of the highest QED value (Figure 6) among the 44 molecules (Figure 7). We plotted the graph images of the molecules that were recommended by the Q function in Figures 8-11.

As the learning taking place, QED values of the molecules created were increased. In episode 2811, there were 65 valid actions at step 19, which yielded 65 molecules (Figure 12). The recommended actions were [11, 52, 9, 34, 12, 44, 14, 14, 12, 57, 3, 14, 12, 12, 57], and action 12 had the most votes. This action resulted in a molecule of highest QED value = 0.80, which was much higher than that in episode 526 (Figure 6). The action 14 also had the most votes (tied with action 12), which could result in a molecule of QED value = 0.78.

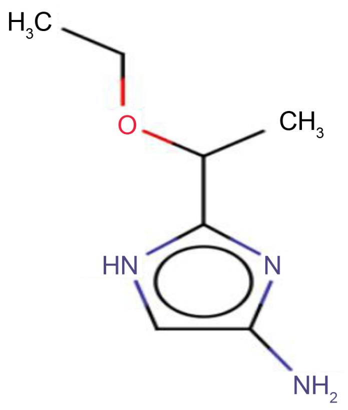

Figure 6. Graph of the molecule that was chosen by BDQN_15 from 15 possible actions, its QED = 0.68 and SMILES = CCOC(C)c1nc(N)c[nH]1.

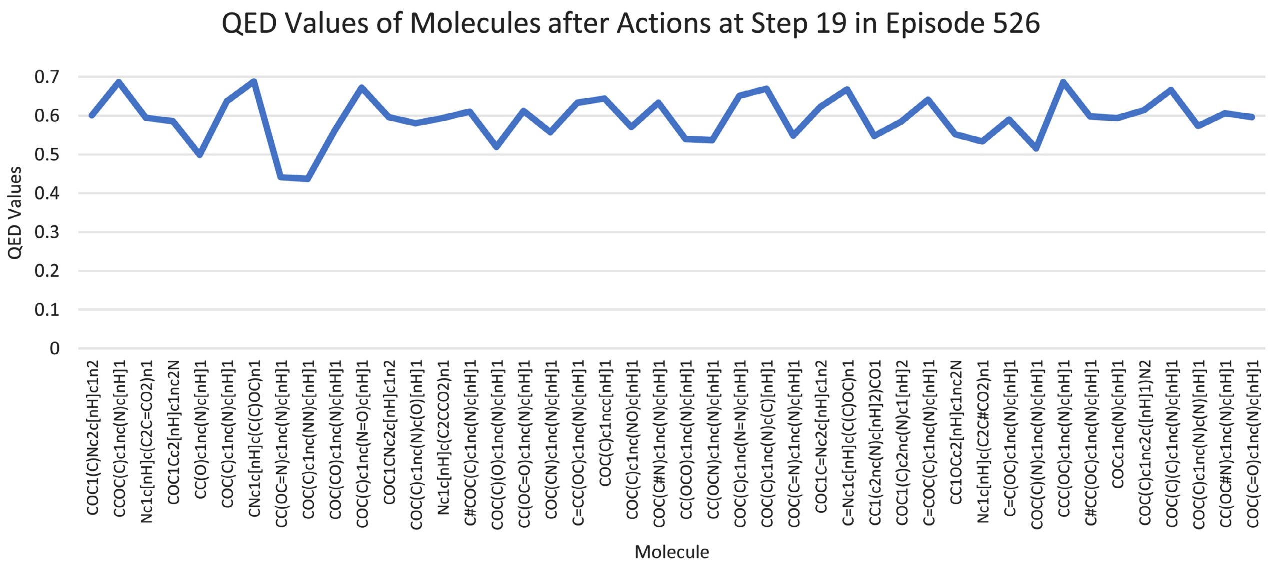

Figure 7. This plot shows the QED values of the molecules after actions at step 19 in episode 526. There are 44 valid actions in total to produce 44 molecules, which are shown on x-axis with their SMILES. The average QED values of 44 mols = 0.59, min = 0.43 (one mol) and max = 0.68 (three mols).

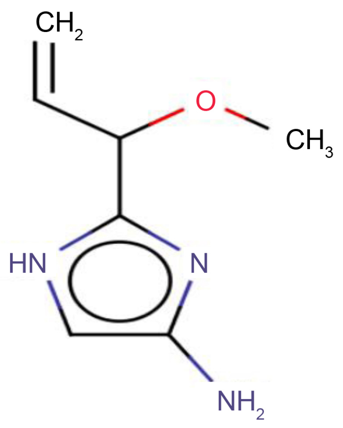

Figure 8. Graph of one of the molecules produced by 15 possible actions, its QED = 0.63 and SMILES = C=CC(OC)c1nc(N)c[nH]1.

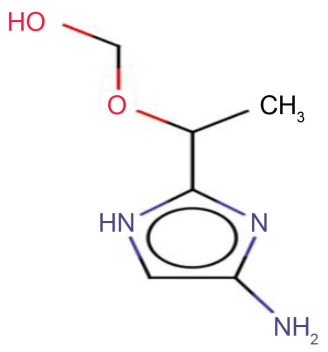

Figure 9. Graph of one of the molecules produced by 15 possible actions, its QED = 0.53 and SMILES = CC(OCO)c1nc(N)c[nH]1.

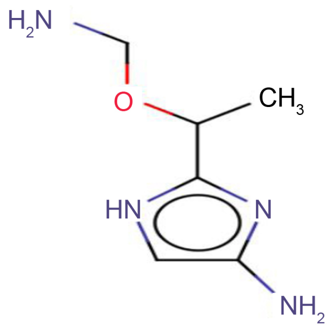

Figure 10. Graph of one of the molecules produced by 15 possible actions, its QED = 0.53 and SMILES = CC(OCN)c1nc(N)c[nH]1.

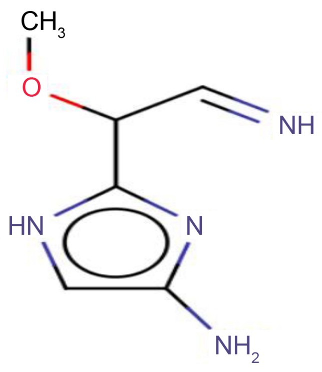

Figure 11. Graph of one of the molecules produced by 15 possible actions, its QED = 0.54 and SMILES = COC(C=N)c1nc(N)c[nH]1.

3.5. Using VAEs to Generate Novel Molecules from the Molecules Created by BDQN_15

Section 3.3 showed that the molecules produced by BDQN enjoyed a great diversity. To further exploit these molecules, we selected 1000 molecules of QED values above 7.0 created by BDQN_15 and used them as a training set to build a VAE as a generative model. After training, this model was utilized to produce 2382 molecules. Some of them belonged to the training set, and we took out the ones that were not part of the training set, which had a total of 1632 new molecules. We considered this set as novel molecules generated by this VAE.

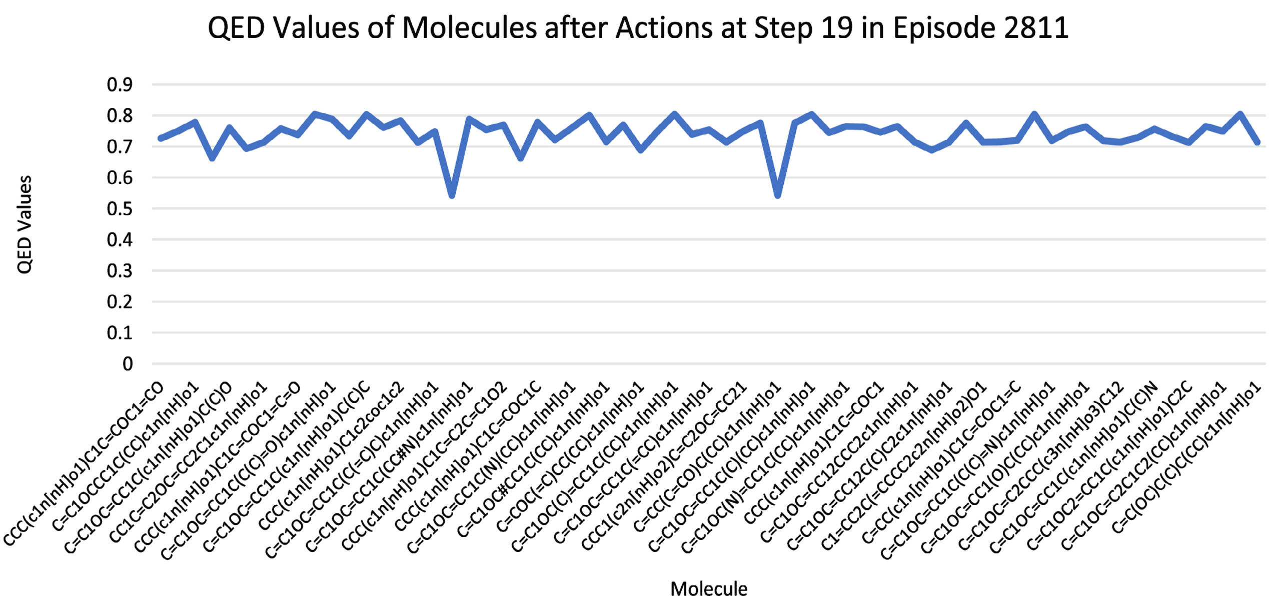

Figure 12. This plot shows the QED values of the molecules after actions at step 19 in episode 2811. There are 65 valid actions in total to produce 65 molecules, which are shown on x-axis with their SMILES. The average QED values of 65 mols = 0.74, min = 0.43 (one mol) and max = 0.68 (three mols).

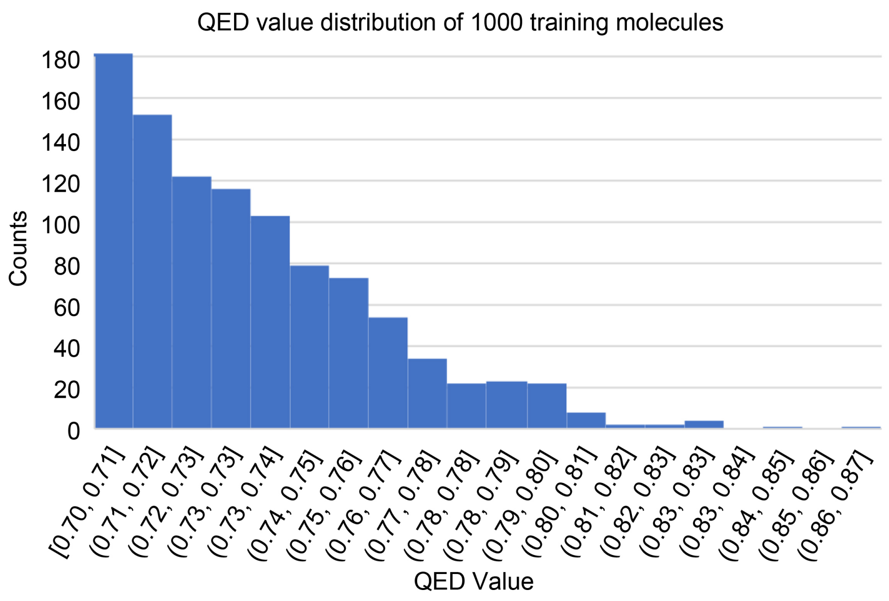

Figure 13. QED value distribution of the 1000 molecules (unique scaffolds = 778) created by BDQN_15 and used as a training set for VAE, their average QED values = 0.73.

To provide more insight into the work of this VAE, we compared the training molecules from BDQN_15 with the VAE generated ones. Because the 1000 molecules were selected to have QED values above 0.7 so their QED distribution was not normal (Figure 13), whereas the distribution of the QED values from the 1632 molecules generated by VAE was normal (Figure 14). It was interesting to see that the length distribution of both sets of molecules were normal and even surprisingly had the same average length of 27.6 (Figure 15, Figure 16).

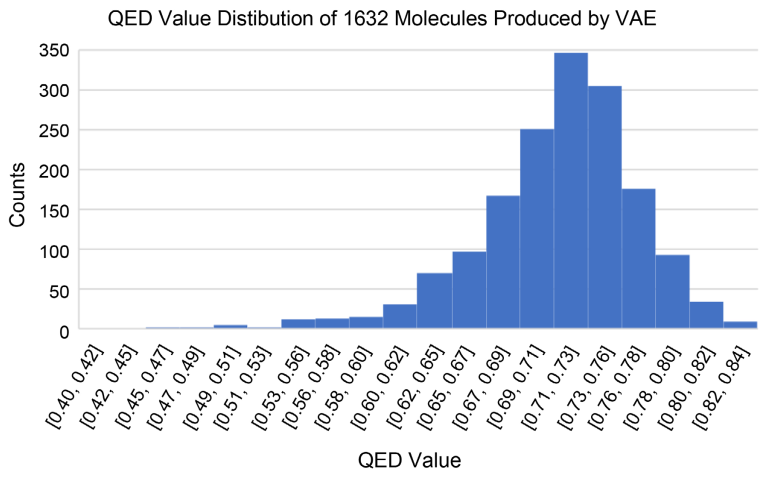

Figure 14. QED value distribution of 1632 molecules (unique scaffolds = 1098) generated by VAE trained with the 1000 molecules produced by BDQN_15,their average QED values = 0.71.

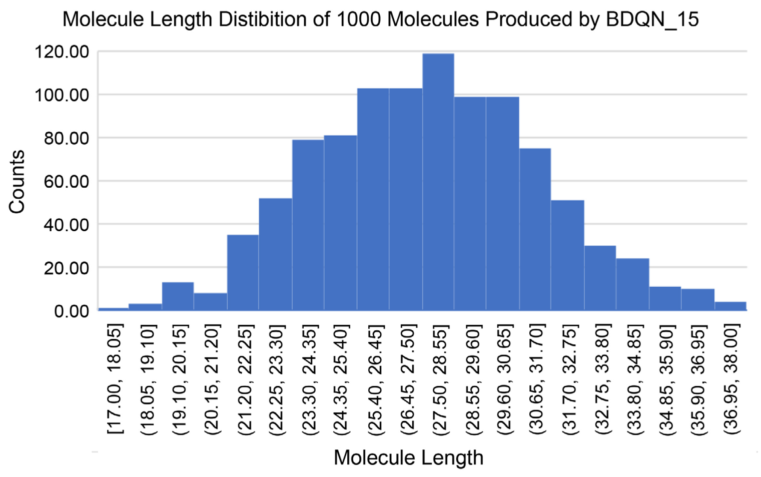

Figure 15. Molecule length distribution of 1000 molecules produced by BDQN_15, their average length = 27.6.

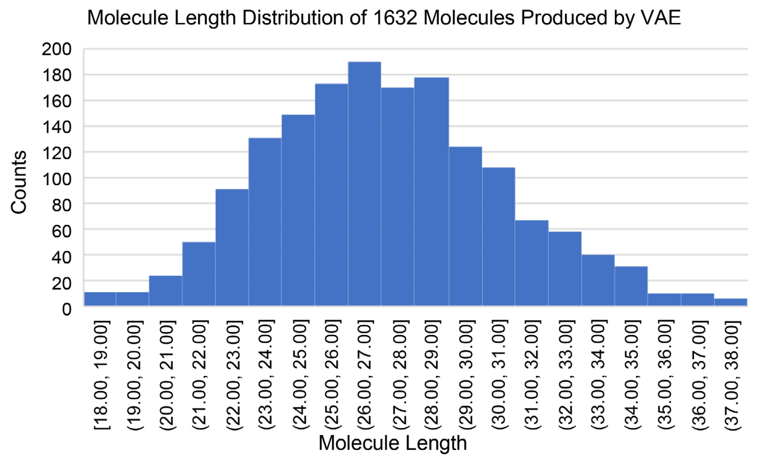

Figure 16. Molecule length distribution of 1632 molecules produced by VAE trained with the 1000 molecules created by BDQN_15, their average length = 27.6.

4. Conclusions

Machine learning has demonstrated its potential to accelerate the process of identifying and designing novel molecules. It has been observed that if a machine learning model is trained with a set of molecules, then it may be difficult to generate novel molecules that are highly dissimilar to those in the training set. RL can be used to create molecules without training datasets; as a result, it can reduce the model bias caused from particular choice of datasets. Discovering novel molecules different from the known ones is another crucial factor for drug devolvement since only by satisfying this condition is it possible to design new drugs.

Uncertainty in machine learning is unavoidable. It can come from data or the learning process. In RL, the uncertainty may originate from the environment or the learning process. Compared to the traditional deterministic neural networks that can only output point estimates, Bayesian neural networks can output any number of sampled predictions. This redundancy of Bayesian predictions is advantageous in the face of uncertainty.

Our findings suggest that Bayesian networks are able to keep the balance between exploration and exploitation in RL, thus performing better than their deterministic counterparts. The additional benefit of using Bayesian networks is that they can yield diverse molecules of desired chemical quality and offer a means to achieve a trade-off between optimization and diversity in molecule design. Because of these advantages, our work has made two leaps: the first leap (0 to 1) is creating drug-like molecules from none by a Bayesian RL agent, and the second leap (1 to n) is producing more novel molecules by a VAE model trained with the molecules generated in the first leap.

Conflicts of Interest

The author declares no conflicts of interest regarding the publication of this paper.

References

[1] Elton, D.C., Boukouvalas, Z., Fugea, M.D. and Chunga, P.W. (2019) Deep Learning for Molecular Design—A Review of the State of the Art. *Molecular Systems Design & Engineering*, **4**, 828-849. https://doi.org/10.1039/C9ME00039A

[2] Patrick Walters, W. and Barzilay, R. (2021) Applications of Deep Learning in Molecule Generation and Molecular Property Prediction. *Accounts of Chemical Research*, **54**, 263-270. https://doi.org/10.1021/acs.accounts.0c00699

[3] Rifaioglu, A.S., Atas, H., Martin, M.J., Cetin-Atalay, R., Atalay, V. and Doğan, T. (2019) Recent Applications of Deep Learning and Machine Intelligence on *in Silico* Drug Discovery: Methods, Tools and Databases. *Briefings Bioinform*, **20**, 1878-1912. https://doi.org/10.1093/bib/bby061

[4] Olivecrona, M., Blaschke, T., Engkvist, O. and Chen, H. (2017) Molecular De-Novo Design through Deep Reinforcement Learning. *Journal of Cheminformatics*, **9**, Article Number: 48. https://doi.org/10.1186/s13321-017-0235-x

[5] Popova, M., Isayev, O. and Tropsha, A. (2018) Deep Reinforcement Learning for De Novo Drug Design. *Science Advances*, **4**, Article ID: 7885. https://doi.org/10.1126/sciadv.aap7885

[6] Jeon, W. and Kim, D. (2020) Autonomous Molecule Generation Using Reinforcement Learning and Docking to Develop Potential Novel Inhibitors. *Scientific Reports*, **10**, Article Number: 22104. https://doi.org/10.1038/s41598-020-78537-2

[7] Ståhl, N., Falkman, G., Karlsson, A., Mathiason, G. and Boström, J. (2019) Deep Reinforcement Learning for Multiparameter Optimization in De Novo Drug Design. *Journal of Chemical Information and Modeling*, **59**, 3166-3176. https://doi.org/10.1021/acs.jcim.9b00325

[8] Zhou, Z., Kearnes, S., Li, L., Zare, R.N. and Riley, P. (2019) Optimization of Molecules via Deep Reinforcement Learning. *Scientific Reports*, **9**, Article Number: 10752. https://doi.org/10.1038/s41598-019-47148-x

[9] Blundell, C., Cornebise, J., Kavukcuoglu, K. and Wierstra, D. (2015) Weight Uncertainty in Neural Networks, ICML'15: *Proceedings of the* 32*nd International Conference on International Conference on Machine Learning*, **37**, 1613-1622.

[10] Hu, W. (2021) Exploring Local Chemical Space in De Novo Molecular Generation Using Multi-Agent Deep Reinforcement Learning. *Natural Science*, **13**, 412-424. https://doi.org/10.4236/ns.2021.139034

[11] Hu, W. (2021) Inverse Molecule Design with Invertible Neural Networks as Generative Models. *Journal of Biomedical Science and Engineering*, **14**, 305-315. https://doi.org/10.4236/jbise.2021.147026

[12] Kingma, D.P. and Welling, M. (2014) Auto-Encoding Variational Bayes. *ICLR* 2014, 14-16 April 2014, Banff, Canada.

[13] Polykovskiy, D., Zhebrak, A., Sanchez-Lengeling, B., Golovanov, S., Tatanov, O., Belyaev, S., Kurbanov, R., Artamonov, A., Aladinskiy, V., Veselov, M., Kadurin, A., Johansson, S., Chen, H.M., Nikolenko, S., Aspuru-Guzik, A. and Zhavoronkov, A. (2020) Molecular Sets (MOSES): A Benchmarking Platform for Molecular Generation Models. Frontiers in Pharmacology. https://doi.org/10.3389/fphar.2020.565644