Abstract

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 of generative models and how these evade detection by current performance metrics.

1. Introduction

In recent years, there has been increased interest in generative models for molecules in drug discovery (Segler et al. 2018; Gómez-Bombarelli et al. 2018; Sanchez-Lengeling and Aspuru-Guzik 2018; Olivecrona et al. 2017). In the area of de novo molecular design (Schneider 2013), 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 (Segler et al. 2018). 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 (Schmidhuber 2015; LeCun, Bengio, and Hinton 2015). These complement more classical approaches (Venkatasubramanian, Chan, and Caruthers 1994; Douguet, Thoreau, and Grassy 2000; Jensen 2019). 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 learning 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 (Segler et al. 2018; Gómez-Bombarelli et al. 2018) used the SMILES (Weininger 1988) representation of compounds in combination with recurrent neural networks (RNNs) (Hochreiter and Schmidhuber 1997; Cho et al. 2014), 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 (O’Boyle and Dalke 2018) or SELFIES (Krenn et al. 2020). Another range of models directly generate molecular graphs, using graph neural networks (Scarselli et al. 2009). Apart from the utilized representation, models differ in the training procedure and model architecture. For a more detailed overview on current approaches, see (Sanchez-Lengeling and Aspuru-Guzik 2018; Elton et al. 2019).

There are two main use cases of generative models for molecules which are distribution-learning and goal-directed generation (Brown et al. 2019). 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 (Preuer et al. 2018; Brown et al. 2019; Polykovskiy et al. 2018). These benchmarks often comprise heuristics that try to capture desired prop-
Goaldirected 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 (Kusner, Paige, and Hernández-Lobato 2017; You et al. 2019; Zhou et al. 2018; Zhang et al. 2019). It should however be noted that it is trivial to achieve stateoftheart results by generating long saturated hydrocarbon chains. The GuacaMol package (Brown et al. 2019) 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 goaldirected 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 druglikeness both of which are hard to quantify (Bickerton et al. 2012; Ertl and Schuffenhauer 2009). 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 3.

In spite of the deluge of publications detailing new approaches towards the generation of molecules, wetlab validations of generative models remain scarce. Merk, Friedrich, et al. (2018) and Merk, Grisoni, et al. (2018) showed that molecules generated using transfer learning (Segler et al. 2018) showed activity in vitro. Zhavoronkov et al. (2019) 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 distributionlearning and goaldirected generation with a focus on the latter.

2. Failure mechanisms in distributionlearning

First, we review problems of existing metrics for distributionlearning 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 distributionlearning models for molecules include auto-regressive approaches (Segler et al. 2018; Li et al. 2018), variational autoencoders (Gómez-Bombarelli et al. 2018; Kusner, Paige, and Hernández-Lobato 2017; Jin, Barzilay, and Jaakkola 2018), generative adversarial networks (Guimaraes et al. 2017; Sanchez-Lengeling, Outeiral, et al. 2017), and adversarial autoencoders (Kadurin, Nikolenko, et al. 2017; Kadurin, Aliper, et al. 2016). Previous studies in distributionlearning, 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. (2018) 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 (Brown et al. 2019; Polykovskiy et al. 2018). While molecules generated by distributionlearning 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 (Preuer et al. 2018; Brown et al. 2019; Polykovskiy et al. 2018). 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 (Brown et al. 2019). 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 S1.1.
On failure modes of molecule generators and optimizers

| 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     |

Table 1. Comparison of the AddCarbon model to the baselines in (Brown et al. 2019), from where the table was adapted. Column names represent molecule generators. Random sampling "RS" randomly picks a molecule from the training set.

We evaluate this model using the GuacaMol distribution-learning benchmark (Brown et al. 2019). 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. While the FCD detects that these naively generated molecules do not reflect the training distribution, it might be tricked as well by a naive model similar to the AddCarbon model. 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 (Brown et al. 2019) and Moses (Polykovskiy et al. 2018) benchmarks are likelihood-based models (Segler et al. 2018; Jin, Barzilay, and Jaakkola 2018; Gómez-Bombarelli et al. 2018). 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.

3. Failure mechanisms in goal-directed generation

3.1. 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 gets more problematic when trying to optimize for multiple properties at once.

Even when high scores are obtained by the generated molecules, this might be achieved in a manner not expected by the practitioner. While, from an optimization perspective, the task can be considered solved if high scoring molecules are generated, the results might not be satisfying or useful. For example the generated molecules shown in Fig. 1 have high scores but contain unstable or synthetically infeasible substructures.

In practical applications, we have found generative models to be highly adept at 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.

This phenomenon of optimization of a score in unexpected ways, has been observed in other applications (Lehman et al. 2019). 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
On failure modes of molecule generators and optimizers

(a) Compounds created by a graph-based genetic algorithm. The left compound contains a reactive diene. The right compound contains reactive imine and diene moieties, as well as an unusual nitrogen-fluorine bond.

(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

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.
learning models on experimental data (Olivecrona et al. 2017; Popova, Isayev, and Tropsha 2018; Segler et al. 2018). 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 (Nguyen, Yosinski, and Clune 2015). Samples generated in such a way have been shown to exploit features unique to the exact models they were optimized for (Szegedy et al. 2014). 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 it is often observed that actives/inactives from the training set receive higher/lower prediction scores than those in the test set. 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.

### 3.2. 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 D2 (DRD2) for which we downloaded data from ChEMBL (Bento et al. 2014). We preprocessed the data to obtain binary classification tasks. Information about the data is displayed in Table 2.

| Target | ChEMBL ID | Active | Inactive | AUC       |
|--------|-----------|--------|----------|-----------|
| JAK2   | CHEMBL3888429 | 140    | 527      | 0.78 ± 0.03 |
| EGFR   | CHEMBL1909203 | 40     | 802      | 0.76 ± 0.05 |
| DRD2   | CHEMBL1909140 | 59     | 783      | 0.86 ± 0.03 |

Table 2. Information on the data sets. AUC shows the performance of the trained classifiers on hold out test data.

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.

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 (Breiman 2001) as implemented in scikit-learn (Pedregosa et al. 2011) 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 (Rogers and Hahn 2010) of size 1024 and radius 2, which were calculated using rdkit (Landrum 2006). We obtained three classifiers for each of the three targets, JAK2, EGFR, and DRD2, with similar predictive performance (see Table 2) 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. Here we employ the two top-performing molecule generators according to Guacamol (Brown et al. 2019). One of these is a graph-based genetic algorithm (GA) (Jensen 2019). 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 (Brown et al. 2019), which is in turn a random subset of the compounds in ChEMBL (Bento et al. 2014). The second optimization algorithm we use is called SMILES-LSTM (LSTM) (Segler et al. 2018). 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 (Brown et al. 2019) to initialize the generator. This model was trained on the same data set, from which the starting population for GA is drawn. 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.

### 3.3. 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, 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. S1. 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. S4), as optimized molecules have a more similar neighbour in split 1 than in split 2 on average, as measured by ECFP4 Tanimoto 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 OS 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
On failure modes of molecule generators and optimizers

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.

suggests that optimization should be stopped early once the control scores cease to increase.

Our experiments and the conclusions drawn from these are limited by the design choices we made. We used the LSTM and GA molecular optimizers as they performed well in GuacaMol (Brown et al. 2019). 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 (Mayr et al. 2018). 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.

4. Discussion and Conclusion

In this work, we reviewed generative models for molecules in light of the 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 (Brown et al. 2019; Polykovskiy et al. 2018) 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 investigate 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 (Gao and Coley 2020) 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.

5. Availability

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

6. Conflict of interest

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

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

References

Segler, M. H. S., Kogej, T., Tyrchan, C., and Waller, M. P. (Jan. 24, 2018). “Generating Focused Molecule Libraries for Drug Discovery with Recurrent Neural Networks”. In: ACS Central Sci 4.1, pp. 120–131. URL: https://doi.org/10.1021/acscentsci.7b00572.

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. (Feb. 28, 2018). “Automatic Chemical Design Using a Data-Driven Continuous Representation of Molecules”. In: ACS Central Sci 4.2, pp. 268–276. URL: https://doi.org/10.1021/acscentsci.7b00572.

Sanchez-Lengeling, B. and Aspuru-Guzik, A. (July 27, 2018). “Inverse Molecular Design Using Machine Learning: Generative Models for Matter Engineering”. In: Science 361.6400, pp. 833–844. URL: https://doi.org/10.1126/science.aat2663.

Olivecrona, M., Blaschke, T., Engkvist, O., and Chen, H. (Sept. 4, 2017). “Molecular De-Novo Design through Deep Reinforcement Learning”. In: J Cheminformatics 9.1, p. 48. URL: https://doi.org/10.1186/s13321-017-0235-x.

Schneider, G. (2013). De Novo Molecular Design. John Wiley & Sons, Ltd. URL: https://doi.org/10.1002/9783527677016.

Schmidhuber, J. (Jan. 1, 2015). “Deep Learning in Neural Networks: An Overview”. In: Neural Netw 61, pp. 85–117. URL: https://doi.org/10.1016/j.neunet.2014.09.003.

LeCun, Y., Bengio, Y., and Hinton, G. (May 2015). “Deep Learning”. In: Nature 521.7553, pp. 436–444. URL: https://doi.org/10.1038/nature14539.

Venkatasubramanian, V., Chan, K., and Caruthers, J. M. (Sept. 1, 1994). “Computer-Aided Molecular Design Using Genetic Algorithms”. In: Comput Chem Eng 18.9, pp. 833–844. URL: https://doi.org/10.1016/0098-1354(93)E0023-3.

Douguet, D., Thoreau, E., and Grassy, G. (July 1, 2000). “A Genetic Algorithm for the Automated Generation of Small Organic Molecules: Drug Design Using an Evolutionary Algorithm”. In: J Comput-Aided Mol Des 14.5, pp. 449–466. URL: https://doi.org/10.1023/A:1008108423895.

Jensen, J. H. (Mar. 20, 2019). “A Graph-Based Genetic Algorithm and Generative Model/Monte Carlo Tree Search for the Exploration of Chemical Space”. In: Chem Sci 10.12, pp. 3567–3572. URL: https://doi.org/10.1039/C8SC05372C.

Weininger, D. (Feb. 1, 1988). “SMILES, a Chemical Language and Information System. 1. Introduction to Methodology and Encoding Rules”. In: J Chem Inf Comput Sci 28.1, pp. 31–36. URL: https://doi.org/10.1021/ci00057a005.

Hochreiter, S. and Schmidhuber, J. (Dec. 1, 1997). “Long Short-Term Memory”. In: Neural comput 9, pp. 1735–80. URL: https://doi.org/10.1162/neco.1997.9.8.1735.

Cho, K., Merrienboer, B. van, Bahdanau, D., and Bengio, Y. (Oct. 7, 2014). On the Properties of Neural Machine Translation: Encoder-Decoder Approaches. arXiv: 1409.1259. URL: http://arxiv.org/abs/1409.1259.

O’Boyle, N. and Dalke, A. (Sept. 19, 2018). DeepSMILES: An Adaptation of SMILES for Use in Machine-Learning of Chemical Structures. chemrxiv: 7097960.v1. URL: https://doi.org/10.26434/chemrxiv.7097960.v1.

Krenn, M., H¨ase, F., Nigam, A., Friederich, P., and Aspuru-Guzik, A. (Mar. 4, 2020). Self-Referencing Embedded Strings (SELFIES): A 100% Robust Molecular String Representation. arXiv: 1905.13741. URL: http://arxiv.org/abs/1905.13741.

Scarselli, F., Gori, M., Tsoi, A. C., Hagenbuchner, M., and Monfardini, G. (Jan. 2009). “The Graph Neural Network Model”. In: IEEE Trans Neural Netw 20.1, pp. 61–80. URL: https://doi.org/10.1109/TNN.2008.2005605.

Elton, D. C., Boukouvalas, Z., Fuge, M. D., and Chung, P. W. (Aug. 5, 2019). Deep Learning for Molecular Design—a Review of the State of the Art”. In: Mol Syst Des Eng 4.4, pp. 828–849. URL: https://doi.org/10.1039/C9ME00039A.

Brown, N., Fiscato, M., Segler, M. H., and Vaucher, A. C. (Mar. 25, 2019). “GuacaMol: Benchmarking Models for de Novo Molecular Design”. In: J Chem Inf Model 59.3, pp. 1096–1108. URL: https://doi.org/10.1021/acs.jcim.8b00839.

Preuer, K., Renz, P., Unterrhiner, T., Hochreiter, S., and Klaban, G. (Sept. 24, 2018). “Fr´echet ChemNet Distance: A Metric for Generative Models for Molecules in Drug Discovery”. In: J Chem Inf Model 58.9, pp. 1736–1741. URL: https://doi.org/10.1021/acs.jcim.8b00234.

Polykovskiy, D., Zhebrak, A., Sanchez-Lengeling, B., Golovanov, S., Tatanov, O., Belyaev, S., Kurbanov, R., Aramanov, A., Aladinskiy, V., Veselov, M., et al. (Nov. 29, 2018). Molecular Sets (MOSES): A Benchmarking Plat-
form for Molecular Generation Models. arXiv: 1811.12823. URL: http://arxiv.org/abs/1811.12823.

Kusner, M. J., Paige, B., and Hernández-Lobato, J. M. (Mar. 6, 2017). Grammar Variational Autoencoder. arXiv: 1703.01925. URL: http://arxiv.org/abs/1703.01925.

You, J., Liu, B., Ying, R., Pande, V., and Leskovec, J. (Feb. 24, 2019). Graph Convolutional Policy Network for Goal-Directed Molecular Graph Generation. arXiv: 1806.02473. URL: http://arxiv.org/abs/1806.02473.

Zhavoronkov, A., Ivanenkov, Y. A., Aliper, A., Veselov, M. S., Aladinskiy, V. A., Polykovskiy, D. A., Kuznetsov, M. D., Asadulaev, A., et al. (Sept. 5, 2017). “druGAN: An Advanced Generative Adversarial Autoencoder Model for de Novo Generation of New Molecules with Desired Molecular Properties in Silico”. In: Mol Pharm 14.9, pp. 10883–10890. URL: https://doi.org/10.1021/acs.molpharmaceut.7b00346.

Kadurin, A., Aliper, A., Kazennov, A., Mamoshina, P., Vahaljen, Q., Khrabrov, K., and Zhavoronkov, A. (Dec. 22, 2016). “The Cornucopia of Meaningful Leads: Applying Deep Adversarial Autoencoders for New Molecule Development in Oncology”. In: Oncotarget 8.7, pp. 10883–10890. URL: https://doi.org/10.18632/oncotarget.14073.

Lehman, J., Clune, J., Misievic, D., Adami, C., Altenberg, L., Beaulieu, J., Bentley, P. J., Bernard, S., Beslgon, L., Bryson, D. M., et al. (Nov. 21, 2019). The Surprising Creativity of Digital Evolution: A Collection of Anecdotes from the Evolutionary Computation and Artificial Life Research Communities. arXiv: 1803.03453. URL: http://arxiv.org/abs/1803.03453.

Nguyen, A., Yosinski, J., and Clune, J. (Apr. 2, 2015). Deep Neural Networks Are Easily Fooled: High Confidence Predictions for Unrecognizable Images. arXiv: 1412.1897. URL: http://arxiv.org/abs/1412.1897.

Szygedy, C., Zaremba, W., Sutskever, I., Bruna, J., Erhan, D., Goodfellow, I., and Fergus, R. (Feb. 19, 2014). Intriguing Properties of Neural Networks. arXiv: 1312.6199. URL: http://arxiv.org/abs/1312.6199.
Bioactivity Database: An Update”. In: Nucleic Acids Res 42.D1, pp. D1083–D1090. URL: https://doi.org/10.1093/nar/gkt1031.

Breiman, L. (Oct. 1, 2001). “Random Forests”. In: Mach Learn 45.1, pp. 5–32. URL: https://doi.org/10.1023/A:1010933404324.

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., et al. (2011). “Scikit-Learn: Machine Learning in Python”. In: J Mach Learn Res 12.85, pp. 2825–2830. URL: http://jmlr.org/papers/v12/pedregosalla.html.

Rogers, D. and Hahn, M. (May 24, 2010). “Extended-Connectivity Fingerprints”. In: J Chem Inf Model 50.5, pp. 742–754. URL: https://doi.org/10.1021/ci100050t.

Landrum, G. (2006). RDKit: Open-Source Cheminformatics. URL: http://www.rdkit.org.

Mayr, A., Klambauer, G., Unterthiner, T., Steijaert, M., Wenginger, J. K., Ceulemans, H., Clevert, D.-A., and Hochreiter, S. (June 6, 2018). “Large-Scale Comparison of Machine Learning Methods for Drug Target Prediction on ChEMBL”. In: Chem Sci. URL: https://doi.org/10.1039/C8SC00148K.

Gao, W. and Coley, C. W. (Apr. 6, 2020). “The Synthesizability of Molecules Proposed by Generative Models”. In: J Chem Inf Model. URL: https://doi.org/10.1021/acs.jcim.0c00174.
S1. Method details

S1.1. AddCarbon model

Sampling from the AddCarbon model is done in the following way. A random compound is drawn from the training set and its canonical SMILES (Weininger 1988) is calculated using rdkit (Landrum 2006). Then a ‘C’ is added to a uniformly random position in this canonical SMILES. If the resulting string is not a valid SMILES another position is tested. If it is valid and the corresponding canonical SMILES has a string edit distance of 1 to the original canonical SMILES and is not already in the training set it is returned. If no position results in a valid SMILES a new compound is chosen from the training set and the same procedure is applied.

S1.2. Preprocessing of assay data

We downloaded the data for all three optimization tasks from ChEMBL. The identifiers for the assays are listed in 2. To obtain binary labels for the JAK2 task we labelled all compounds with a pIC\textsubscript{50} greater than 8 as active. For the EGFR and DRD2 tasks we extracted the label from the "Comment" column.

S1.3. Molecular optimizers

We used the implementation of both the graph-based genetic algorithm and the hill-climbing SMILES-LSTM provided by (Brown et al. 2019). For the genetic algorithm we used following settings:

- population\_size:100 (Population size)
- offspring\_size:200 (size of offspring)
- generations:150 (Number of optimization generations)
- mutation\_rate:0.01 (Rate of mutation)
- random\_start:True (Whether to use a random starting population)
- patience:5 (How long to be patient without improvements)
- smi\_file:described below (List of molecules to draw initial population from)

The starting population was drawn from the distribution-learning training set also provided by (Brown et al. 2019). For the hill-climbing SMILES-LSTM we used these settings:

- mols_to_sample:1028 (Molecules to sample in each step)
- keep\_top:512 (Number of molecules to keep in each iteration)
- optimize\_n\_epochs:1 (Number of fine-tuning epochs per iteration)
- max\_len:100 (Maximum SMILES length)
- optimize\_batch\_size:64 (Finetune optimization batch size)
- number_final_samples:1028 (Number of final samples returned)
- sample_final\_model\_only: False (Whether intermediate samples are returned among results)
- random\_start:True (Whether to start randomly or pretrain on best compounds of a starting population)
Figure S1. Scatter plots of OS vs. DCS on molecules during the course of training. Different colors correspond to different runs. Grey points are random compounds from ChEMBL. The contour lines show the region where split 1 actives lie. Due to symmetry the region of split 2 actives could be obtained by mirroring the contour lines across the diagonal. During the course of training the optimized molecules move towards the region of split 1 actives.
On failure modes of molecule generators and optimizers

Figure S2. Random samples generated for the EGFR task, with actives from the training set for comparison.
Figure S3. Random samples generated for the JAK2 task, with actives from the training set for comparison.
Figure S4. Distribution of nearest neighbour similarities of the optimized molecules to split 1/2. The results of all runs have been merged for each combination of data set and optimizer. In most cases the nearest neighbour similarities are greater for split 1 than for split 2. Tanimoto similarities are based on the same ECFP4 fingerprints that were used for training the classifiers.