ELSEVIER

DRUG DISCOVERY TODAY TECHNOLOGIES

Drug Discovery Today: Technologies

Vol. 32, No. 2019

Editors-in-Chief
Kelvin Lam – Simplex Pharma Advisors, Inc., Boston, MA, USA
Henk Timmerman – Vrije Universiteit, The Netherlands

On failure modes in molecule generation and optimization

Philipp Renz[1], Dries Van Rompaey[2], Jörg Kurt Wegner[2], Sepp Hochreiter[1], Günter Klambauer[1,*]

[1]LIT AI Lab & Institute for Machine Learning, Johannes Kepler University Linz, Altenberger Strasse 69, A-4040 Linz, Austria
[2]High Dimensional Biology and Discovery Data Sciences, Janssen Research & Development, Janssen Pharmaceutica N.V., Turnhoutseweg 30, Beerse B-2340, Belgium

There has been a wave of generative models for molecules triggered by advances in the field of Deep Learning. These generative models are often used to optimize chemical compounds towards particular properties or a desired biological activity. The evaluation of generative models remains challenging and suggested performance metrics or scoring functions often do not cover all relevant aspects of drug design projects. In this work, we highlight some unintended failure modes in molecular generation and optimization and how these evade detection by current performance metrics.

Section editor: Johannes Kirchmair – University of Vienna, Department of Pharmaceutical Chemistry, Althanstrasse 14, 1090 Vienna, Austria.

Introduction

In recent years, there has been increased interest in generative models for molecules in drug discovery [1–4]. In the area of de novo molecular design [5], those models are used to generate molecules with desired properties from scratch. This is often seen as an alternative to virtual screening, which is limited by the size of the libraries that can be searched in practice. Instead of screening existing libraries, generative models can be used to directly create focused libraries for given targets [1]. They can also prove useful in lead optimization, where the aim is to identify a small number of molecules with optimized profiles. Such a profile can encompass a large number of dimensions, e.g. potency, metabolic stability, physicochemical parameters or permeability. This oftentimes arduous process may be accelerated through generative models, which are capable of directly optimizing molecules towards a given profile.

Novel generative models for molecules are mostly based on machine learning (ML), in particular Deep Learning [6–9], and complemented more classical approaches [10–12]. The main advantage of machine learning methods over classical approaches is their potential ability to learn what viable compounds look like from data. To this end, machine learn-

*Corresponding author.

 https://doi.org/10.1016/j.ddtec.2020.09.003

ing methods use a training set of molecules from which they try to learn the underlying distribution of the data. The learned distribution is then used to generate new molecules, where methods differ in the concrete way how the generation process is employed.

The first wave of Deep Learning methods for molecule generation [1,2] used the SMILES [13] representation of compounds in combination with recurrent neural networks (RNNs) [14,15], which are well suited to process sequences. More recent versions of these models have moved towards more robust line notations of chemical structure, such as DeepSmiles [16] or SELFIES [17]. Another range of models directly generate molecular graphs, using graph neural networks [18]. Apart from the utilized representation, models differ in the training procedure and model architecture. For a more detailed overview on current approaches, see [3,19].

There are two main use cases of generative models for molecules which are *distribution-learning* and *goal-directed generation* [20]. Distribution-learning is concerned with the task of generating molecules that resemble a given set of molecules in distribution. In goal-directed generation, the aim is to produce molecules with some desired property profile, for example physical/chemical properties, bioactivities or a combination thereof.

Distribution-learning models can be used to generate molecule libraries or can serve as a starting points for goal-directed generation. Evaluation of these models can be problematic as demonstrated by recent efforts to establish benchmarks for their evaluation [20–22]. These benchmarks often comprise heuristics that try to capture desired properties of the generated molecules; for example, measuring whether the distributions of certain molecular properties match. Many of these heuristics can however be tricked by naive models as we show in Section ''Failure mechanisms in distribution-learning''.

Goal-directed generative models for molecules are trained to produce molecules with some desired property profile, for example physical or chemical properties, bioactivities or a combination thereof. Many of such models were tested on their ability to generate compounds with a high penalized logP score [23–26]. It should however be noted that it is trivial to achieve state-of-the-art results by generating long saturated hydrocarbon chains. The GuacaMol package [20] was an important step in improving evaluation. We agree with the authors, however, that the benchmarks are easy to solve and that the quality of generated compounds is insufficiently addressed.

A goal-directed molecule generator is typically paired with a scoring function, which reflects how closely a molecule resembles the desired profile. Unfortunately, finding a good scoring function is not trivial for many tasks, since this function should quantify biological effects of a molecule, together with synthetic feasibility and drug-likeness both of which are hard to quantify [27,28]. Most scoring functions used in practice do not include the intuitive constraints that practitioners have. The scoring function might therefore be optimized by a molecule generator in a way that was not intended. This problem is intensified when using machine learning models as scoring functions, which we show in Section ''Failure mechanisms in goal-directed generation''.

In spite of the deluge of publications detailing new approaches towards the generation of molecules, wet-lab validations of generative models remain scarce. Merk et al. [29,30] showed that molecules generated using transfer learning [1] showed activity in vitro. Zhavoronkov et al. [31] identified a DDR1 kinase inhibitor using generative Deep Learning models. What is not apparent from these studies is how ''inventive'' the proposed methods are and how they would fare in comparison to an approach in which chemists design compound libraries using more traditional methods. As the field matures, we hope that such comparisons will become more prevalent.

In this work, we aim at highlighting current weak points in evaluating generative models for molecules. We cover both distribution-learning and goal-directed generation with a focus on the latter.

Failure mechanisms in distribution-learning

First, we review problems of existing metrics for distribution-learning and demonstrate that many of them can be easily tricked by a model that just minimally edits the molecules in the training set.

Examples of distribution-learning models for molecules include auto-regressive approaches [1,32], variational auto-encoders [2,23,33], generative adversarial networks [34,35], and adversarial auto-encoders [36,37]. Previous studies in distribution-learning, often used only a handful of heuristics, such as visual inspection, *novelty* and *validity*, to measure performance. Checking the *novelty* of generated compounds is the most commonly used way to detect copying of molecules in the training set. As this merely checks for exact compound matches in the training set this metric is relatively insensitive. The *validity* metric tests if a generated molecule is syntactically valid. Additionally, *uniqueness* measures if molecules appear multiple times in a set of generated molecules. To improve evaluation, Preuer et al. [21] introduced the Frechet Chemnet Distance (FCD), which has been shown to capture many such heuristics in a single score. The FCD is also included in more comprehensive benchmarking suites [20,22]. While molecules generated by distribution-learning models can make sense intuitively, it is hard to objectively quantify if the model actually captured the existing patterns in the data distribution or if it just largely copies training inputs.

Overall, evaluation of generative models is not straightforward and required the development of new metrics [20–22].

Table 1. Comparison of the AddCarbon model to the baselines in [20], from where the table was adapted. Column names represent molecule generators. Random sampling ``RS'' randomly picks a molecule from the training set.

Benchmark	RS	LSTM	GraphMCTS	AAE	ORGAN	VAE	AddCarbon
Validity	1.000	0.959	**1.000**	0.822	0.379	0.870	**1.000**
Uniqueness	0.997	**1.000**	**1.000**	**1.000**	0.841	0.999	0.999
Novelty	0.000	0.912	0.994	0.998	0.687	0.974	**1.000**
KL divergence	0.998	**0.991**	0.522	0.886	0.267	0.982	0.982
FCD	0.929	**0.913**	0.015	0.529	0.000	0.863	0.871

However, these metrics do not detect if methods reproduce the training data with minimal changes, which we term the *copy problem*.

To illustrate this *copy problem*, we show how a trivial model, we call *AddCarbon model*, can trick most of the distribution learning metrics in [20]. Our model samples a ''new'' molecule by first taking a random molecule from the training set. Then a carbon atom is inserted at some random spot in its SMILES representation. If this yields a syntactically valid SMILES and a molecule not already in the training set it is returned as a new random sample. If the insertion of the carbon atom leads to an invalid SMILES string, other insert positions are tried. If none of the positions work another molecule from the training set is drawn and the steps are repeated until success. For a complete description of the model see Section S1.1.

We evaluate this model using the GuacaMol distribution-learning benchmark [20]. The AddCarbon model obtains near perfect benchmarking scores and outperforms many complex generative models (see Table 1). By construction it has a perfect novelty and validity of 100%, while also having an almost perfect uniqueness, and a very high Kullback–Leibler (KL) divergence metric.

The only metric for which we could not easily obtain a competitive score was the FCD. This was surprising as the FCD is based on the SMILES representation which in our naive model was enforced to be similar to those in the training set, specifically in order to exploit the FCD. It is worth noting that using this simple model, we could beat all of the baselines except the LSTM model. The fact, that the simple AddCarbon model is useless in practice and still obtains good scores, casts some doubt if currently used metrics are sufficient to estimate performance.

Our experiments show that a more detailed quantification of novelty would be highly desirable. The currently used metrics do not allow to provide insights whether the molecule generators act in a similar way as our AddCarbon model. What is also worth noting is that many of the methods performing best in the GuacaMol [20] and Moses [22] benchmarks are likelihood-based models [1,2,33]. For these performance could be evaluated using the likelihood on a hold-out test set, similar to natural language processing. We argue that future studies should report this metric if applicable.

Failure mechanisms in goal-directed generation

Background

In this section, we inspect possible failure modes of goal-directed molecule generators. We argue that a core problem in this setting is the definition a scoring function that includes *all* of the desired properties a molecule should have in a robust way. We show that this problem is exacerbated when predictions of a machine learning model are used as part of the score.

Goal-directed generation focuses on the task of finding molecules that maximize some desired scoring function, which has to capture the requirements of the task at hand. This task in itself can be challenging as it is often hard to condense complex chemical properties into a single number and is made even harder given that it is challenging to encode the extensive background knowledge of medicinal chemists into an exhaustive list of rules [20].

Because of these difficulties scoring functions might not reflect all the requirements a practitioner might have. This incompleteness makes it possible that the scores are optimized in a manner not intended by the practitioner. While a task can be considered solved from an optimization perspective as soon as high scoring molecules are generated, those resulting molecules might not be useful. For example the generated molecules shown in Fig. 1 have high scores but contain unstable, synthetically infeasible, or highly uncommon (see Table S2) substructures and the molecules generated in [20] often do not pass the used quality filters.

In principle the scoring function could have been adapted in these tasks in order to avoid these undesired properties. However, lists of undesired properties are likely to be incomplete [20]. In practical applications[1] we have found generative models to be highly adept at exploiting this incompleteness and generating surprising solutions which are numerically superior but of little use. This often necessitates several iterations between generating molecules and refining the scoring function to account for previously unexpected behavior from the generator, before arriving at molecules which could be useful for drug discovery projects. We have observed that this behavior can even intensify as additional machine learning models are included in the scoring function.

[1] Unpublished, in-house data.

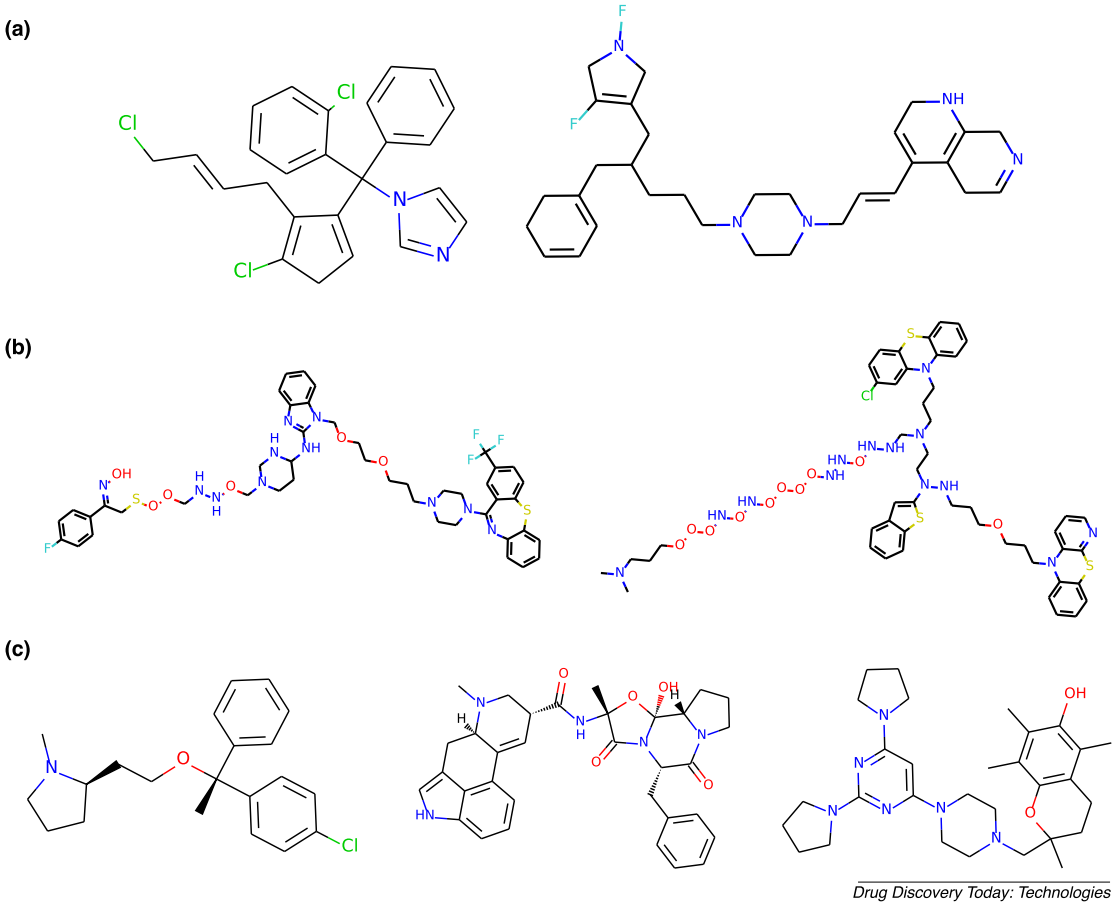

Figure 1. High scoring compounds generated for the DRD2 task described below, with actives from the training set for comparison. The generated compounds show unwanted substructures. (a) Compounds created by a graph-based genetic algorithm. The left compound contains a reactive diene. The right one contains reactive nitrogen-fluorine [38], imine and diene moieties. (b) Compounds generated by a SMILES-based LSTM. The compounds shown contain long hetero-atom chains which are unstable and synthetically infeasible. (c) Reference compounds from the training set.

This phenomenon of optimization of a score in unexpected ways, has been observed in other applications [39]. In one illustrative setting, the aim was to develop a body capable of locomotion, but the optimization procedure instead discovered the simpler solution of a tall body falling over, which also satisfied the specified scoring function.

For many tasks in drug discovery, exact scoring functions are not available. While some properties like molecular mass can be calculated accurately given a compound, this is not the case for more complex properties like bioactivities. These instead are often approximated by fitting machine learning models on experimental data [1,4,40]. Such models can then be used as a scoring function or a component of it. It has been shown, however, that guiding generation by the output of a machine learning model is not a good strategy to generate meaningful samples [41]. Samples generated in such a way have been shown to exploit features unique to the exact models they were optimized for [42]. This is problematic as the generated samples should exhibit the true desired characteristics and not artifacts of a learned model. We call these effects *model specific biases*.

Machine learning models also exhibit biases on the exact data they were trained on. It is often the case that models have near perfect predictive performance on the training data, while performance on hold-out data is usually lower. This is achieved by making predictions based on spurious patterns that can be used to link samples to their labels, but are not true explanatory factors of the output. Consequently, when the outputs of a learned model are optimized these spurious patterns might be recovered.

Additionally we observed that actives/inactives from the training set receive higher/lower prediction scores than those in the test set, as shown in Fig. S2. This might bias the generation towards compounds similar to the training set actives as scores are skewed in favour of those. We refer to these problems as *data specific biases*.

Experimental setup

We now describe our experimental design to illustrate these failure modes. Our goal is to generate molecules that are active against some biological target. We selected three well-known targets; Janus kinase 2 (JAK2), epidermal growth

factor receptor (EGFR) and dopamine receptor D_2 (DRD2) for which we downloaded data from ChEMBL [43]. We preprocessed the data to obtain binary classification tasks. Information about the data is displayed in Table S1. For the exact preprocessing procedure, see Section S1.2. In contrast to previous studies, we split the data into two halves which we will call *split 1/2*, before we obtain a scoring function. The ratio of actives to inactives in both splits is kept equal. Fig. S1 shows a distribution of nearest neighbour similarities between the two splits.

Our aim is to train three bioactivity models with relatively equivalent predictive performance, of which one will be used as a scoring function for molecular optimization and the other two will serve to evaluate performance. To this end, we train three classifiers. The first classifier, trained on split 1, will be used as a scoring function for optimizing molecules and we will refer to its output as *optimization score* (OS). The second classifier is also trained on split 1 using a different random seed than the OS model. This classifier is used to control if the scores of optimized molecules generalize across two classifiers trained on the same data and quantifies model specific biases. We refer to its outputs as *model control scores* (MCS). The third classifier, trained on split 2 is used to control if the optimized molecules also score highly when evaluated with a model trained on different samples and quantifies data specific biases. We will refer to the output of this classifier as *data control score* (DCS).

We use a random forest classifier [44] as implemented in scikit-learn [45] as a classification algorithm. The ratio of trees predicting that a compound is active is used as a scoring function. As features we use binary folded ECFP fingerprints [46] of size 1024 and radius 2, which were calculated using rdkit [47]. We obtained three classifiers for each of the three targets, JAK2, EGFR, and DRD2, with similar predictive performance (see Table S1) suitable for goal-directed generation. For each model the performance was evaluated on the split that was not used for training. There is only one performance value for all three classifiers, because in expectation it does not depend on the used data split and random seed.

We then trained a goal-directed generation algorithm to produce molecules with high optimization scores. We employed three different molecule generators. The first is a graph-based genetic algorithm (GA) [12]. GA optimizes molecules by iteratively applying random mutations and crossovers to the molecules in a population and keeping the best ones in each generation. The starting population were random molecules from the distribution-learning training set defined in [20], which is in turn a random subset of the compounds in ChEMBL [43].

The second optimization algorithm we use is called SMILES-LSTM (LSTM) [1]. This method generates molecules by predicting the probabilities of the next character in SMILES strings based on the previous ones. The molecules are optimized using a hill climbing algorithm which can be understood as iteratively sampling molecules, keeping the best ones and fine-tuning the model on these high scoring molecules. We used the pretrained model provided by [20] to initialize the generator. This model was trained on the same data set, from which the starting population for GA is drawn.

The third algorithm we employed is the method proposed in [48]. This method relies on the continuous latent space of a SMILES based sequence-to-sequence model introduced in [49]. Compounds are optimized in this latent space using particle swarm optimization (PS). Each particle in the swarm represents a position in the latent space. The position of a particle is then iteratively updated into the direction of its best encountered position and the best position over all particles. The starting compounds were drawn from the same set as for GA.

We included GA and LSTM as they are the two top-performing molecule generators according to [20]. PS was included as it showed promising performance when it was proposed and because it is different to the other methods in that it optimizes compounds in a latent space. We run each optimization algorithm ten times for each data set.

The suggested experimental setup with optimization score and control scores, allows us to get insights into how a generative model optimizes the score and whether it is sensitive to mentioned biases in the bioactivity model. The setup with an optimization and control score mirrors the use of a training and test set in supervised learning methods. In supervised learning the goal is to obtain good performance on a test set that was not used during optimization, which in our case corresponds to the control scores.

Results

First we investigate how the optimization score (OS) and data control score (DCS) evolve during optimization for the EGFR task (see Fig. 2). For GA and PS we show the scores of molecules in the population at different iterations, while for the LSTM we sample molecules in each iteration. Starting from random ChEMBL compounds the optimization process increases the OS more dominantly than the DCS. This behaviour is also present for the DRD2 and JAK2 data sets, for which the corresponding plots can be found in Fig. S3. For all of the tasks considered here, there is a mismatch between OS and DCS, which shows that the optimization procedure suffers from model and/or data specific biases.

The optimized molecules seem to populate the same region as the split 1 actives that were used for training the OS, as shown by their migration towards the contours in the plot. This suggests that the molecular optimizer finds compounds similar to the actives that were used to obtain the optimization scoring function. This is in fact the case (see Fig. S10), as optimized molecules have a more similar neighbour in split 1 than in split 2 on average, as measured by ECFP4 Tanimoto

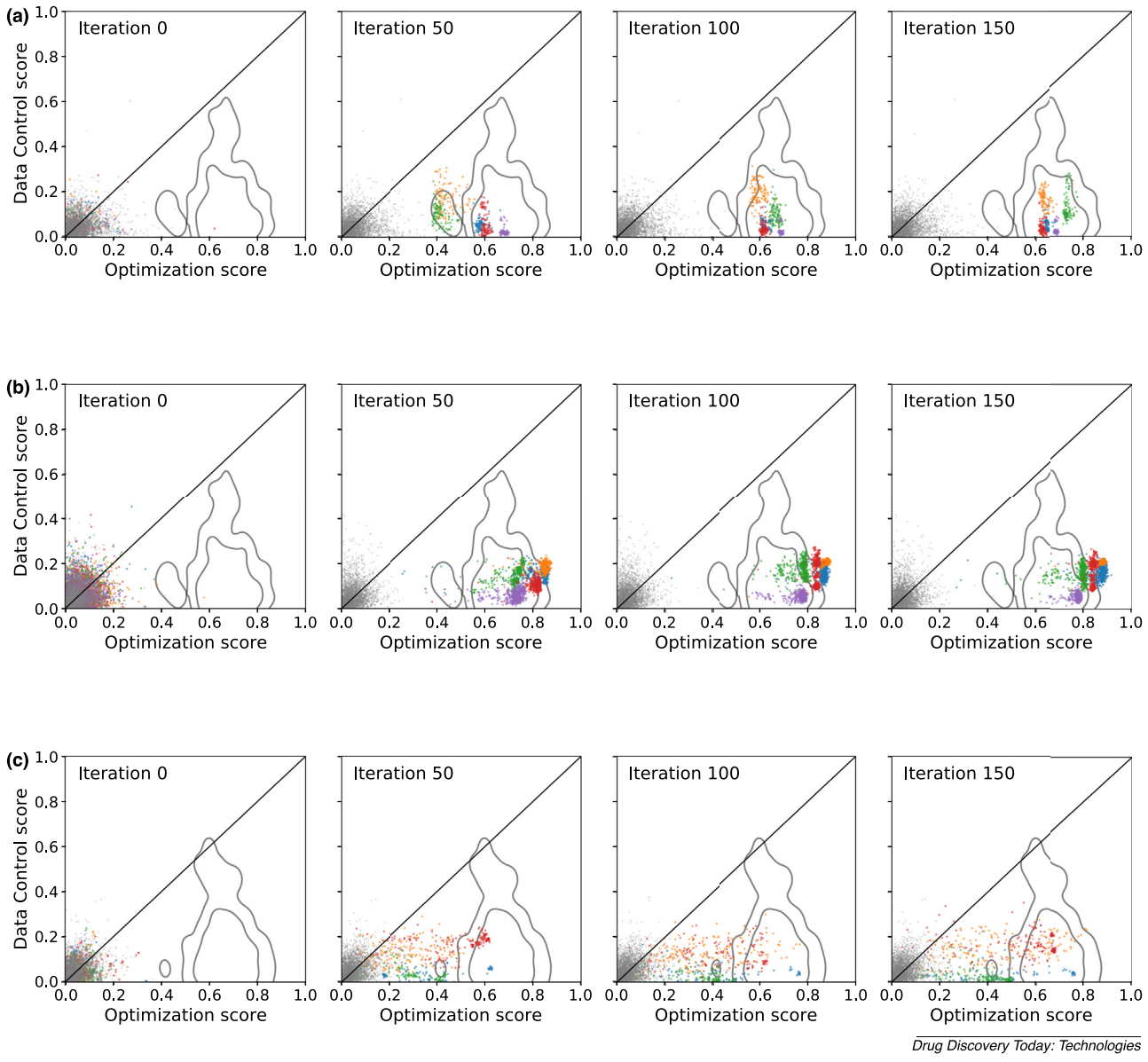

Figure 2. Scatter plots of OS vs. DCS on molecules during the course of training for the EGFR task and optimizer (a) GA, (b) LSTM and (c) PS. Each point represents a compound. Different colors correspond to different runs. Grey points are random compounds from ChEMBL. The contour lines approximately show the region where split 1 actives lie. During the course of training the optimized molecules move towards the region of split 1 actives, which were used to train the OS.

similarity. This shows that at least some of the difference between OS and DCS can be explained by data specific biases.

Next we investigate to which extent the difference between OS and DCS arise from model and data specific biases. Fig. 3 shows how all three scores develop during the course of optimization. We observe that the OS and MCS diverge in the course of training. This suggests that optimization exploits features unique to the OS to obtain improvements that do not generalize to the MCS, despite being trained on exactly the same data. It can also be observed that the larger part of the difference between OC and DCS is due to data specific biases. Fig. 3 also shows that while the OS grows monotonically, the control scores sometimes stagnate or even decrease. This suggests that optimization should be stopped early once the control scores cease to increase.

Next we more closely investigate the optimized molecules. Figs. 1, S4, S5, and S6 show some examples of optimized molecules. Similar to [50], Fig. S7 shows logP [51] and molecular weight (MW) values for the generated molecules and known actives and inactives at the start and end of optimization. It can be observed that the optimized molecules cover different parts of chemical space than the known actives. Fig. S8 shows a t-SNE embedding based on ECFP4 Tanimoto-distance at the start and end of optimization. At the start of optimization the generated molecules overlap well with the known actives and inactives for DRD2 and EGFR but not

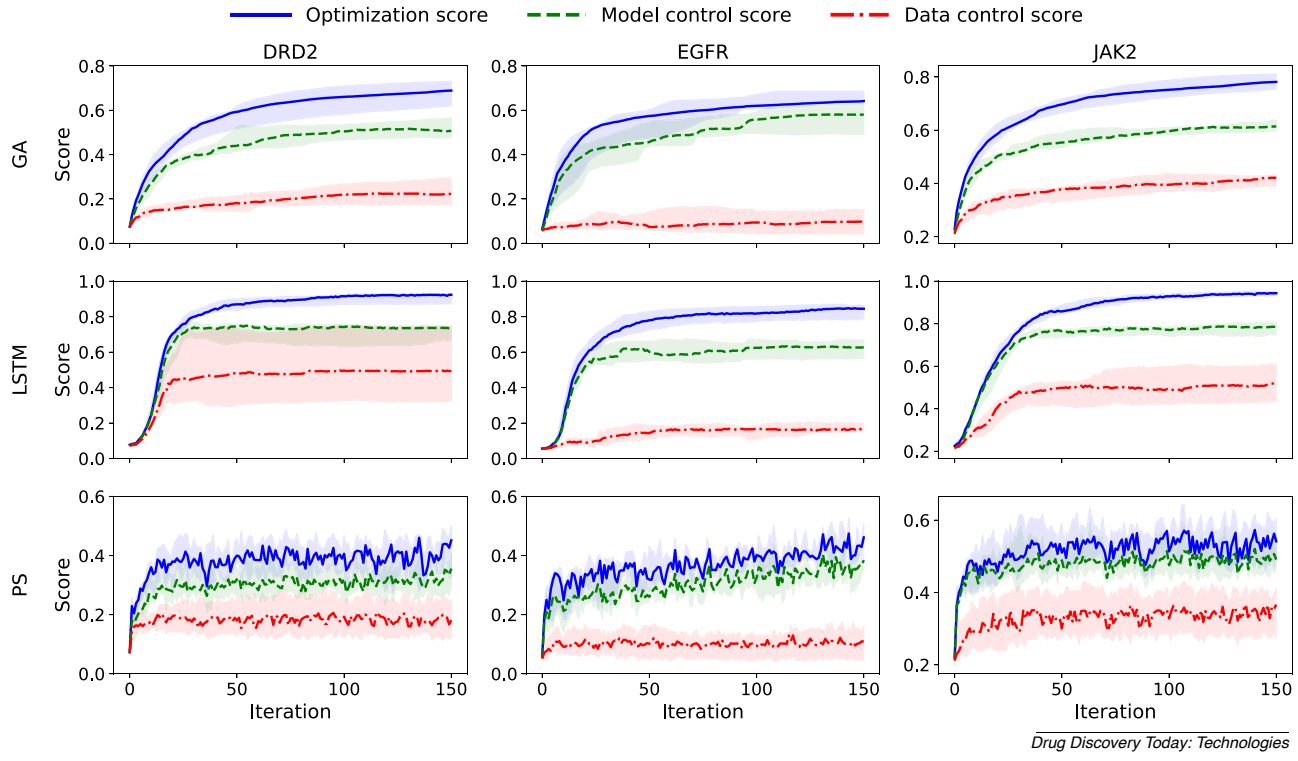

Figure 3. Scores during optimization. For each curve we first took the mean of the scores for each run. The bold line corresponds to the median over those means while the shaded areas correspond to the interquartile range.

for JAK2. At the end of optimization the generated molecules populate different parts of chemical space than the known actives and inactives for all tasks. Compounds generated within the same run form clusters.

Additionally, we analyze the development of chemical properties, concretely logP, MW, SMILES length, diversity and distance to nearest split 1 active/inactive of the generated molecules (see Fig. S9) during optimization. We observe that the LSTM optimizer shows a bias towards compounds with large MW and for low diversity when compared to the other optimizers. A bias towards high logP for the DRD2 and EGFR tasks can also be seen for LSTM. See Section S2.4 for details.

The conclusions drawn from our experiments are limited by the design choices we made. We used the LSTM and GA molecular optimizers as they performed well in GuacaMol [20] and PS as it showed promising results in [48]. It may however be the case that other algorithms do not exhibit the same deficient behaviour shown above. We chose ECFP fingerprints as they have been shown to often perform well [52]. Random Forest classifiers were chosen as they can be trained relatively robustly and also have acceptable performance. Results may vary when using other data sets. More extensive experimentation, however, is the scope of future studies.

Discussion and conclusion

In this work, we pointed out possible failure modes in molecular generation tasks that are not captured by current evaluation techniques. Our conclusion regarding distribution-learning is that extremely simple and practically useless models can get near perfect scores on many metrics currently in use. We observed that many of the best technologies according to benchmark studies [20,22] are likelihood based models and we argue that future comparisons should include test set likelihoods. This would correspond to a comprehensive metric that could improve upon previously suggested measures which can be ''tricked''.

We discussed possible failure modes in goal-directed learning, with an emphasis on problems that are introduced when using machine learning models as scoring functions. While many technological tools and techniques are suitable to optimize a scoring function, the central challenge lies in the scoring function itself. We showed that optimization exploits model-based scoring functions using their data- and model-specific biases. Given that the original aim of generative models for de novo design is to explore *all* of chemical space, biases towards the training data may indicate failure in this respect. We believe that control scores can help in detecting such biases and guide practitioners to potentially more meaningful results. Future experiments should investi-

gate more closely the influence that the choice of classifiers and molecular optimization methods have on the gap between optimization and control scores.

Apart from the issues highlighted in this study, we consider synthetic accessibility [53] as a major challenge for practical adaption, as even the best in silico scores remain fruitless if the suggested compounds cannot be synthesized. Generative models that respect synthesis rules could even be more robust against the overfitting problem that we elaborated on in this work. Overall, considerations regarding synthesis efforts should receive more attention, as costly designs interfere with reduction to practice in the lab.

Availability

We publish data, code and results at https://github.com/ml-jku/mgenerators-failure-modes.

Conflict of interest

None declared. JKW and DVR are employed at Janssen Pharmaceutica N.V.

Acknowledgment

This work was supported by Flanders Innovation & Entrepreneurship (VLAIO) with the project grant HBC.2018.2287 (MaDeSMart).

Appendix A. Supplementary data

Supplementary data associated with this article can be found, in the online version, at https://doi.org/10.1016/j.ddtec.2020.09.003.

References

[1] Segler MHS, Kogej T, Tyrchan C, Waller MP. Generating focused molecule libraries for drug discovery with recurrent neural networks. ACS Central Sci 2018;4(1):120–31. http://dx.doi.org/10.1021/acscentsci.7b00512.

[2] Gómez-Bombarelli R, Wei JN, Duvenaud D, Hernández-Lobato JM, Sánchez-Lengeling B, Sheberla D, et al. Automatic chemical design using a data-driven continuous representation of molecules. ACS Central Sci 2018;4(2):268–76. http://dx.doi.org/10.1021/acscentsci.7b00572.

[3] Sanchez-Lengeling B, Aspuru-Guzik A. Inverse molecular design using machine learning: generative models for matter engineering. Science 2018;361(6400):360–5. http://dx.doi.org/10.1126/science.aat2663.

[4] Olivecrona M, Blaschke T, Engkvist O, Chen H. Molecular de-novo design through deep reinforcement learning. J Cheminform 2017;9(1):48. http://dx.doi.org/10.1186/s13321017-0235-x.

[5] Schneider G. De novo molecular design. John Wiley & Sons, Ltd; 2013. http://dx.doi.org/10.1002/9783527677016.

[6] Schmidhuber J. Deep learning in neural networks: an overview. Neural Netw 2015;61:85–117. http://dx.doi.org/10.1016/j.neunet.2014.09.003.

[7] LeCun Y, Bengio Y, Hinton G. Deep learning. Nature 2015;521(7553):436–44. http://dx.doi.org/10.1038/nature14539.

[8] Klambauer G, Hochreiter S, Rarey M. Machine learning in drug discovery. J Chem Inf Model 2019;59(3):945–6. http://dx.doi.org/10.1021/acs.jcim.9b00136.

[9] Hochreiter S, Klambauer G, Rarey M. Machine learning in drug discovery. J Chem Inf Model 2018;58(9):1723–4. http://dx.doi.org/10.1021/acs.jcim.8b00478.

[10] Venkatasubramanian V, Chan K, Caruthers JM. Computer-aided molecular design using genetic algorithms. Comput Chem Eng 1994;18(9):833–44. http://dx.doi.org/10.1016/0098-1354(93)E0023-3.

[11] Douguet D, Thoreau E, Grassy G. A genetic algorithm for the automated generation of small organic molecules: drug design using an evolutionary algorithm. J Comput-Aided Mol Des 2000;14(5):449–66. http://dx.doi.org/10.1023/A:1008108423895.

[12] Jensen JH. A graph-based genetic algorithm and generative model/Monte Carlo tree search for the exploration of chemical space. Chem Sci 2019;10(12):3567–72. http://dx.doi.org/10.1039/C8SC05372C.

[13] Weininger D. SMILES, a chemical language and information system. 1. Introduction to methodology and encoding rules. J Chem Inf Comput Sci 1988;28(1):31–6. http://dx.doi.org/10.1021/ci00057a005.

[14] Hochreiter S, Schmidhuber J. Long short-term memory. Neural Comput 1997;9:1735–80. http://dx.doi.org/10.1162/neco.1997.9.8.1735.

[15] Cho K, Merrienboer B van, Bahdanau D, Bengio Y. On the properties of neural machine translation: encoder-decoder approaches; 2014, arXiv:1409.1259.

[16] O'Boyle N, Dalke A. DeepSMILES: an adaptation of SMILES for use in machine-learning of chemical structures; 2018. http://dx.doi.org/10.26434/chemrxiv.7097960.v1.

[17] Krenn M, Häse F, Nigam A, Friederich P, Aspuru-Guzik A. Self-referencing embedded strings (SELFIES): a 100% robust molecular string representation; 2020, arXiv:1905.13741.

[18] Scarselli F, Gori M, Tsoi AC, Hagenbuchner M, Monfardini G. The graph neural network model. IEEE Trans Neural Netw 2009;20(1):61–80. http://dx.doi.org/10.1109/TNN.2008.2005605.

[19] Elton DC, Boukouvalas Z, Fuge MD, Chung PW. Deep learning for molecular design—a review of the state of the art. Mol Syst Des Eng 2019;4(4):828–49. http://dx.doi.org/10.1039/C9ME00039A.

[20] Brown N, Fiscato M, Segler MH, Vaucher AC. GuacaMol: benchmarking models for de novo molecular design. J Chem Inf Model 2019;59(3):1096–108. http://dx.doi.org/10.1021/acs.jcim.8b00839.

[21] Preuer K, Renz P, Unterthiner T, Hochreiter S, Klambauer G. Fréchet ChemNet distance: a metric for generative models for molecules in drug discovery. J Chem Inf Model 2018;58(9):1736–41. http://dx.doi.org/10.1021/acs.jcim.8b00234.

[22] Polykovskiy D, Zhebrak A, Sanchez-Lengeling B, Golovanov S, Tatanov O, Belyaev S, et al. Molecular sets (MOSES): a benchmarking platform for molecular generation models; 2018, arXiv:1811.12823.

[23] Kusner MJ, Paige B, Hernández-Lobato JM. Grammar variational autoencoder; 2017, arXiv:1703.01925.

[24] You J, Liu B, Ying R, Pande V, Leskovec J. Graph convolutional policy network for goal-directed molecular graph generation; 2019, arXiv:1806.02473.

[25] Zhou Z, Kearnes S, Li L, Zare RN, Riley P. Optimization of molecules via deep reinforcement learning; 2018, arXiv:1810.08678.

[26] Zhang C, Lyu X, Huang Y, Tang Z, Liu Z. Molecular graph generation with deep reinforced multitask network and adversarial imitation learning. IEEE Int Conf Bioinform Biomed 2019. http://dx.doi.org/10.1109/BIBM47256.2019.8983277.

[27] Bickerton GR, Paolini GV, Besnard J, Muresan S, Hopkins AL. Quantifying the chemical beauty of drugs. Nat Chem 2012;4(2):90–8. http://dx.doi.org/10.1038/nchem.1243.

[28] Ertl P, Schuffenhauer A. Estimation of synthetic accessibility score of drug-like molecules based on molecular complexity and fragment contributions. J Cheminform 2009;1(1):8. http://dx.doi.org/10.1186/1758-2946-1-8.

[29] Merk D, Friedrich L, Grisoni F, Schneider G. De novo design of bioactive small molecules by artificial intelligence. Mol Inform 2018;37(1–2). http://dx.doi.org/10.1002/minf.201700153.

[30] Merk D, Grisoni F, Friedrich L, Schneider G. Tuning artificial intelligence on the de novo design of natural-product-inspired retinoid X receptor modulators. Nat Commun Chem 2018;1(1):1–9. http://dx.doi.org/10.1038/s42004-018-0068-1.

[31] Zhavoronkov A, Ivanenkov YA, Aliper A, Veselov MS, Aladinskiy VA, Aladinskaya AV, et al. Deep learning enables rapid identification of potent DDR1 kinase inhibitors. Nat Biotechnol 2019;37(9):1038–40. http://dx.doi.org/10.1038/s41587-019-0224-x.
[32] Li Y, Vinyals O, Dyer C, Pascanu R, Battaglia P. Learning deep generative models of graphs; 2018, arXiv:1803.03324.
[33] Jin W, Barzilay R, Jaakkola T. Junction tree variational autoencoder for molecular graph generation; 2018, arXiv:1802.04364.
[34] Guimaraes GL, Sanchez-Lengeling B, Outeiral C, Farias PLC, Aspuru-Guzik A. Objective-reinforced generative adversarial networks (ORGAN) for sequence generation models; 2017, arXiv:1705.10843.
[35] Sanchez-Lengeling B, Outeiral C, Guimaraes GL, Aspuru-Guzik A. Optimizing distributions over molecular space. An objective-reinforced generative adversarial network for inverse-design chemistry (ORGANIC); 2017. http://dx.doi.org/10.26434/chemrxiv.5309668.v3.
[36] Kadurin A, Nikolenko S, Khrabrov K, Aliper A, Zhavoronkov A. druGAN: an advanced generative adversarial autoencoder model for de novo generation of new molecules with desired molecular properties in silico. Mol Pharm 2017;14(9):3098–104. http://dx.doi.org/10.1021/acs.molpharmaceut.7b00346.
[37] Kadurin A, Aliper A, Kazennov A, Mamoshina P, Vanhaelen Q, Khrabrov K, et al. The cornucopia of meaningful leads: applying deep adversarial autoencoders for new molecule development in oncology. Oncotarget 2016;8(7):10883–90. http://dx.doi.org/10.18632/oncotarget.14073.
[38] Lal GS, Pez GP, Syvret RG. Electrophilic NF fluorinating agents. Chem Rev 1996;96(5):1737–56. http://dx.doi.org/10.1021/cr941145p.
[39] Lehman J, Clune J, Misevic D, Adami C, Altenberg L, Beaulieu J, et al. The surprising creativity of digital evolution: a collection of anecdotes from the evolutionary computation and artificial life research communities; 2019, arXiv:1803.03453.
[40] Popova M, Isayev O, Tropsha A. Deep reinforcement learning for de novo drug design. Sci Adv 2018;4(7). http://dx.doi.org/10.1126/sciadv.aap7885. eaap7885.
[41] Nguyen A, Yosinski J, Clune J. Deep neural networks are easily fooled: high confidence predictions for unrecognizable images; 2015, arXiv:1412.1897.
[42] Szegedy C, Zaremba W, Sutskever I, Bruna J, Erhan D, Good-fellow I, et al. Intriguing properties of neural networks; 2014, arXiv:1312.6199.
[43] Bento AP, Gaulton A, Hersey A, Bellis LJ, Chambers J, Davies M, et al. The ChEMBL bioactivity database: an update. Nucleic Acids Res 2014;42(D1):D1083–90. http://dx.doi.org/10.1093/nar/gkt1031.
[44] Breiman L. Random forests. Mach Learn 2001;45(1):5–32. http://dx.doi.org/10.1023/A:1010933404324.
[45] Pedregosa F, Varoquaux G, Gramfort A, Michel V, Thirion B, Grisel O, et al. Scikit-learn: machine learning in python. J Mach Learn Res 2011;12(85):2825–30, http://jmlr.org/papers/v12/pedregosa11a.html.
[46] Rogers D, Hahn M. Extended-connectivity fingerprints. J Chem Inf Model 2010;50(5):742–54. http://dx.doi.org/10.1021/ci100050t.
[47] Landrum G. RDKit: open-source cheminformatics; 2006, http://www.rdkit.org.
[48] Winter R, Montanari F, Steffen A, Briem H, Noé F, Clevert D-A. Efficient multi-objective molecular optimization in a continuous latent space. Chem Sci 2019;10(34):8016–24. http://dx.doi.org/10.1039/C9SC01928F.
[49] Winter R, Montanari F, Noé F, Clevert D-A. Learning continuous and data-driven molecular descriptors by translating equivalent chemical representations. Chem Sci 2019;10(6):1692–701. http://dx.doi.org/10.1039/C8SC04175J.
[50] Liu X, Ye K, Vlijmen HWT van, IJzerman AP, Westen GJP van. An exploration strategy improves the diversity of de novo ligands using deep reinforcement learning: a case for the adenosine A2A receptor. J Cheminform 2019;11(1):35. http://dx.doi.org/10.1186/s13321-019-0355-6.
[51] Wildman SA, Crippen GM. Prediction of physicochemical parameters by atomic contributions. J Chem Inf Comput Sci 1999;39(5):868–73. http://dx.doi.org/10.1021/ci9903071.
[52] Mayr A, Klambauer G, Unterthiner T, Steijaert M, Wegner JK, Ceulemans H, et al. Large-scale comparison of machine learning methods for drug target prediction on ChEMBL. Chem Sci 2018. http://dx.doi.org/10.1039/C8SC00148K.
[53] Gao W, Coley CW. The synthesizability of molecules proposed by generative models. J Chem Inf Model 2020. http://dx.doi.org/10.1021/acs.jcim.0c00174.