JOHANNES KEPLER
UNIVERSITY LINZ

Submitted by
Philipp Renz
01126686

Submitted at
Institute for Machine Learning

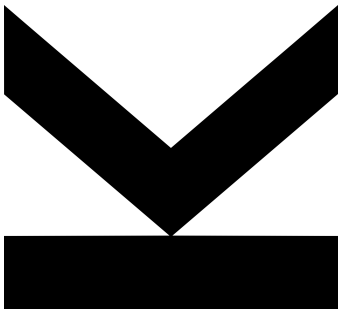
Thesis Supervisor / First Evaluator
Univ.-Prof. Mag. Dr.
Günter Klambauer

Co-Supervisor
Univ.-Prof. Dr. **Sepp Hochreiter**

Second Evaluator
Prof. Dr. **Andreas Bender**

September 2024

Generative Models in Drug Discovery:
Advancing Assessments, Metrics and Retrosynthesis Prediction



Doctoral Thesis

to obtain the academic degree of

Doktor der Naturwissenschaften

in the Doctoral Program

Naturwissenschaften

JOHANNES KEPLER UNIVERSITY LINZ
Altenbergerstraße 69
4040 Linz, Austria
www.jku.at
DVR 0093696

Abstract

In recent years the use of generative models in drug discovery has seen increased attention, as novel deep learning architectures have shown great flexibility in generating molecular structures. Two main applications of generative models in drug discovery are de novo molecular design and retrosynthesis prediction. However, the evaluation of generative models in de novo molecular design is challenging and existing benchmarks and evaluation metrics often may not reflect the practical utility of the models. In this thesis, investigate the limitations of current evaluation metrics and propose new evaluation strategies for generative models in drug discovery. In addition, we introduce a novel approach for retrosynthesis prediction, a crucial task in computer-aided synthesis planning.

The first part of this thesis focuses on observed failure modes in the evaluation of generative models for de novo molecular design. In particular we show that commonly used metrics used to evaluate distribution-learning are not sufficient to differentiate complex models from trivial baseline generators. Secondly, we show how goal-directed molecular generators can overfit to machine learning-based objective functions, leading to biases towards the training data and overestimation of the model's performance.

The second part introduces a diversity-based benchmark for goal-directed molecule generators, which measures the ability of a model to generate diverse sets of molecules with desired properties. Previous studies on diverse molecule optimization have been limited by inadequate diversity measures, non-standardized compute budgets, and lack of model adaptation to the diverse optimization setting. Our benchmark addresses these shortcomings, providing a standardized framework for evaluating diverse, goal-directed molecule generators and enabling fair model comparisons.

The third part of this thesis focuses on retrosynthesis prediction a crucial task in computer-aided synthesis planning (CASP). We propose a novel template-based retrosynthesis prediction model based on Modern Hopfield Networks. Our model takes both the target molecule and the reaction templates as input, which allows it to generalize over reaction templates, which improves performance, particularly on

rare templates. Our model achieves state-of-the-art performance while maintaining a significantly lower computational cost compared to existing methods.

Through our work, we provide insights into the capabilities and limitations of current generative models for molecules while proposing novel evaluation strategies. Additionally, our contributions in retrosynthesis prediction enables more accurate computer-aided synthesis planning. Collectively, these advances have the potential to accelerate the drug discovery pipeline and facilitate the development of novel pharmaceutical treatments.

Acknowledgement

Thank you all ..

Contents

List of Acronyms

CASP computer-aided synthesis planning

DMTA Design-Make-Test-Analyze

QSPR quantitative structure-property relationship

VS virtual screening

1 Introduction

1.1 Small molecule drug design

The discovery of novel drugs has significantly contributed to the improvement of human health and well-being. There is a continuous demand for new drugs to expand the range of treatable diseases, improve the efficacy of existing treatments, and respond to the emergence of new health challenges.

Small molecule drugs constitute the majority of medicines in use, accounting for approximately 90% of global sales (Makurvet, 2021). These molecules, typically defined as having a molecular weight of less than 1,000 Da, offer several advantages. They are generally stable, do not require specialized storage conditions, and can be conveniently administered orally. Moreover, they are relatively inexpensive to produce and can be easily synthesized in large quantities (Southey et al., 2023).

For a small molecule to be considered a viable drug candidate, it must fulfill a range of properties:

- **On-target activity:** The molecule must be active against the desired target to exhibit the intended therapeutic effect. At the molecular level, this means binding to the target and modulating its activity in the desired manner.
- **Specificity:** The molecule should demonstrate high specificity, selectively interacting with the intended target while minimizing undesirable off-target interactions. Such selectivity is crucial to prevent adverse side effects and maintain the drug's safety and efficacy profile.
- **Toxicity:** The absence of toxic effects is essential, as the molecule must be well-tolerated and free from potential harmful side effects. Toxicity can arise from various factors, including off-target interactions, metabolic byproducts, or allergic reactions.
- **Pharmacokinetics:** The molecule must possess favorable pharmacokinetic properties, encompassing adsorption, distribution, metabolism, and excretion (ADME). These properties determine how the molecule is absorbed into the

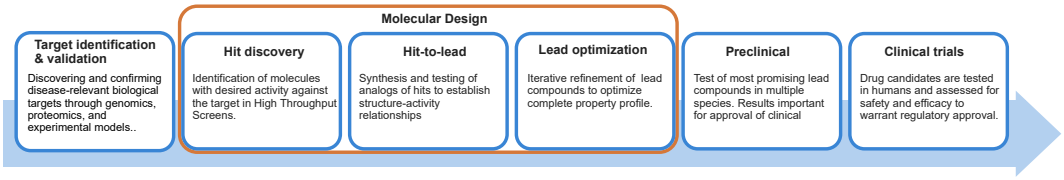


Figure 1.1: The drug discovery pipeline consists of several stages, ranging from target identification and validation to clinical trials. Molecular design (orange box) aims at identifying molecules with desired properties property profiles. This phase is particularly amenable to computational methods, which can accelerate the discovery process and reduce the cost of experimental testing.

.

body, distributed throughout, metabolized, and ultimately excreted. They are crucial for ensuring the molecule reaches its target effectively and is processed safely by the body.

- **Synthesizability:** The molecule must be synthesizable in a cost-effective manner to be practically viable for large-scale production.

In addition to these properties, the molecule must be novel and not infringe on existing patents. While this is not inherently necessary for a drug's efficacy, it represents a significant practical consideration in the pharmaceutical industry.

The primary challenge in drug discovery lies in identifying a molecule that satisfies all these criteria simultaneously. The development of a new drug is a complex and expensive process, which can take 12–15 years and cost estimates range between $1.8-2.5 billion (Paul et al., 2010; DiMasi et al., 2016).

1.1.1 The drug discovery pipeline

The drug discovery process is usually divided into several stages (**todo**) depicted in Figure 1.1 and described below.

- **Target Identification and Validation:** The drug discovery process begins with the identification of a biological target, which is usually a molecule, protein, or gene involved in a disease pathway. Understanding the target's role in the disease is crucial for developing therapeutic interventions.
- **Hit Discovery:** This stage aims at identifying "hits", which are molecules that exhibit activity against the target. High-throughput screening (HTS) is a common approach used to test large libraries of molecules against the target

in a rapid and automated manner. Computational methods such as virtual screening can also be employed to increase the hit-rate of wet-lab experiments.

- **Hit-to-lead and lead optimization:** Promising hits are then refined and optimized to produce lead compounds. This stage focuses on improving the activity, selectivity, and pharmacological properties of the molecules. The goal is to find a lead compounds with a desirable balance of potency, selectivity, and drug-like properties.
- **Preclinical Development:** The most promising lead compounds are then tested in preclinical studies, which are typically conducted in animal models. These studies assess the safety, efficacy, pharmacokinetics, and toxicology of the drug candidate in vivo. The data generated during this stage are critical for determining whether the candidate is suitable for clinical trials in humans.
- **Clinical Trials:** Drug candidates that pass preclinical development proceed to clinical trials, which are conducted in humans and are typically divided into three phases:
  - **Phase I:** This phase focuses on assessing the safety, tolerability, and pharmacokinetics of the drug in a small group of healthy volunteers or patients.
  - **Phase II:** In this phase, the efficacy of the drug is tested in a larger group of patients with the target disease. Safety and dosage optimization are also evaluated.
  - **Phase III:** This phase involves large-scale testing of the drug's safety and efficacy in a diverse patient population. It provides the critical data needed for regulatory approval.

  The success rates of clinical trials are low, with only about 10% of drugs that enter clinical trials eventually being approved by regulatory agencies More specifically, the success rates in Phase I/II/III and the final regulatory approval are 63%, 31%, 58% and 85% respectively (Mullard, 2016). This translates to 63%, 19.5%, 11.3% and 9.6% of projects that make it to the respective stages.
- **Regulatory Approval:** Upon successful completion of clinical trials, the drug is submitted for regulatory approval. Agencies such as the U.S. Food and Drug Administration (FDA) or the European Medicines Agency (EMA) review the comprehensive data package, including preclinical and clinical trial results. If

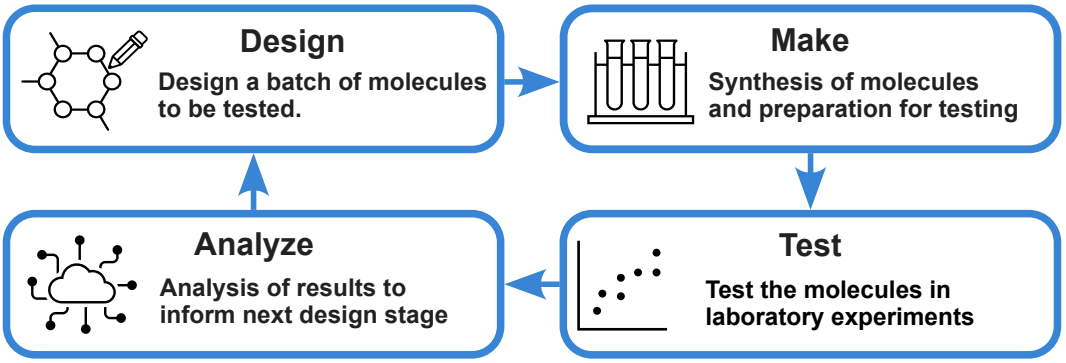


Figure 1.2: The Design-Make-Test-Analyze cycle is a key concept in drug discovery. The cycle consists of four stages: Design, Make, Test, and Analyze. Generative models can be used to design promising molecules to be tested. Computer-aided synthesis planning tools can be employed to make sure the molecules can be synthesized in the Make stage. The In the Analyze stage the experimental results can be used to update the property prediction models underlying the Design stage.

the drug is deemed safe and effective, it receives approval for marketing and distribution.

- **Post-Market Surveillance:** Post-market surveillance, or Phase IV studies, are conducted to monitor the long-term safety and efficacy of the drug in the general population. This stage can reveal rare side effects or long-term risks that were not apparent during clinical trials, and it may lead to further modifications, warnings, or even withdrawal of the drug from the market.

The general strategy of this process is to start with a large number of molecules and then systematically reduce the number to a few candidates that finally are submitted to clinical trials. The early stages have lower per molecule costs but higher uncertainty about the success chances of a molecule. The later stages are more expensive, provide more accurate information whether a molecule is safe and effective in humans. The early stages of the drug discovery process, ranging from hit discovery to lead optimization, are particularly amenable to computational methods, and this thesis focuses on the molecular design part of the drug discovery pipeline.

1.1.2 The Design-Make-Test-Analyze cycle

The hit discovery, hit-to-lead and lead optimization stages usually operate in an iterative manner, resulting in a cycle of choosing molecules to be tested, synthesizing

them, testing them in laboratory experiments and analyzing the results to guide the selection of the next molecule to be tested. This cycle is usually referred to as the *Design-Make-Test-Analyze* (*DMTA*)-cycle (Wesolowski et al., 2016):

- **Design:** Under consideration of previous experimental results, the molecules to be tested are designed. The design generally aims to optimize the desired properties of the molecule, but also to explore chemical space to find structure activity relationships.
- **Make:** The designed molecules are then synthesized and prepared for testing in the laboratory. This step requires a synthesis plan that outlines the steps needed to synthesize the molecule.
- **Test:** The synthesized molecules are then tested in laboratory experiments to measure the properties of interest. This can range from the activity of the molecule against a target, to its pharmacokinetic properties, to its toxicity and others.
- **Analyze:** The results of the experiments are analyzed. The obtained insights can then be used to guide the design of the next molecules to be tested. evaluation of the performance of the prediction models used in the design phase. The results of the analysis are then used to guide the design of the next molecule to be tested.

1.1.3 Computer-aided drug design

Recent years have seen a surge in the application of machine learning (ML) to various stages of the

Computer-aided drug design (CADD) has long been an integral part of the pharmaceutical research and development process. It encompasses a range of computational methods aimed at supporting and enhancing drug discovery. While traditional CADD approaches have proven valuable, the integration of machine learning (ML) has significantly expanded their capabilities.

Machine learning has become an important tool in modern CADD, offering new approaches for predicting molecular properties and activities. ML models, trained on datasets molecular structures and their associated properties, can screen chemical libraries to identify potential drug candidates, reducing the time and resources required for experimental testing

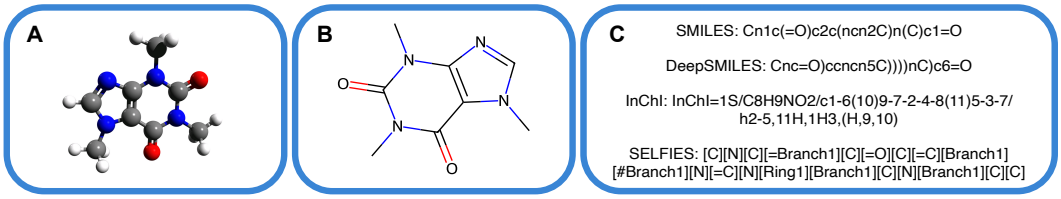


Figure 1.3: Different ways to represent a caffeine molecule **A:** The 3D structure of a molecule is given by the positions of its atoms in space. This structure is not necessarily fixed as some bonds can rotate and bonds can vibrate (Image source: (*English: Caffeine 3D Structure* 2010)). **B:** The graph representation of the same molecule. **C:** Smiles, DeepSmiles, SELFIES and InChI are line notations that linearize the molecules graph representation.

Recent advances in deep learning have led to a surge in interest in generative models, introducing new possibilities in drug discovery. These models expand the application of ML beyond property prediction to the creation of novel molecular structures. In this thesis we focus on two key applications of generative models in drug discovery:

- **De Novo Drug Design:** Generative models can create new molecular structures with desired property profiles. This approach enables the efficient exploration of chemical space, without the need of explicit enumeration large parts of chemical space.
- **Computer-Aided Synthesis Planning:** Generative models are also applied to accurately model chemical reactions. These models can be used to propose synthetic routes for target molecules, addressing a core challenge in drug development. By suggesting potential synthesis pathways, these models aim to support the transition from in silico design to experimental realization.

These applications of generative models show promise for improving the efficiency of pharmaceutical research. By facilitating the design of new molecules and plan their synthesis, these tools may accelerate the drug development process and increase the likelihood of identifying successful drug candidates.

1.2 Generative models in drug discovery

1.2.1 Molecular representations

Molecules, though fundamentally complex quantum mechanical entities, can be represented through various simplified models for practical purposes. The most

common representation depicts molecules as graphs, where atoms are nodes and chemical bonds are edges. Figure 1.3b shows a graph representation of caffeine. This graph structure captures the molecule's atoms and the chemical bonds between them. Additional properties such as atom type or charge are incorporated as features of the nodes and edges. While this representation does not capture the full quantum complexity, it provides a stable and practical framework for understanding and working with molecular structures in many scientific and computational contexts.

Molecular graphs can be linearized into one-dimensional character sequences, known as line notations. Figure 1.3c shows examples of various line notations. SMILES (Simplified Molecular Input Line Entry System) (Weininger, 1988) is a widely used line notation that represents molecules as strings of characters. SMILES strings encode the molecular graph in a human-readable format, making them convenient for storage and processing. Line notations have proven particularly valuable for generative models, as they are easily processed by sequence-based models like recurrent neural networks (RNNs) and Transformers (Vaswani et al., 2017). While SMILES strings have seen widespread adoption, other line notations have been proposed to make them more amenable for use in machine learning models. DeepSmiles (O'Boyle et al., 2018) aim to make it easier to generate syntactically valid molecules, by changing the notation of branches and ring closures. SELFIES (Krenn et al., 2022) provide a representation of molecules in which any sequence of tokens parses into a valid molecule. SAFE (Noutahi et al., 2023) provides a representation of molecules in which the substructures are represented by contiguous regions of a SMILES string. InChI (Heller et al., 2015) is less human-readable and less used in machine learning contexts, but provides a non-proprietory representation of molecules, with strict uniqueness and canonicalization rules.

Molecules can be represented in various complex forms beyond simple graphs and strings. Three-dimensional structures provide a spatial description of a molecule, detailing atomic positions in 3D space along with information about atom types and bonds, as shown in Figure 1.3. The most comprehensive representation is the quantum mechanical wavefunction, which captures the full complexity of molecular behavior. While these more detailled representations are valuable for modeling a wide range of molecular properties and interactions, but are not covered in the rest of this thesis.

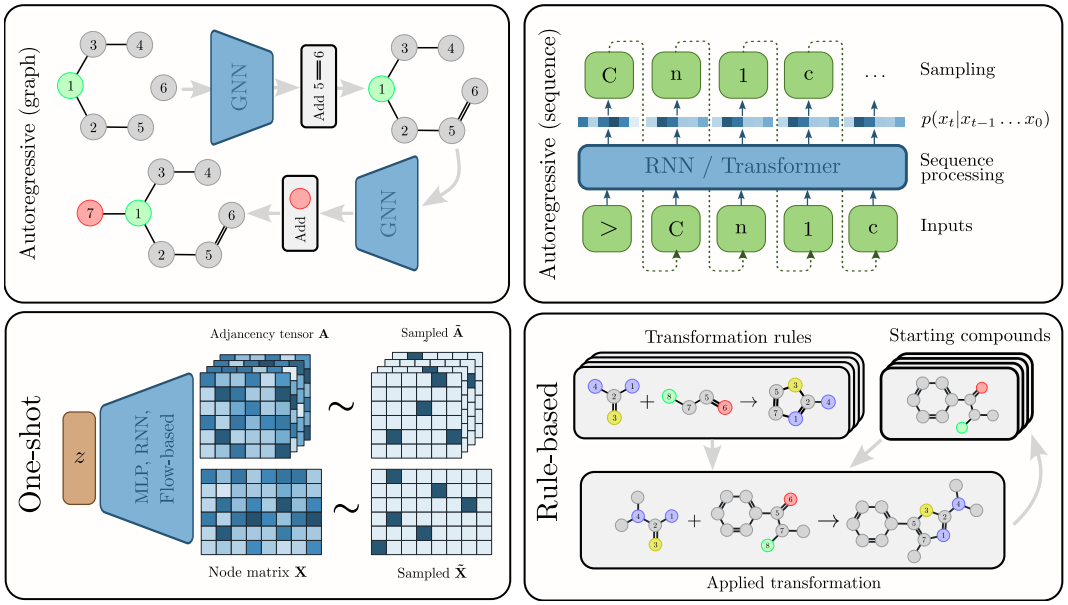


Figure 1.4: Different approaches to generate molecules in drug discovery. **A:** Sequence-based autoregressive models generate molecules by sampling one token at a time conditioned on the previously sampled tokens. **B:** Graph-based autoregressive models generate molecules by iteratively adding nodes and edges to the graph making use of Graph Neural Networks to decide which node or edge to add next. **C:** One-shot methods generate molecules in a single step by generating an adjacency matrix and node feature matrix of a molecule. **D:** Rule-based models generate molecules by applying a set of pre-defined graph transformation rules to iteratively combine a set of starting molecules.

1.2.2 Generation strategies

There are several approaches to constructing up molecular graphs making use of different molecular representations and strategies to generate molecules in that representation. These generation strategies can be adapted to solve different tasks by modifying the the model architecture and training procedure, which we describe in the following sections. Some of the most common approaches to molecular generation are shown in Figure 1.4 and described next.

Sequence-based autoregressive models constitute one of the most popular approaches for generating molecules. This approach makes use of a linearized representation of the molecule, such as a SMILES string. The model generates molecules by sampling one token at a time, conditioned on the previously sampled tokens, similarly to how a language model generates text. Early work by (Segler et al., 2018a) and (Gómez-Bombarelli et al., 2018) relied on recurrent neural networks (RNNs) to generate SMILES strings. This approach has since been popular and there has been work on string-based representations more suitable to generation (O'Boyle et al., 2018; Krenn et al., 2020; Noutahi et al., 2023), parsing the molecules into specialized data structures (Kusner et al., 2017; Jin et al., 2018) and using other deep learning architectures such as transformers (Vaswani et al., 2017; Noutahi et al., 2023; Schwaller et al., 2019; Bagal et al., 2022; Mazuz et al., 2023).

Graph-based autoregressive models work similarly to its sequence-based counterpart, but instead of relying on a linearization of the molecule, they work directly on the graph representation of the molecule. The model generates the molecular graph by iteratively adding nodes and edges to the graph, often relying on graph neural networks to decide which node or edge to add next. These models are similar to the sequence-based models described above, as they also generate the molecule in an autoregressive manner, but are more complex as there is no prescribed order in which nodes and edges are added to the graph. This approach has been used in several works in drug discovery (Liu et al., 2018; Li et al., 2018; You et al., 2019; Cohen-Karlik et al., 2024).

One-shot methods are a class of models that generate molecules in one step, without the need for an iterative generation process. These models generate a node feature matrix encoding the atom types and an adjacency tensor encoding the connectivity and bond types between the atoms. This is done in a single step, without the need for an iterative generation process. However, usually continuous versions of the

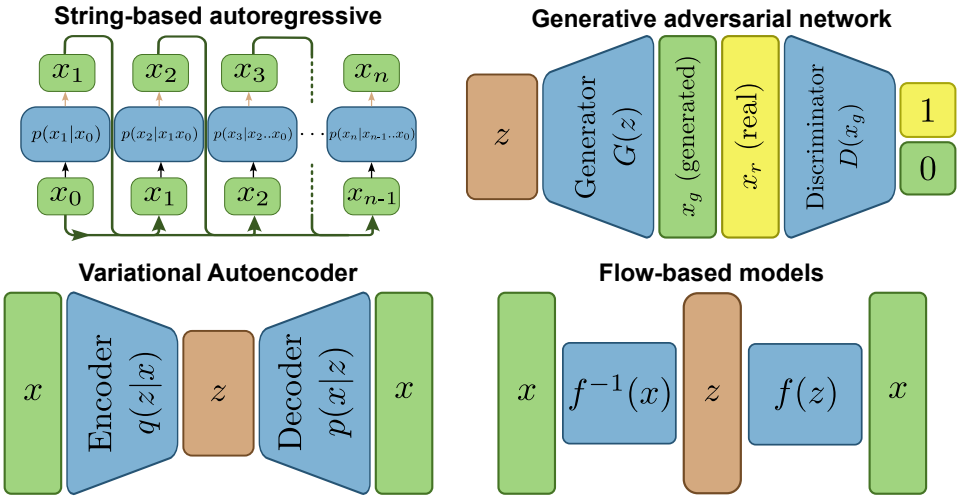


Figure 1.5: Different types of distribution-learning models. All model types try to fit the data distribution $p(x)$, but differ in the way they achieve this goal. While autoregressive models and generative flows the exact likelihood of the data can be calculated, and optimized, VAEs rely on a variational approximation of the likelihood, and generative adversarial networks indirectly fit the data distribution using a game-theoretic approach.

molecule are generated and then discretized to a valid molecule (De Cao et al., 2018; Madhawa et al., 2019).

Rule-based models generate molecules by applying a set of pre-defined graph transformation rules to combine a set of starting compounds. By iteratively applying these rules, the model is able to generate new molecular structures. The BRICS (Degen et al., 2008) method provides a set of molecular fragments and rules how to meaningfully combine them. Jensen (2019) defines graph mutation and crossover operations to generate new molecules, which are then applied in the context of molecule optimization. Transformation rules are well suited to describe chemical reactions. The DOGS method (Hartenfeller et al., 2012) generates new molecules by applying chemical reaction rules to a set of starting molecules, which allows the generation of new molecules with known synthetic routes. Rule-based models are often combined with machine learning models to predict the outcome of chemical reactions (Segler et al., 2017; Segler et al., 2018b; Fortunato et al., 2020).

1.2.3 Distribution-learning

Distribution-learning is a fundamental application of generative models in drug design. Its objective is to create a model that captures the distribution of molecules within a dataset. These models can be trained on large chemical libraries of stable

molecules, PubChem (Kim et al., 2016), ChEMBL (Bento et al., 2014) or ZINC (Irwin et al., 2012). Using this process the resulting models can learn what reasonable molecules look like in the context of drug discovery and learn the syntax of the selected molecular representation. This makes them useful for their two main purposes: they can expand virtual libraries and, more crucially, act as a foundation for other applications such as goal-directed generation, which we will explore in the next section.

In recent years there has been a surge in interest distribution-learning models based on deep neural networks. Many architectures and training strategies originally proposed for text and image generation have been adapted and specialized to generate molecules. While all of them aim to approximate the data distribution, they differ in the way they model the distribution and the choice of molecular representation.

Autoregressive models make use of the chain rule of probability to factorize the likelihood of a molecule into a product of conditional probabilities of the individual tokens in the molecule. The model is trained maximizing the likelihood of the training data with respect to the model parameters. Autoregressive models are explicit density models, as the likelihood of sampling a molecule can be calculated exactly, which facilitates model evaluation. Models trained in this way form the backbone of many generative models in drug discovery (Gómez-Bombarelli et al., 2018; Segler et al., 2018a; Olivecrona et al., 2017; Guo et al., 2023; Thomas et al., 2022b; Jaques et al., 2016; Cohen-Karlik et al., 2024).

Variational autoencoders (VAEs) (Kingma et al., 2013) generate molecules by first sampling from a simple latent distribution and then mapping the samples to molecular space via a probabilistic decoder network. To make training tractable a second network, the encoder network is used to map the data to the latent space. The model is then trained to maximize the evidence lower bound (ELBO) of the data. This model has the advantage of providing a continuous latent space, which can be used to interpolate between molecules and allows the use continuous optimization algorithms for molecule optimization. VAEs belong in the class of approximate density models, as the likelihood of a given molecule can be calculated approximately via Monte Carlo sampling. VAEs have been a popular choice for generating molecules (Gómez-Bombarelli et al., 2018; Kusner et al., 2017; Simonovsky et al., 2018; Samanta et al., 2018; Jin et al., 2018; Dai et al., 2018; Liu et al., 2018).

Generative flows (Rezende et al., 2016) are based on the idea of learning a bijective mapping between molecular space and a latent space. Generative flows transform a simple latent space distribution to the data distribution via a sequence of invertible transformations parameterized by neural networks. This makes it possible to calculate the likelihood of a given molecule using the change of variables formula of probability. Similarly to autoregressive models, generative flows are explicit density models, and are optimized by maximizing the likelihood of the training data. Originally generative flows have been proposed for continuous data, but have been adapted to discrete data such as molecules by using a continuous relaxation of the molecule (Madhawa et al., 2019). Similarly to VAEs, generative flows provide a continuous latent space, which can be used for molecule optimization.

Generative adversarial networks (GANs) (Goodfellow et al., 2014) are latent space models that map a simple distribution in latent space to molecular space, but rely on a game-theoretic approach to training. A generator network is trained to generate data, which is then fed to a discriminator network. The two networks then engage in a minimax game, where the discriminator is trained to distinguish between real and generated data, while the generator is trained to generate samples that fool the discriminator. GANs are implicit density methods they sample from the model distribution, but do not provide a likelihood for a given sample. This approach has been combined with different generation strategies in the context of drug discovery (De Cao et al., 2018; Kadurin et al., 2017; Guimaraes et al., 2017; Méndez-Lucio et al., 2018; Tang et al., 2024). GANs also provide a continuous latent space but in contrast to VAEs and generative flows, the latent representation of a given molecule cannot be calculated, as the generator is a one-way mapping.

1.2.4 Goal-directed molecule generation

Goal-directed molecule generation (Schneider, 2013) is a computational approach for automatically designing molecules with desired property profiles. The desired property profile can be a single molecular property or a combination of multiple properties. In this thesis we assume that we are given a scoring function which assigns a score to each molecule based on whether it satisfies the desired property profile. The goal of goal-directed generation is to find molecules with the highest possible score.

Goal-directed generation expands upon *virtual screening* (*VS*), which searches for molecules with the desired profile in a library of molecules. Walters (2019) estimates

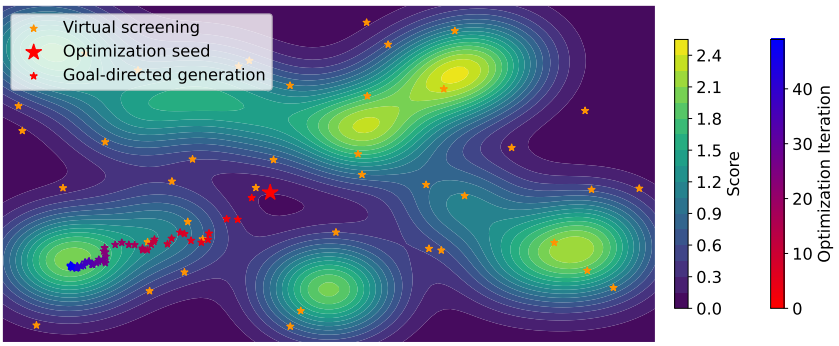


Figure 1.6: Illustration of the difference between goal-directed molecule generation and virtual screening in a 2D chemical space, where each point represents a molecule. The background color represents the molecules' scores. Goal-directed generation works akin to a numerical optimization algorithm and efficiently finds high-scoring molecules, shown in the transition from red to blue stars. In contrast VS amounts to a random search in chemical space, which is less efficient and is likely to miss high-scoring regions of chemical space.

that approximately 10^{13} molecules can be routinely tested in a *VS* experiment. While this number can vary significantly depending on the computational cost of running the *quantitative structure-property relationship* (*QSPR*) model, it is dwarfed by the size of drug-like chemical space, which is estimated to contain between 10^{30} and 10^{60} molecules (Walters, 2019; Ruddigkeit et al., 2012). Consequently, *VS* is limited to exploring only a small fraction of chemical space and cannot fully leverage the vast number of possible candidates that drug-like chemical space offers.

Goal-directed generators address this limitation of *VS* by focusing the search on the most relevant parts of chemical space. In contrast to the random search approach taken by *VS*, goal-directed generators act more like optimizers that are able to efficiently locate maxima as illustrated in Figure 1.6. This is achieved by an iterative process in which a model generates a set of molecules, which are then scored by a *QSPR* model. These scores are then used to update the model, shifting the sampling distribution to regions of chemical space with higher scores.

Recently, there has been a surge of deep learning-based goal-directed generators (Elton et al., 2019; Sanchez-Lengeling et al., 2018; Du et al., 2024). A multitude of different models have been proposed, which are based on a variety of neural network architectures, training strategies and molecular representations. These methods augment traditional rule-based generation approaches that have been combined with graph search and evolutionary algorithms. (Schneider et al., 2005; Schneider, 2013). The new wave of deep-learning methods has shown great promise in generating novel molecules with desired property profiles and has led to success

in a variety of applications, such as the design of new drugs, materials or catalysts (**todo**).

Some of the most commonly used approaches to goal-directed molecular generation are:

- **Hill-climbing** (Segler et al., 2018a; Xie et al., 2021; Thomas et al., 2022b) is a simple optimization algorithm that relies on an underlying distribution-learning model. Molecules are sampled from the model's distribution and their scores are evaluated. The model is then retrained on the top-scoring molecules and the process is repeated.
- **Reinforcement learning** uses the scores of generated molecules as a reward signal to update the model distribution. This is achieved through methods based on the REINFORCE algorithm (Williams, 1992) which allows to update the model distribution in a way that increases the scores of the generated molecules (Olivecrona et al., 2017; Thomas et al., 2022b; You et al., 2019; Guo et al., 2023).
- **Genetic algorithms** in molecular generation operate by evolving an initial population of molecules through iterative cycles of mutation, crossover, and selection (Jensen, 2019; Nigam et al., 2021; Yoshikawa et al., 2018). Starting from an initial set of molecules, new molecules are generated by applying mutation and crossover operations. The molecules are then scored, and the best ones are selected for the next generation. This process is repeated for multiple generations, gradually optimizing the population towards desired molecular characteristics.
- **Tree search** methods build a tree of possible molecules by recursively applying a set of transformation rules to some initial molecules. Using techniques such as Monte Carlo Tree Search, the tree is explored to find the most promising molecules (Yang et al., 2017; Jensen, 2019).
- **Continuous optimization** employ classical optimization algorithms in the continuous latent space of (variational) autoencoders (Gómez-Bombarelli et al., 2018; Kusner et al., 2017; Winter et al., 2019) or generative flows (Madhawa et al., 2019). If the scoring function can be evaluated in the continuous space, these methods can be used to directly optimize the molecular properties, without the need for sampling the molecular graph.

- **Generative Flow Networks** (Bengio et al., 2021) aim to generate molecules with probability proportional to their score. This method relies on an iterative generation process and models chemical space as a directed acyclic graph, with nodes being intermediate molecules and edges graph edits. The transition probabilities between nodes are given by a "flow" of probability mass from the root node to finished molecules, such that the probability of each finished molecule is proportional to its score. This has the advantage of being able to explore multiple modes of the scoring function.

1.2.5 Challenges in Evaluating Generative Models in de Novo Design

The evaluation of generative models is a challenging task, as generative tasks usually allow for a range of valid solutions. This makes it hard to define a single ground truth, on which model evaluation usually relies on. In this section we discuss some of the challenges in evaluating generative models in the context of de novo design.

1.2.5.1 Evaluation of distribution-learning models

The most basic and commonly used checks to assess the quality of the generated compounds are the validity, uniqueness and novelty of the generated molecules. A molecule is valid if it obeys chemical valence rules, which is usually checked using chemoinformatics toolkits such as RDKit (Landrum, 2006). The uniqueness of a set of molecules measures the fraction of unique molecules in the set and can flag models that output many duplicates. The novelty a set of generated molecules is the fraction of molecules that are not in the training set and can, to a certain extent, detect whether a model overfits to the training set.

A variety of approaches exists to assess how well a model can learn the distribution of the training set. Explicit/approximate density models allow principled evaluation based on the likelihood on a hold-out test set. However, this is not applicable for implicit density models such as GANs. The KL-divergence between the distributions of scalar molecular properties (e.g. molecular weight or logP) of the generated molecules and the training set is a commonly used metric to evaluate the distribution fit (Brown et al., 2019), but is usually determined using a limited number of properties. The Frechet ChemNet Distance (FCD) (Preuer et al., 2018) provides a more comprehensive check of the distribution fit. The FCD compares the

distributions of the activations of a neural network trained to predict bioactivities and has been shown to be sensitive to distributional differences in many different molecular properties.

The MOSES (Polykovskiy et al., 2020) and GuacaMol (Brown et al., 2019) benchmarks provide standardized frameworks for evaluating distribution-learning models in molecular generation. While these benchmarks represent progress in assessment methodology, questions remain about their comprehensiveness and ability to fully capture the complexities of molecular generation tasks in drug discovery contexts.

1.2.5.2 Goal-directed optimization of ML-based scoring functions

Machine learning models have shown promise in predicting molecular properties (Mayr et al., 2016; Klambauer et al., 2019; Vamathevan et al., 2019; Chen et al., 2018; Stokes et al., 2020), which makes them an attractive target in goal-directed generation. However, the fact that such machine learning models are trained on limited amounts of experimental data, adds additional aspects to a proper model evaluation. In this setting there already are known molecules with high scores that were used to train the scoring function. The task thus becomes to find *novel* high-scoring molecules that differ from the training data. Machine learning models, however, are often biased towards their training data, which might lead to a lack of novelty in the generated molecules. The extent of this bias and its impact on molecular novelty and how to quantify it remains an open question.

Furthermore, it has been shown that optimizing an ML model's output with respect to its input can lead to generated samples that incorrectly receive high scores from the model (Szegedy et al., 2014; Goodfellow et al., 2015). This can happen when the optimization leaves the applicability domain of the model, where the scoring function is no longer reliable but scores can still be high. Especially for scoring functions trained on small datasets, the probability that the model drifts outside the applicability domain is high. While this effect was initially studied in the image domain, where human vision can easily provide ground truth evaluation, it is more challenging to identify this effect in molecular optimization. It remains unclear whether this issue extends to the context of goal-directed molecule generation and how to quantitatively assess it.

1.2.5.3 Diversity of generated molecules

Generating diverse sets of high scoring molecules can increase the success chances of a drug discovery project (Martin, 2001; Gorse, 2006). Having multiple high-scoring molecules provides some insurance against the uncertainties and incomplete scope of the scoring function and the conducted experiments. Given that it is expected that some of the generated molecules will fail in downstream testing, it is important to have a multitude of candidates. Diversity among those encourages uncorrelated outcomes in downstream testing, which increases the chances of finding at least one successful candidate. In essence, a varied molecular portfolio serves as a hedge against the inherent uncertainties in the modeling and experimental outcomes.

The concept of diversity in molecular generation is complex, with some of the most commonly used diversity metrics being inadequate for goal-directed generation. The most arguably most popular diversity metric in the context of molecular generation is the internal diversity, which measures the average pairwise distance between molecules. However, this metric has been shown to be inadequate for goal-directed generation (Waldman et al., 2000; Xie et al., 2021; Thomas et al., 2021). Thomas et al. (2021) proposed the sphere exclusion diversity (SEDiv) metric, which aligns better with chemical intuition but can be misleading for differently sized sets.

Xie et al. (2023) recently addressed these shortcomings and introduced the #Circles metric which better reflects the requirements of evaluating goal-directed generators. #Circles is a non-normalized version of SEDiv and provides an absolute measure of the chemical space covered by the generated molecules. It is motivated by an axiomatic approach similar to the one used in (Waldman et al., 2000) and has been shown to correlate better with the functional diversity of the generated molecules. While initial evaluations of goal-directed generators using #Circles have been conducted, most comparisons are limited by the fact that the models were not adapted to the diverse optimization setting.

A comprehensive comparison of models specifically designed for diverse optimization is still missing, leaving open the question of how well different approaches perform in generating diverse, high-scoring molecules.

1.2.5.4 Standardized Computational Resources

A frequently neglected aspect in evaluating goal-directed models is the use of standardized computational resources. At its core, goal-directed generation is a

search problem that—given infinite resources—can be solved through exhaustive enumeration of drug-like chemical space. Consequently, the primary challenge in goal-directed generation lies in identifying high-scoring molecules within reasonable computational budgets.

However, many studies compare different models without accounting for this crucial factor, potentially leading to biased comparisons. This might lead to scenarios where generators running for days are compared to models that run for minutes, which can lead to unfair comparisons. Recently, this issue has gained increased attention, after (Gao et al., 2022) proposed a benchmark that measures the sample efficiency of goal-directed generation algorithms, arguing that in practical applications, the bottleneck is the number of scoring function evaluations. Other researchers have adapted to this approach (Thomas et al., 2022a; Thomas et al., 2022b; Guo et al., 2023), putting a stronger emphasis on controlling for computational budgets.

However, sample efficiency is most relevant when using scoring functions that are expensive to evaluate. In many cases, the cost of evaluating the scoring function is on the same order of magnitude or smaller than the cost of generating the molecules. In this case, the computational budget needs to account for both the generation and scoring of molecules. Consequently, there is a need for a more comprehensive evaluation framework for goal-directed generation models that controls for various types of computational budgets, especially in the context of finding diverse high-scoring molecules.

1.2.6 Retrosynthesis prediction

Drug candidates, whether designed by generative models or other means, need to be synthesized for testing and eventually for use in patients. However, finding a synthesis route for a given molecule can be a complex and time-consuming process. *Computer-aided synthesis planning* (*CASP*) methods help chemists to find synthesis routes, enabling synthesis of previously inaccessible molecules or making synthesis more efficient and cheaper.

This problem is often approached using a retrosynthesis approach (Corey et al., 1969; Corey, 1991), which recursively deconstructs the target molecule into simpler precursors until they match available starting materials. At each step, single-step retrosynthesis prediction models suggest sets of reactants that could theoretically combine to produce the current, intermediate molecule. The success of retrosynthesis

planning hinges on highly accurate chemical reaction models, as these ensure that the proposed synthetic routes are actually feasible.

Early work in retrosynthesis prediction relied on carefully curated expert rules encoding possible reactions. Recently, machine learning models that learn the patterns of chemical reactions from examples stored in reaction databases have received increased attention (Coley et al., 2018). One line of work relies on sequence-to-sequence originally developed for machine translation, to predict the SMILES strings of reactants given the that of the target molecule (Schwaller et al., 2019; Nam et al., 2016; Schwaller et al., 2018; Karpov et al., 2019; Tetko et al., 2020). Another set of approaches exploit the fact that connectivity in a reaction is often preserved, and use graph neural networks to edit the connectivity of the target molecule in order to yield possible reactants (Sacha et al., 2020; Shi et al., 2020; Somnath et al., 2020; Yan et al., 2020).

Template-based methods represent another approach to retrosynthesis prediction (Segler et al., 2017; Segler et al., 2018b; Dai et al., 2020; Sun et al., 2020). These models rely on a set of graph transformation rules, or templates, which encode the connectivity changes that occur in a reaction. These templates can either be extracted from reaction databases or created by chemists. Template-based methods are then trained to predict which templates can be applied to which target molecules using machine learning models trained on reaction databases. Finally, given a target molecule of interest, the model returns the templates most likely to lead to feasible reactions, which can be used to suggest synthesis routes.

While template-based methods have shown excellent performance in retrosynthesis prediction, they face challenges with rare templates. Template extraction from databases often leads to many templates being represented by only a few training samples(Fortunato et al., 2020), resulting in a few-shot learning problem where models struggle to perform well on these uncommon templates. While some strategies have been proposed to alleviate this issue, such as data augmentation (Fortunato et al., 2020) and specialized architectures and training objectives (Dai et al., 2020), the problem remains a challenge in the field.

1.3 Contributions

1.3.1 Identifying Failure Modes in Generative Model Evaluation

In (Renz et al., 2019b) we investigate possible failure modes in the evaluation of distribution-learning and goal-directed generative models. We show that the distribution-learning benchmark proposed in GuacaMol (Brown et al., 2019) is not able to distinguish recently published generative models from a simple baseline model. The tested generative models underperform or only marginally outperform the baseline model. While this does not necessarily mean that the generative models are not useful, it calls for a more comprehensive evaluation of distribution-learning models.

In the context of goal-directed optimization we introduce control scores that give information whether the optimization leads to the generation of molecules that are biased towards the training data and whether the optimization overfits to the used scoring function. The control scores are obtained by retraining the scoring function on a hold-out set of the training data or using a different random initialization, and serve as a diagnostic tool to detect these issues.

We show that the generated molecules are biased towards the high-scoring molecules in the training set, which leads to a lack of novelty in the generated molecules. We also show that the generative models are able to overfit to the scoring function's random initialization. This might lead to the generation of molecules that wrongly receive high scores from the scoring function, leading to an overestimation of the models' performance. Section 2.1 reprints the corresponding publication.

1.3.2 Diversity-based comparison of goal-directed generators

In (Renz et al., 2024) we introduce a benchmark for diverse optimization that addresses the above-mentioned issues. In this benchmark, we evaluate the diversity of the generated molecules using a recently proposed diversity metric #Circles (Xie et al., 2023). We compare the performance of diverse optimization approaches under two different compute budgets, namely a fixed number of scoring function evaluations and a fixed time budget. The first setting is relevant for applications where the cost of evaluating the scoring function dominates the optimization process,

while the second setting is relevant for scoring functions that are relatively cheap to evaluate. Using this setup we test 14 goal-directed optimization methods and show how SMILES-based auto-regressive models dominate the benchmark. Section 2.2 reprints the corresponding publication.

1.3.3 Improving few-shot and zero-shot retrosynthesis prediction

In (Seidl et al., 2022) we propose a novel approach to template-based retrosynthesis prediction. We use a multimodal learning approach that learns to associate relevant templates to product molecules using a Modern Hopfield Network (Ramsauer et al., 2020). In contrast to previous template-based methods, our model processes the structure of templates and can make use of similarities between them. This allows for improved generalization, especially for templates with few training samples and even for unseen templates. This model is several times faster than comparable methods and shows good predictive performance. Section 2.3 reprints the corresponding publication.

1.4 List of publications

This thesis comprises the work published in the following papers:

- P. Renz et al. (2019b). "On Failure Modes in Molecule Generation and Optimization". In: *Drug Discovery Today: Technologies*. Artificial Intelligence 32–33, pp. 55–63. DOI: 10.1016/j.ddtec.2020.09.003
- P. Renz et al. (2024). "Diverse Hits in De Novo Molecule Design: Diversity-Based Comparison of Goal-Directed Generators". In: *J. Chem. Inf. Model.* DOI: 10.1021/acs.jcim.4c00519
- P. Seidl et al. (2022). "Improving Few- and Zero-Shot Reaction Template Prediction Using Modern Hopfield Networks". In: *J. Chem. Inf. Model.* 62.9, pp. 2111–2120. DOI: 10.1021/acs.jcim.1c01065

Other Publications Besides the papers listed above, I have also contributed to the following publications:

- K. Preuer et al. (2018). "Fréchet ChemNet Distance: A Metric for Generative Models for Molecules in Drug Discovery". In: *J. Chem. Inf. Model.* 58.9, pp. 1736–1741. DOI: 10.1021/acs.jcim.8b00234
- P. Renz et al. (2019a). "Uncertainty Estimation Methods to Support Decision-Making in Early Phases of Drug Discovery". In: *NeurIPS-2019 Workshop on Safety and Robustness in Decision Making*
- M. Hofmarcher et al. (2020). *Large-Scale Ligand-Based Virtual Screening for SARS-CoV-2 Inhibitors Using Deep Neural Networks*. DOI: 10.48550/arXiv.2004.00979. arXiv: 2004.00979 [cs, q-bio, stat]. Pre-published
- P. Renz et al. (2023). "Low-Count Time Series Anomaly Detection". In: *2023 IEEE 33rd International Workshop on Machine Learning for Signal Processing (MLSP)*. 2023 IEEE 33rd International Workshop on Machine Learning for Signal Processing (MLSP), pp. 1–6. DOI: 10.1109/MLSP55844.2023.10285979

Publications

2

This chapter presents publications as originally published, reprinted with permission from the corresponding publishers. The copyright of the original publications is held by the respective copyright holders. In order to fit the paper dimension, reprinted publications may be scaled in size and/or cropped.

2.1 On Failure Modes in Molecule Generation and Optimization

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.

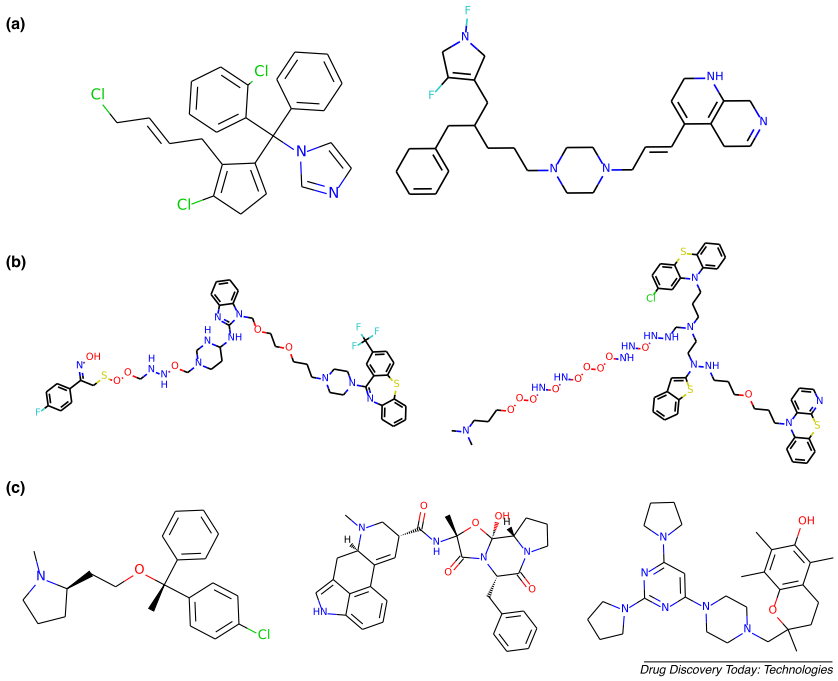


Figure 1. High scoring compounds generated for the DRD2 task described below, with actives from the training set for comparison. The generated compounds show unwanted substructures. (a) Compounds created by a graph-based genetic algorithm. The left compound contains a reactive diene. The right one contains 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

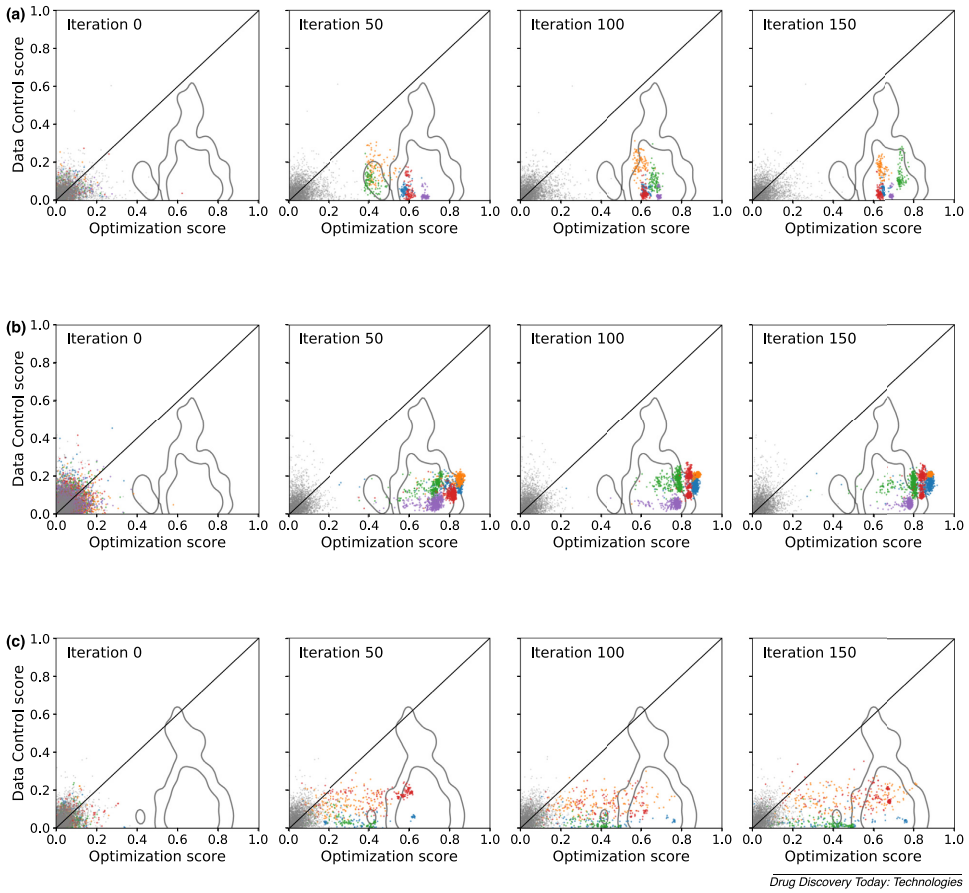


Figure 2. Scatter plots of OS vs. DCS on molecules during the course of training for the EGFR task and optimizer (a) GA, (b) LSTM and (c) PS. Each point represents a compound. Different colors correspond to different runs. Grey points are random compounds from ChEMBL. The contour lines approximately 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

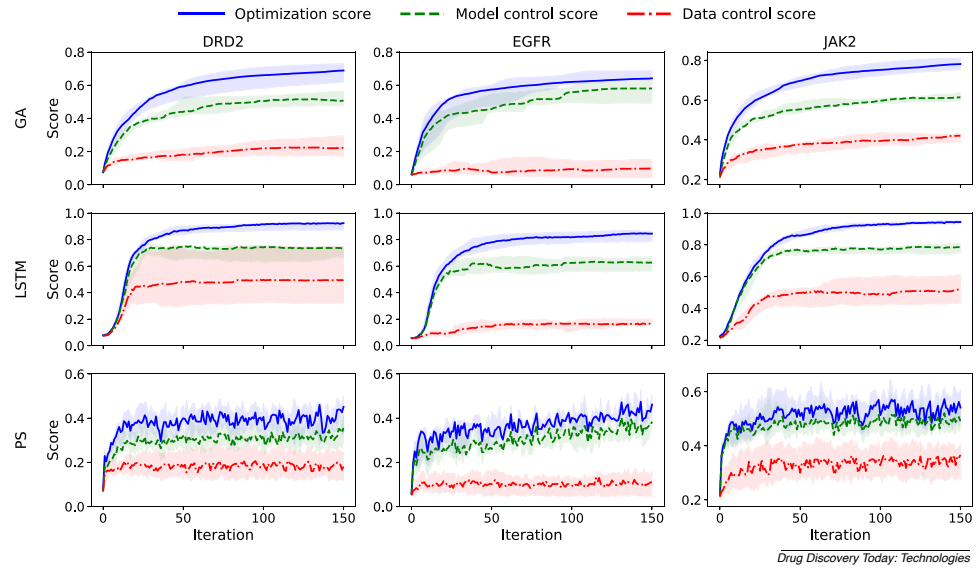


Figure 3. Scores during optimization. For each curve we first took the mean of the scores for each run. The bold line corresponds to the median over those means while the shaded areas correspond to the interquartile range.

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.

Supporting information for "On failure modes in molecule generation and optimization"

Philipp Renz[1], Dries Van Rompaey[2], Jörg Kurt Wegner[2], Sepp Hochreiter[1], and 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

Contents

List of Figures

List of Tables

Target	ChEMBL ID	Active	Inactive	ROC AUC	AP	Fraction pos.
JAK2	CHEMBL3888429	140	527	0.78±0.03	0.44±0.05	0.21
EGFR	CHEMBL1909203	40	802	0.72±0.08	0.15±0.05	0.05
DRD2	CHEMBL1909140	59	783	0.85±0.03	0.53±0.09	0.07

Table S1: Information on the data sets. Area under ROC curve (ROC AUC) and average precision (AP) show the performance of the trained classifiers on hold out test data. The last column denotes the fraction of positive samples which serves as a random baseline for AP.

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 [1] is calculated using rdkit [2]. 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 Table S1. To obtain binary labels for the JAK2 task we labelled all compounds with a pIC_{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 [3]. 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 [3]. For the hill-climbing SMILES-LSTM we used these settings:

- `n_epochs`: 151 (Number of iterations)
- `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)

For the particle swarm optimization we used the implementation provided by the authors and the following settings:

- `init_smiles`: Randomly drawn list of molecules from the same set as used for GA (Molecules from which to start optimization)
- `num_part`: 200 (Number of particles)
- `num_swarms`: 1 (Number of swarms)

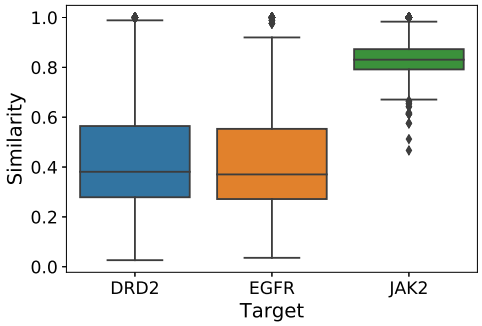


Figure S1: Distribution of similarities to the nearest neighbour in split 1 of compounds in split 2.

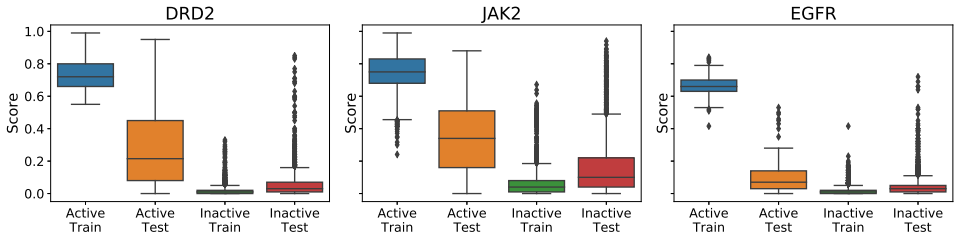


Figure S2: Distribution of scores for different groups of compounds. The distribution of scores of actives from the training set is shifted positively with respect to those from the test set, and vice versa for inactives.

S2 Additional Results

S2.1 Additional information on data splits

Fig. S1 shows the distribution of similarities of compounds in split 2 to the nearest neighbour in split 1. For the JAK2 task many compounds have a very similar neighbour in the other split.

S2.2 Training data bias of classifiers

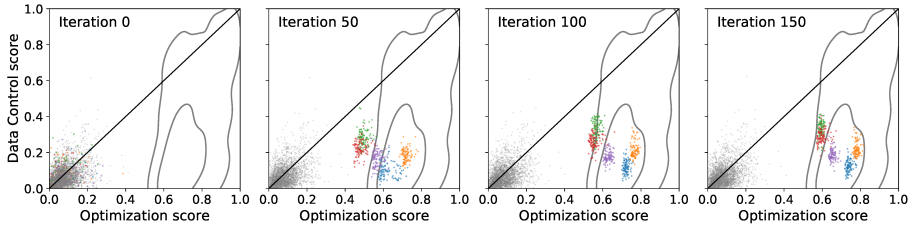
Fig. S2 shows the distribution of scores for different groups of compounds. Actives from the training set obtain a higher median score than those from the test set by the trained classifier. Conversely, inactives from the training set obtain a lower median score than those from the test set. These results are aggregated using ten different random seeds for splitting the data and for fitting the random forest.

S2.3 Complementary results

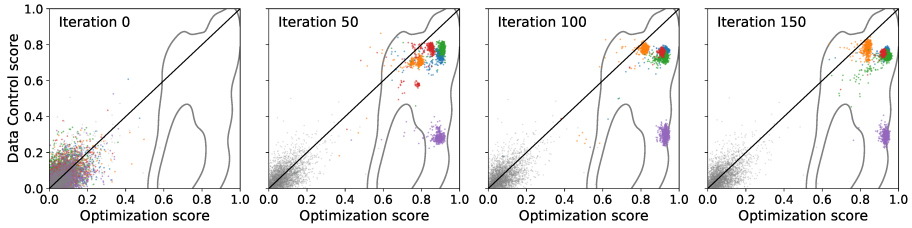
This section contains complementary results for the optimizers and tasks not shown in Section "Failure mechanisms in goal-directed generation" and includes Figs. S3, S4, S5 and S6.

S2.4 Additional information on optimized molecules

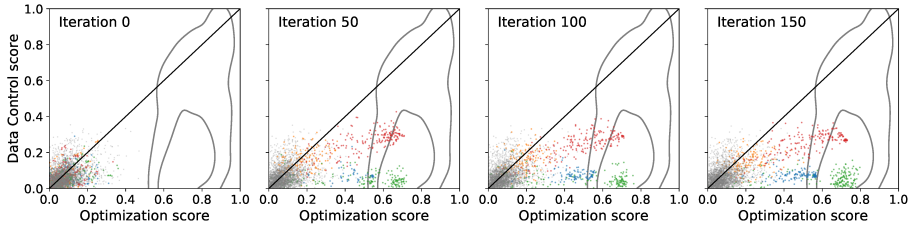
Fig. S9 shows additional metrics over the course of optimization. These metrics include:



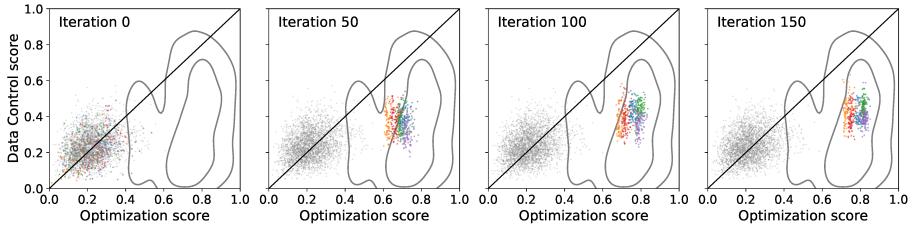
(a) GA on DRD2



(b) LSTM on DRD2



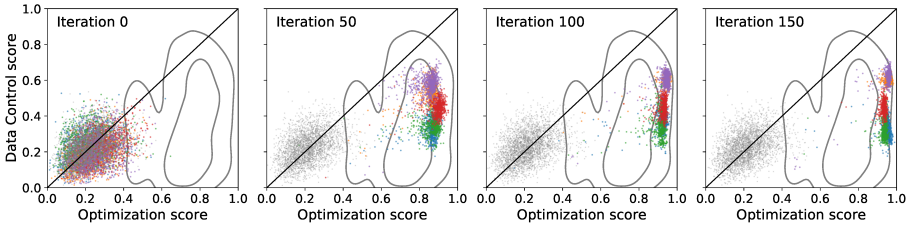
(c) PS on DRD2



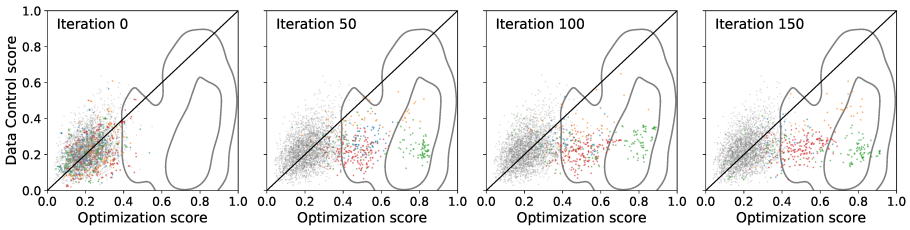
(d) GA on JAK2

#	SMARTS	Counts	rel. freq.
1	O-O-O	1	6.3e-07
2	ON([H])O	4	2.5e-06
3	N-F	2	1.3e-06
4	S-O-O	12	7.5e-06

Table S2: Uncommon small substructures contained in Fig. 1 and how often they occur in 1,591,378 ChEMBL compounds. The columns provide the substructure as SMARTS pattern, the absolute ("Counts") and the relative frequency ("rel. freq.") of occurrences of this pattern.

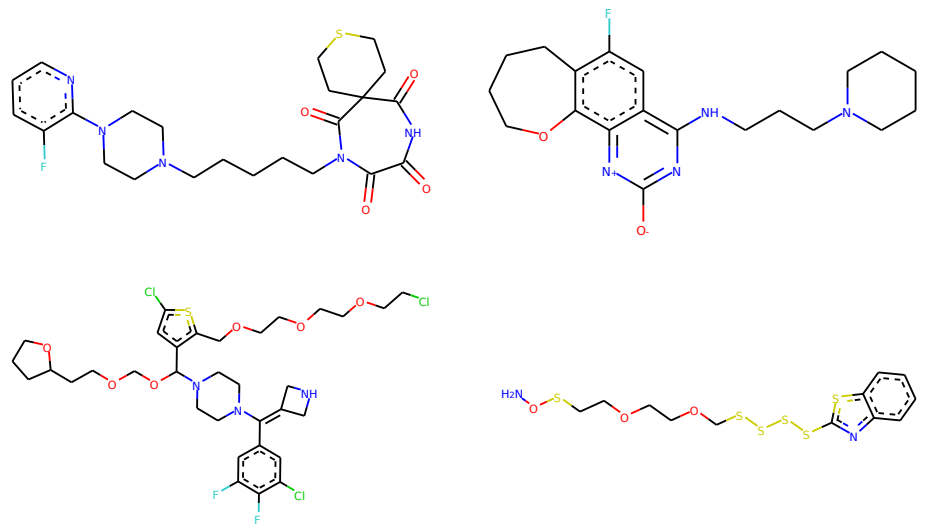


(e) LSTM on JAK2

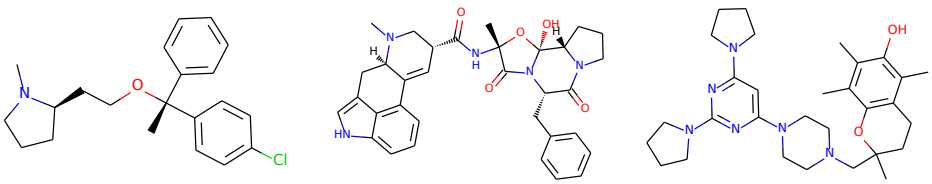


(f) PS on JAK2

Figure S3: 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.

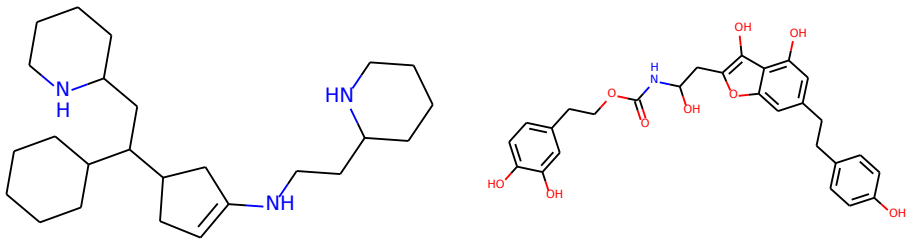


(a) Compounds generated by PS

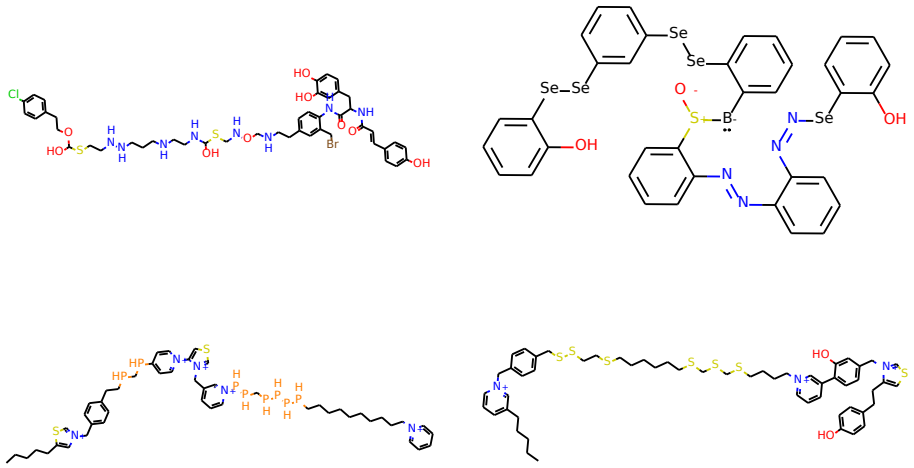


(b) Reference compounds from the training set

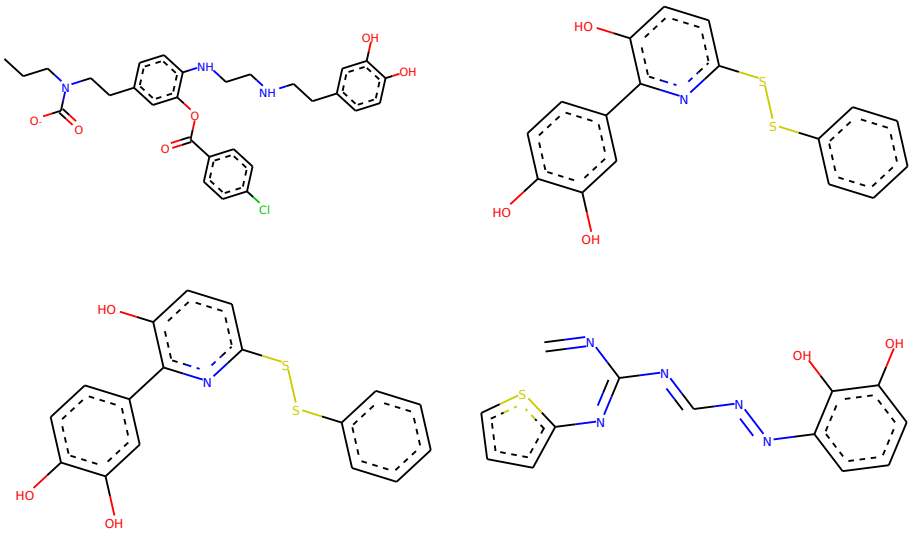
Figure S4: High scoring compounds generated for the DRD2 task by PS, with actives from the training set for comparison.



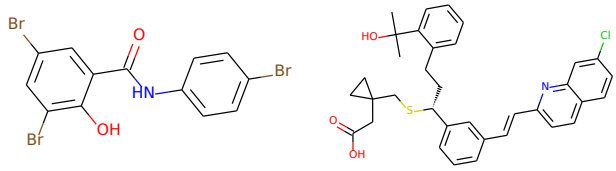
(a) Compounds generated by a graph-based GA



(b) Compounds generated by a SMILES-based LSTM

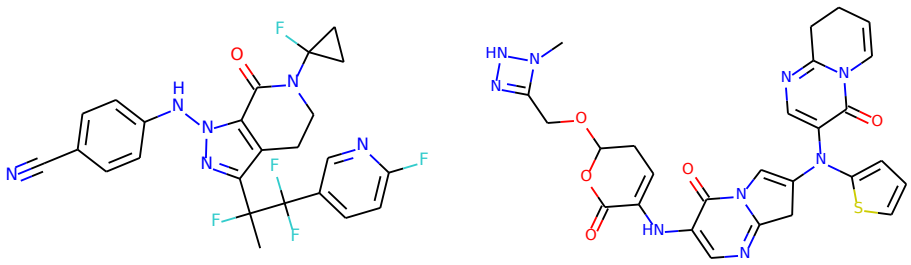


(c) Compounds generated by PS

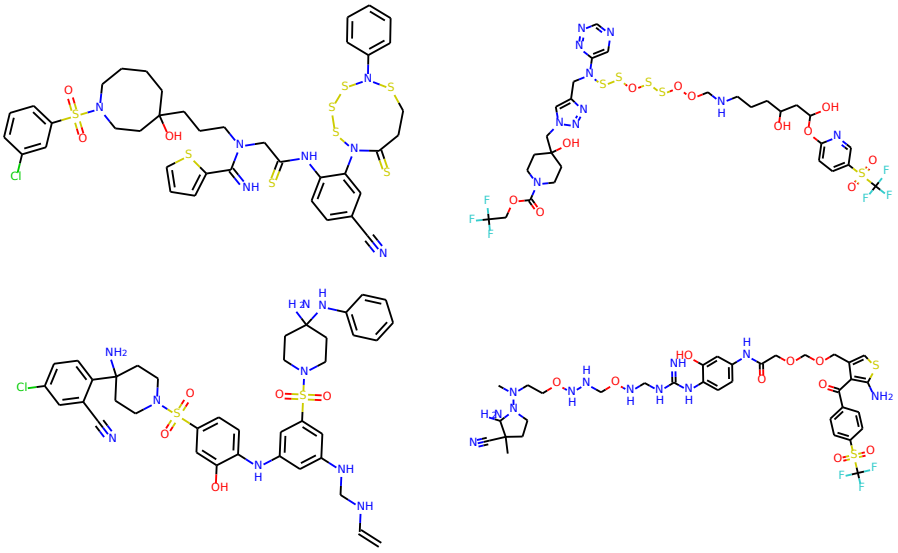


(d) Reference compounds from the training set

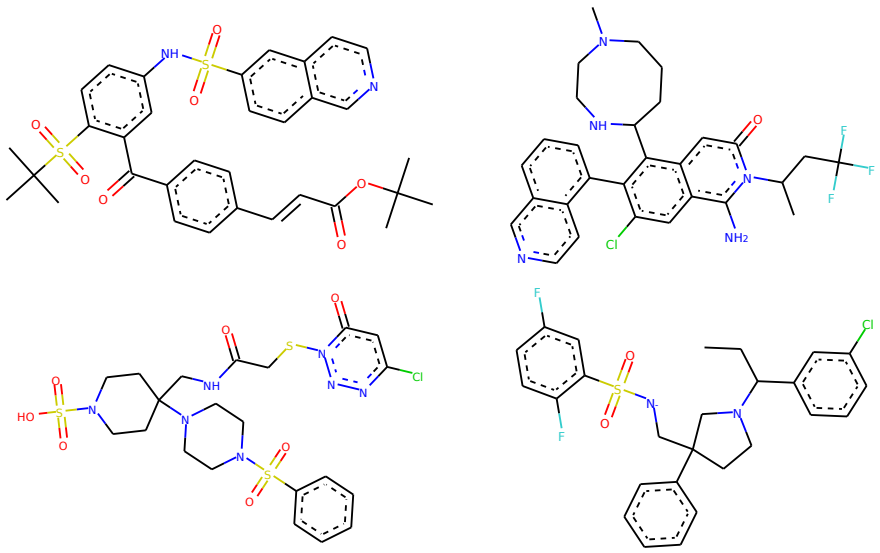
Figure S5: Random samples generated for the EGFR task, with actives from the training set for comparison.



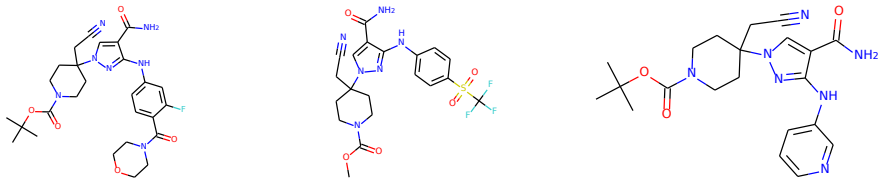
(a) Compounds generated by a graph-based GA



(b) Compounds generated by a SMILES-based LSTM

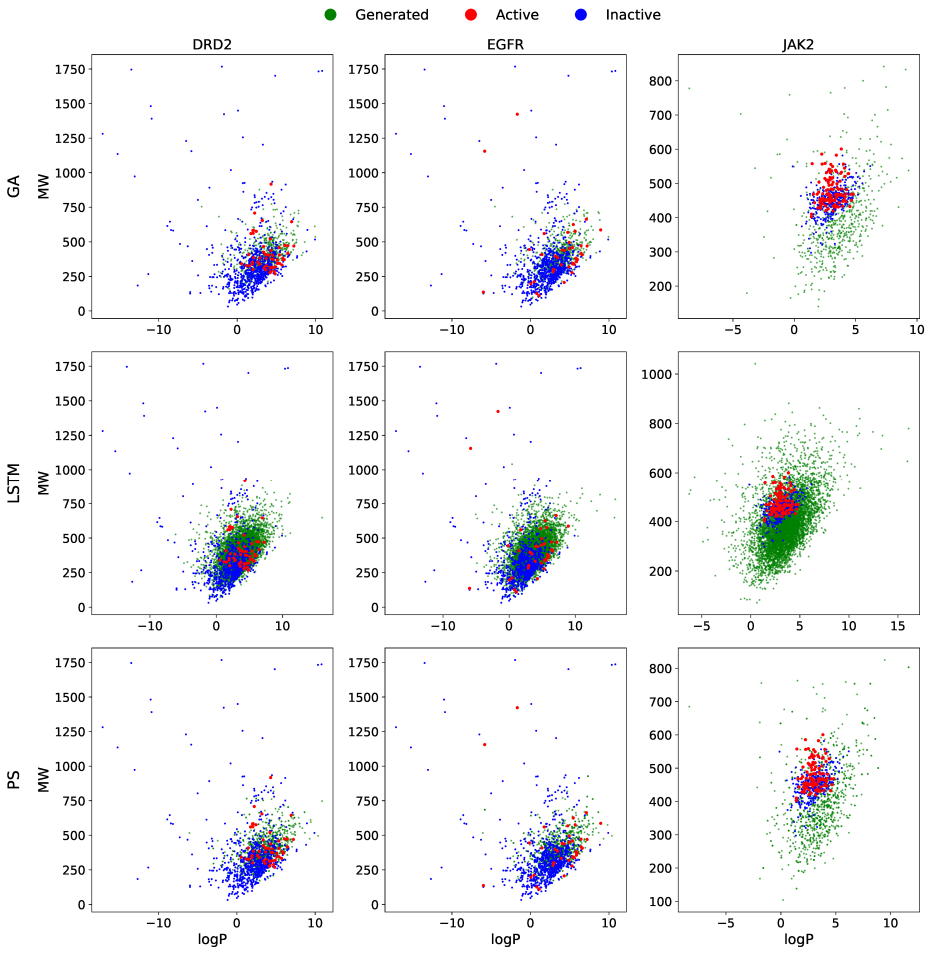


(c) Compounds generated by PS

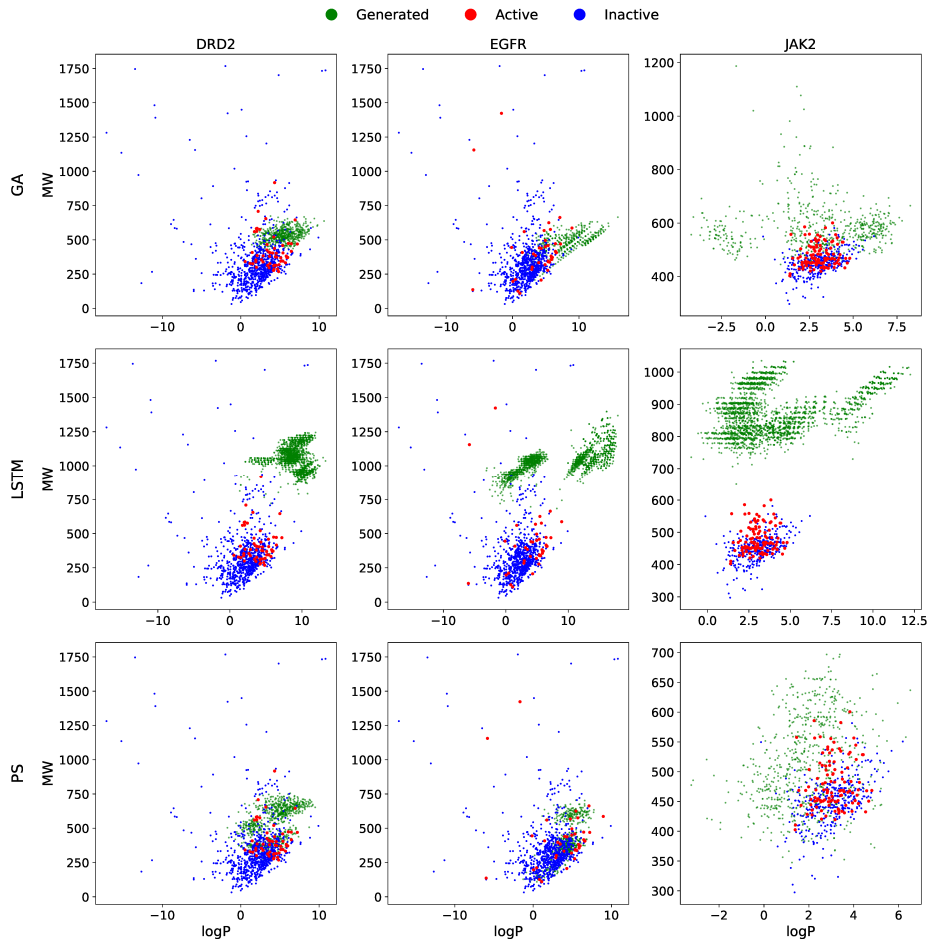


(d) Reference compounds from the training set

Figure S6: Random samples generated for the JAK2 task, with actives from the training set for comparison.

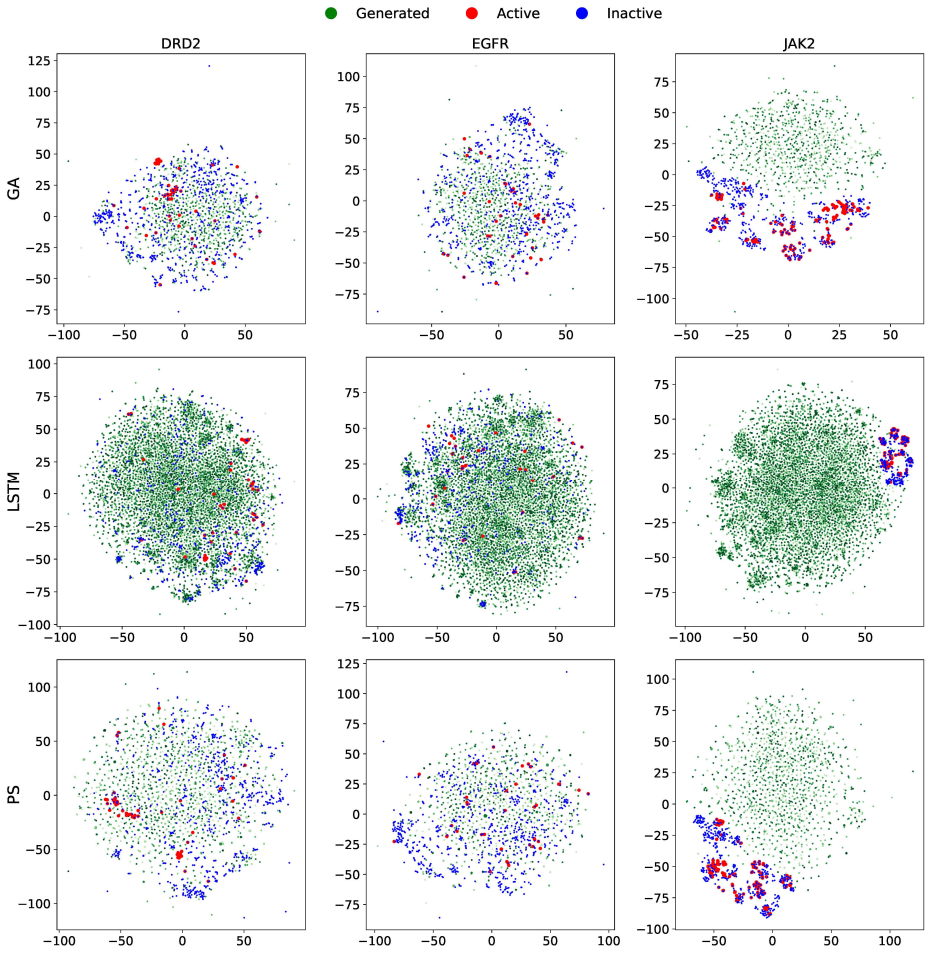


(a) First iteration

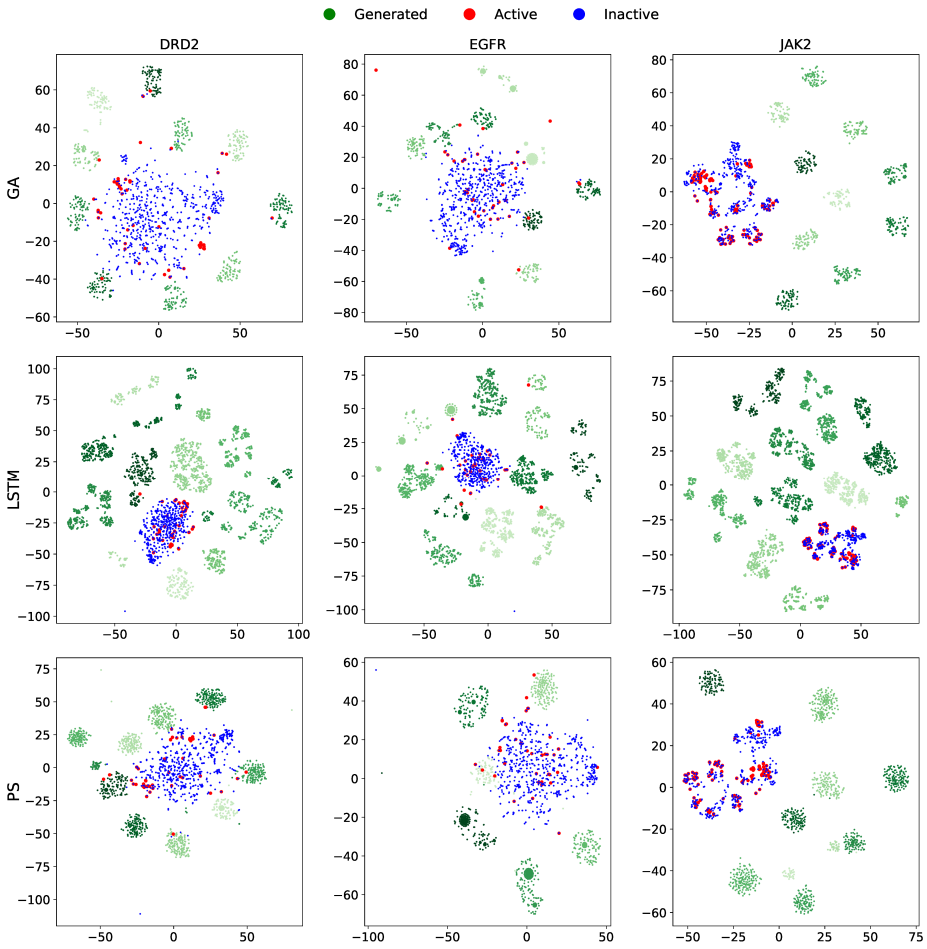


(b) Last iteration

Figure S7: Visualization of logP/MW-values for generated compounds, known actives and known inactives for different optimizers and tasks at the first and last iteration of optimization.



(a) First iteration



(b) Last iteration

Figure S8: t-SNE embedding for generated compounds, known actives and known inactives for different optimizers and tasks at the first and last iteration of optimization. The embedding is based on Tanimoto distance on the same ECFP4 fingerprints as used for training the classifiers. We used the t-SNE implementation from scikit-learn v0.22.1 [4] with standard settings and precomputed metric. The different shades for the generated molecules indicate independent runs.

- logP: water-octanol partition coefficient calculated according to [5] as implemented in [2]
- MW: molecular weight as implemented in [2]
- Length: length of SMILES strings
- Diversity: mean pairwise Tanimoto distance between generated molecules
- NN act: mean Tanimoto distance of generated molecules to nearest training active
- NN ina: mean Tanimoto distance of generated molecules to nearest training inactive

For all average Tanimoto distances we used a subset of size 100 of the generated compounds for computational reasons and binary ECFP4 fingerprints calculated in rdkit [2]. For all curves we first calculated the mean over the generated compounds in a run and then calculated the median and interquartile ranges depicted based on them.

There is a prominent increase in MW and SMILES length for the LSTM optimizer, of which the latter reaches its upper bound of 100 as stated in the optimizer settings. This trend can also be observed for the genetic algorithm, but to a lesser degree.

The diversity among the generated molecules decreases during optimization for LSTM and GA, while for PS it drops in the first iterations and then follows a more constant trend. The average distance to training actives also drops for GA and PS during optimization. For the LSTM an interesting pattern can be observed, where this metric first de- and then increases again.

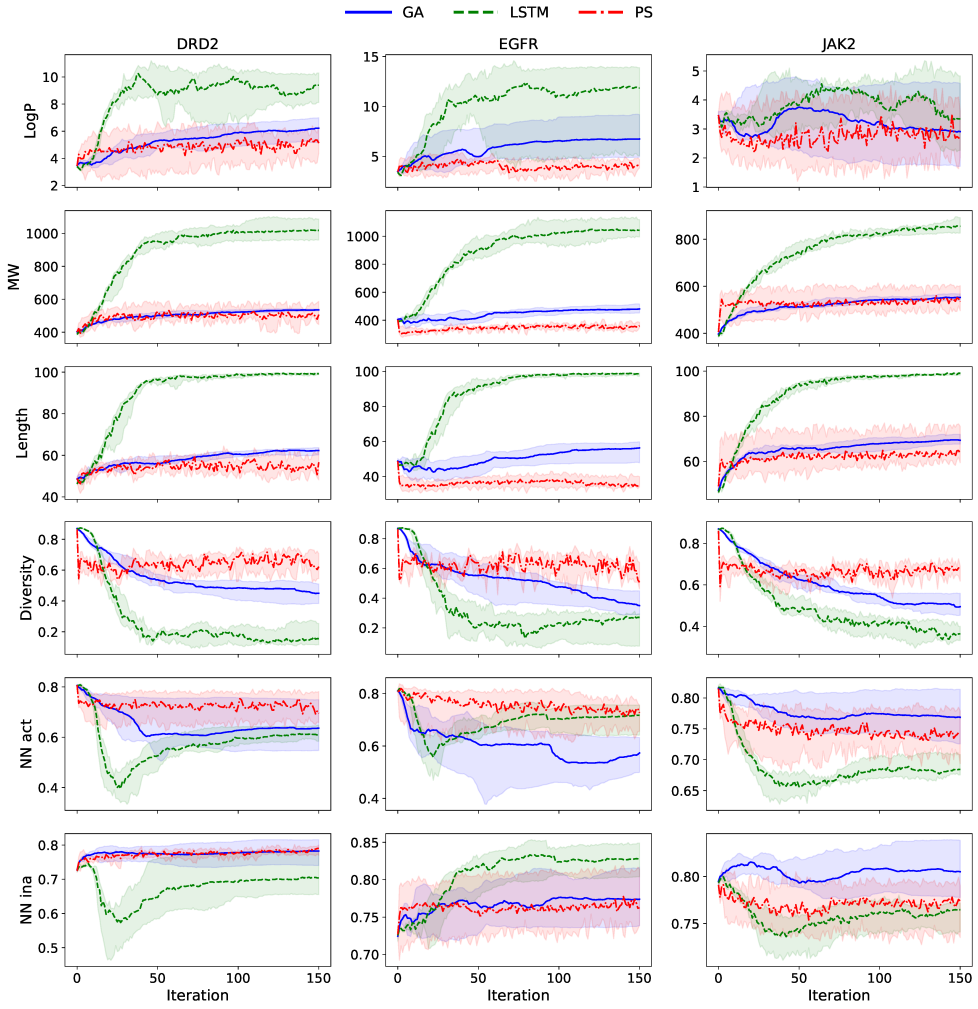


Figure S9: The course of additional metrics during optimization. The bold lines corresponds to medians and the shaded areas correspond to the interquartile range, taken over the means of single runs.

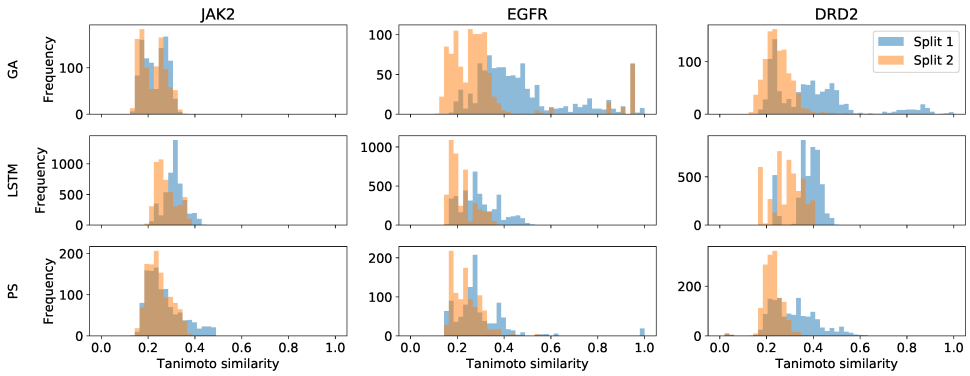


Figure S10: Distribution of nearest neighbour similarities of the optimized molecules to split 1/2. The results of all runs have been merged for each combination of data set and optimizer. In most cases the nearest neighbour similarities are greater for split 1 than for split 2. Tanimoto similarities are based on the same ECFP4 fingerprints that were used for training the classifiers.

References

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

[2] Landrum, G. RDKit: Open-Source Cheminformatics. 2006. URL: http://www.rdkit.org.

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

[4] Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., et al. "Scikit-Learn: Machine Learning in Python". In: J Mach Learn Res 12.85 (2011), pp. 2825–2830. URL: http://jmlr.org/papers/v12/pedregosa11a.html.

[5] Wildman, S. A. and Crippen, G. M. "Prediction of Physicochemical Parameters by Atomic Contributions". In: J Chem Inf Comput Sci 39.5 (1999), pp. 868–873. URL: https://doi.org/10.1021/ci9903071.

2.2 Diverse Hits in De Novo Molecule Design: Diversity-Based Comparison of Goal-Directed Generators

JCIM JOURNAL OF CHEMICAL INFORMATION AND MODELING

pubs.acs.org/jcim

Letter

Diverse Hits in De Novo Molecule Design: Diversity-Based Comparison of Goal-Directed Generators

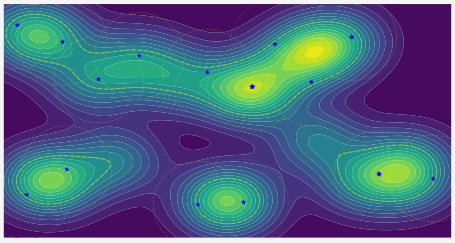
Philipp Renz, Sohvi Luukkonen, and Günter Klambauer*

Cite This: https://doi.org/10.1021/acs.jcim.4c00519

Read Online

ACCESS | Metrics & More | Article Recommendations | Supporting Information

ABSTRACT: Since the rise of generative AI models, many goal-directed molecule generators have been proposed as tools for discovering novel drug candidates. However, molecule generators often produce highly similar molecules and tend to overemphasize conformity to an imperfect scoring function rather than capturing the true underlying properties sought. We rectify these two shortcomings by offering diversity-based evaluations using the #Circles metric and considering constraints on scoring function calls or computation time. Our findings highlight the superior performance of SMILES-based autoregressive models in generating diverse sets of desired molecules compared to graph-based models or genetic algorithms.



INTRODUCTION

Goal-directed *de novo* drug design (DNDD) aims to generate small molecules possessing specific properties like efficacy, low toxicity, and drug-likeness,[1] by exploring the vast space of drug-like molecules.[2] This process involves generating novel chemical structures, guided by on-the-fly feedback from a scoring function to incorporate desired properties efficiently. With the surge of generative artificial intelligence, the field has witnessed a surge in interest, leading to the development of numerous new molecule generators, particularly those based on deep learning.[3−6]

Generating diverse sets of high-scoring molecules is essential in drug discovery. While most methods focus on producing individual high-scoring molecules, the reliance on quantitative structure−property relationship (QSPR) models introduces uncertainties and biases due to limited training data.[7] These errors propagate to molecule generators, emphasizing the need for diverse molecule sets to enhance the chances of identifying successful drug candidates.[8−12] Furthermore, diversity in molecule generation can lead to the exploration of novel chemical spaces beyond patented compounds.[13] However, many existing generators suffer from "mode collapse," producing only a limited range of similar molecules.[12,14−17] Various approaches have been proposed to address this issue and improve diversity in generated molecules.[16,18−23]

Previous comparative studies of molecule generators have often used insufficient diversity metrics. Well-known DNDD benchmarking platforms and leaderboards, such as GuacaMol[17] and MOSES,[14] include some classic diversity metrics: uniqueness and/or internal diversity, in the case of nongoal-directed molecule generation. But to our knowledge, there has been no systematic benchmark study of the capacity of different goal-directed molecule generators to generate a diverse set of high-scoring molecules.

Moreover, traditional metrics exhibit significant limitations in accurately characterizing the chemical space represented by a set of molecules.[24,25] For example, in Figure 1 we show how the arguably most commonly used diversity metric, *internal diversity*, fails to capture coverage of chemical space. More simple metrics, like the fraction of unique molecules and unique Bemis-Murcko scaffolds[26] are also inadequate as they can be optimized by generating many highly similar molecules, differing only in minor features. Recently diversity metrics based on sphere exclusion,[27] such as *SEDiv*[28] and *#Circles*,[25] have been proposed to quantify chemical space coverage. These metrics have been shown to align well with chemical intuition regarding the chemical diversity of known libraries and correlate well with the coverage of biological functionalities.

In addition to the aforementioned mode collapse problem, molecule generators tend to overoptimize molecules to their accessed scoring functions rather than to the actual desired properties.[7] A high number of scoring function calls can lead to this overfitting to the biased QSPR models and a decline in molecule quality over the optimization cycles. Therefore, it is essential to evaluate and compare generators within a

Received: March 28, 2024
Revised: July 10, 2024
Accepted: July 11, 2024

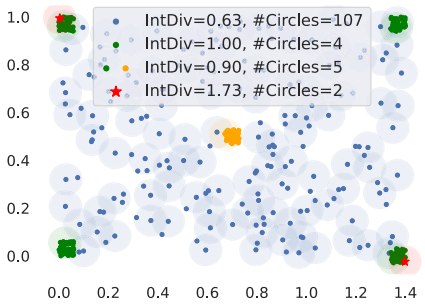


Figure 1. Comparison of internal diversity and #Circles. Internal diversity fails to capture high coverage of chemical space (blue), can be large for a few clusters of very similar molecules (green), and can decrease when adding additional molecules (green/yellow). IntDiv is maximized by two molecules with maximal distance (red). #Circles accurately captures the coverage of all sets.

standardized computational budget, by restricting a) the overall compute time or b) the number of scoring function evaluations. Additionally, there's growing interest in replacing machine learning-based scoring functions with more accurate but computationally expensive physics-based methods like docking.[28−31] Generative methods that can efficiently learn from a few scoring function evaluations are preferable for such costly scoring functions. Gao et al.[32] tested the sample efficiency of a range of generative methods given a constraint on the number of scoring function calls. Still, no studies focus on the generated molecules' diversity under computational constraints.

In this work, we address the two shortcomings of previous comparisons, a) the insufficient diversity metrics and b) the generation without limitations on the computational budget. We systematically benchmark the performance of established molecule generators at generating diverse high-scoring molecules, referred to as *diverse hits*. We evaluate these generators within the framework of goal-directed optimization, where they operate under constraints such as a limited number of scoring function calls or time, emphasizing computational cost. We utilize the *#Circles* diversity metric as a key performance indicator, providing a comprehensive assessment of generative model efficiency in practical scenarios.

BENCHMARK SETUP

Diverse Hits. We evaluate the performance of the tested generators based on the diversity of the generated high-scoring molecules. We define high-scoring molecules as ones with a score above a threshold S and refer to them as *hits*. We use the #Circles[25] metric to measure diversity of the found hits. This metric counts the number of generated hits that are pairwise distinct by a distance threshold D. We refer to this as the number of *diverse hits*.

More specifically, given the set of generated hits, $\mathcal{G}$, the number of diverse hits is given by

$$\mu(\mathcal{G}; D) = \max_{C \in \mathcal{P}(\mathcal{G})} |C| \text{ s.t. } \forall\, x \neq y \in C\colon d(x, y) \geq D$$

where $\mathcal{P}$ denotes the power set and $d(x, y)$ is the distance between molecules x and y. This metric ensures that each found hit that is sufficiently different from those already found adds to the performance, and that hits similar to each other are not double-counted.

Figure 1 illustrates the computation of this metric as finding the largest set of circles centered on the molecules, such that no center lies within another circle. We provide a more detailed description of this metric in Supporting Information Section S1.

Scoring Functions. *Bioactivity Prediction Models.* We evaluate the methods on three well-established molecule binary bioactivity label optimization tasks: JNK3,[33] GSK3β,[33] and DRD2.[16] For each target we train a Random Forest classifier[34] as a basis for our scoring functions. Table S1 provides details on the data sets and the performance of the predictive models. All scoring functions exhibit robust predictive performance, as indicated by their ROCAUC and Average Precision (AP) values.

During optimization, we use the classifier's probabilistic activity output, $p_{\mathrm{RF}}(s)$, as a scoring function. When predicting if a molecule is a hit, we adopt a score threshold of $S = 0.5$. Further details on the QSAR models are given in Supporting Information Section S2.1.

Property Filters. Generative models often generate molecules with high molecular weights (MW) or water-octanol partition coefficients (logP) and may contain idiosyncratic substructures, rendering them impractical for drug discovery projects,[7,35] and often these molecules would be discarded in real-world applications. We address this issue by incorporating lenient property constraints into the scoring functions[35] by defining acceptable ranges for MW ([157,761]Da) and logP ([-2.0,8.3]) values, and the fraction of idiosyncratic substructures ([0.00,0.08]). Further details on these filters are given in Supporting Information Section S2.2. During scoring, molecules violating any of these ranges have their score set to zero.

Diversity Filter. Most goal-directed molecule generators are not suitable for diverse generation out of the box, as they tend to get stuck in local optima of the scoring function.[12,14−17] To enable diverse molecule generation, we enhance the scoring functions with the *diversity filter* (DF) from Blaschke et al.[16] It assigns zero scores to molecules that are within a distance threshold $D_{\mathrm{DF}} = 0.7$ to previously found hits. This approach prevents the optimization process from getting trapped in local optima and promotes the exploration of new chemical space regions. The DF proved to be crucial for performance in preliminary experiments and its use allows for the meaningful inclusion of generative algorithms originally designed for single-molecule optimization. A detailed description of the DF is given in Supporting Information Section S2.3.

The final scoring function is given by the product of the bioactivity model prediction, and the binary property and diversity filters.

Compute Constraints. We evaluate the performance of the generators to create diverse hits under two computational constraint settings: (a) **Sample limit**, we limit the number of scoring function evaluations to 10K as proposed by Gao et al.,[32] and (b) **Time limit**, we limit the time available to the algorithms to 600 s. All algorithms are executed using 8 cores of an AMD Ryzen Threadripper 1920X and a single NVidia RTX 2080 GPU.

Generative Models. We utilize our benchmark setup to assess the following 12 methods. The methods were chosen based on their performance in previous benchmarks[17,32] and to

https://doi.org/10.1021/acs.jcim.4c00519
J. Chem. Inf. Model. XXXX, XXX, XXX−XXX

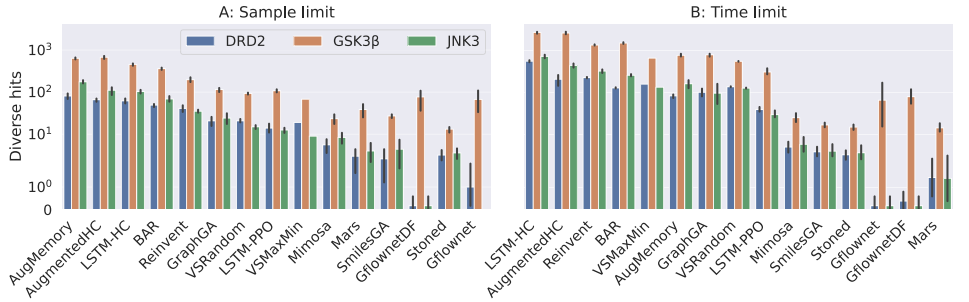


Figure 2. Number of diverse hits found by the tested methods (ordered by average rank) for the three studied optimization tasks. **A.** Results under a constraint of 10K scoring function evaluations. **B.** Results under a time constraint of 600 s. Error bars show the range of the results.

ensure that a range of methodically different approaches is included.

We test six LSTM-based autoregressive models operating on SMILES: **LSTM-HC**[36] optimizing with a hill-climb algorithm, **LSTM-PPO**[37] optimizing with the PPO algorithm, **Reinvent**[38] optimizing with the REINFORCE algorithm,[39] and three extensions of Reinvent: **AugmentedHC** (mixture of Reinvent and hill-climb),[40] **AugMemory**,[41] and BestAgentReminder (**BAR**).[42] We also test three genetic algorithms making use of mutations of different molecular representations: **GraphGA**[43] operating on molecular graphs, **SmilesGA**[44] operating on SMILES, and **Stoned**[45] operating on SELFIES.[46] We further test three models that generate molecules via sequential graph edits: **Mars**,[47] **Mimosa**,[48] and **GFlowNet/GFlowNetDF**,[21] which is tested with and without the DF as it supports diverse generation by default.

We compare these methods against two virtual screening (VS) baselines using the GuacaMol data set[17] as a screening library. VS methods are deemed inefficient as they ignore feedback from already scored molecules but serve as a valuable baseline. The **VS Random** baseline evaluates the molecules with the scoring function in random order from the library. In contrast, the **VS MaxMin** baseline first sorts the molecules in the library with the MaxMin algorithm.[49] This promotes diversity by ensuring that molecules screened first have as large pairwise distances as possible and prevents evaluating redundant molecules. Further details about these methods and the choice to exclude others are discussed in Supporting Information Section S3.

Optimization. We conducted a hyperparameter search to optimize each combination of generative algorithm, scoring function, and computational constraint. Employing a random search with 15 trials per combination, we explored various hyperparameter ranges, and to assess result stability, we executed five independent runs with distinct random seeds. The selected hyperparameters are detailed in Supporting Information Table S2.

Throughout the optimizations, we tracked all generated molecules, their corresponding scores, and the generation time. This comprehensive recording prevents valuable molecules from being discarded unnecessarily. This is particularly crucial when using a diversity filter that steers the search away from already discovered solutions.

RESULTS AND DISCUSSION

We benchmarked the capacity of a wide range of goal-directed molecule generators to design diverse hits under two computational constraint settings for three protein targets. The main results, i.e., the number of diverse hits under the constraints are shown in Figure 2 and discussed here. Extended results with complementary metrics are given in Supporting Information Section S5.

Large Differences in Performance between Models and Tasks. Above all, we observe in Figure 2 a significant difference in the capacity to produce diverse hits between the different algorithms, tasks, and computational constraints. The number of diverse hits ranges from several molecules for Mars (worst) to several thousand molecules for LSTM-HC (best) in the time-constrained setting. We also see that the performance is highly task-dependent: all approaches find ∼10 × more diverse hits for GSK3β than DRD2/JNK3 with the most extreme difference between the GFlowNet generators. Finally, the absolute, as well as relative performance of the algorithms depends strongly on the used computational constraint and on the available compute budget as shown in Supporting Information Sections S5.2 and S5.3.

SMILES-Based LSTM Models Perform Best in Generating Diverse Hits. Generally, the top ranks are dominated by autoregressive SMILES-based models. In the sample limit setting AugMemory performs best, making use of experience replay with selective purge and data augmentation, which leads to high sample efficiency. This allows the model to outperform its parent Reinvent. Similarly, the AugmentedHC model can also outperform its parent models Reinvent and LSTM-HC as shown in the original paper.[40] LSTM-HC attains the third rank and can outperform Reinvent, which is in contrast to results in single molecule optimization tasks.[32] The increase in performance of the extensions compared to their parent methods, comes with a significant computational cost as under the time limit the parent models are more competitive with their extensions, with LSTM-HC and Reinvent taking the top and third rank, respectively. We found that LSTM-HC is the most versatile algorithm achieving on average 84% of the top performance in each of the six settings (see Supporting Information Section S5.4). We think that this model class benefits from their stochastic generation policy which allows them to sample from different regions of chemical space at each optimization step. Thus, they can easily move to new

https://doi.org/10.1021/acs.jcim.4c00519
J. Chem. Inf. Model. XXXX, XXX, XXX−XXX

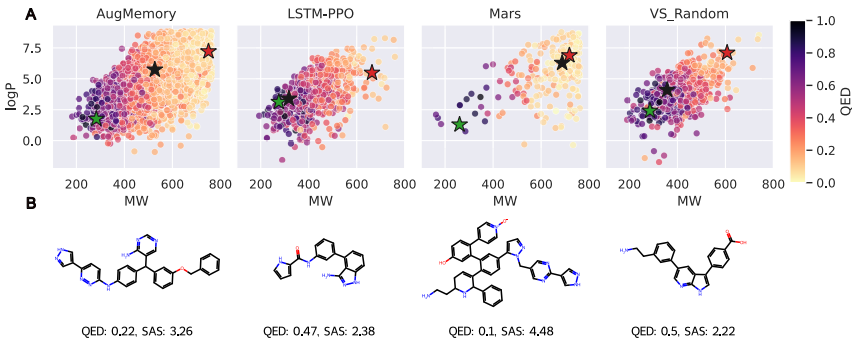


Figure 3. A. Chemical space cover by generated diverse hits by best (AugMemory), median (LSTM-PPO), and worst (Mars) methods, and the VS Random baseline, in the sample-constrained setting (all tasks combined). The green, black, and red stars represent the molecules with the highest, median, and lowest QED, respectively. **B.** Molecules with median QED.

regions once a new diverse hit has been found, resulting in high sample/compute efficiency.

Limited Number of Diverse Hits with Graph-Based and Genetic Algorithms. The graph-based models generally occupy the lower ranks in this comparison. Among them, GraphGA is the only model that outperforms the virtual screening baselines. SmilesGA and Stoned both perform poorly in this comparison. Along with GraphGA, they are not able to match their competitive performance in the single molecule optimization tasks.[32] We also found Mars and GFlowNet to perform poorly in this comparison, despite comparing well in previous diverse optimization studies.[21,25] This discrepancy highlights the importance of a meaningful benchmark setup and comparison to models suited for diverse optimization. We hypothesize that genetic algorithms encounter challenges in diverse generation tasks as they traverse chemical space using incremental modifications to existing molecules. This might cause a slow transition to new high-scoring regions once a hit is found.

Diversity at the Cost of Drug-likeness. We observed that some models achieve high diversity at the cost of drug-likeness. In Figure 3 we illustrate the chemical space covered and drug-likeness of the generated diverse hits by the best (AugMemory), median (LSTM-PPO), and worst (Mars) method, and the VS Random baseline, in the sample-constrained setting. We see that even though LSTM-PPO generates 10 times less diverse hits than AugMemory, these hits are generally more drug-like based on the QED score and overlap better with the VS baseline. In Supporting Information Section S5.5, we present distributions for additional properties of the generated diverse hits for all the methods. These distributions confirm that increased diversity is often achieved by generating larger, less drug-like molecules. We note that also the models achieving the lowest number of diverse hits (Mars and GFlowNet) struggle to generate drug-like molecules.

CONCLUSION

In our study, we rigorously tested a range of molecule generators in diverse *de novo* design tasks, introducing a benchmark setup that overcomes limitations identified in previous studies. Our findings underscore the crucial importance of considering computational resources in the generation of molecules and employing a meaningful diversity measure in the context of DNDD.

We found that SMILES-based autoregressive models perform well, compared to graph-based models and genetic algorithms in generating diverse sets of high-scoring molecules, and that single molecule optimization performance does not necessarily translate to diverse optimization settings. Performance values range over several orders of magnitude for different models and tasks and compute constraints. The latter highlights the importance of considering the specific application and available resources when choosing a generative model for practical applications.

Due to the broad range of possible applications of generative models in drug discovery, this study cannot cover all relevant aspects in detail, such as the used scoring functions, synthesizability issues, or the amount of available compute budgets, which may drastically differ in real-world applications. Nevertheless, we believe that our findings will generalize to other settings, and that our benchmark setup will be useful for future studies in the field.

ASSOCIATED CONTENT

Data Availability Statement

The data, code, and instructions necessary to reproduce the results of this study are available for download at https://github.com/ml-jku/diverse-hits.

Supporting Information

The Supporting Information is available free of charge at https://pubs.acs.org/doi/10.1021/acs.jcim.4c00519.

S1 - Diversity metric details, S2 - Scoring function details, S3 - Generative model details, S4 - Hyperparameter optimization, S5 - Extended results (PDF)

Additional drawings of generated structures (ZIP)

AUTHOR INFORMATION

Corresponding Author

Günter Klambauer − *Johannes Kepler University Linz, ELLIS Unit Linz, LIT AI Lab, Institute for Machine Learning, Linz, AT 4040, Austria;* orcid.org/0000-0003-2861-5552; Email: klambauer@ml.jku.at

https://doi.org/10.1021/acs.jcim.4c00519
J. Chem. Inf. Model. XXXX, XXX, XXX−XXX

Authors

Philipp Renz − *Johannes Kepler University Linz, Linz, AT 4040, Austria;* orcid.org/0000-0002-3323-7632

Sohvi Luukkonen − *Johannes Kepler University Linz, ELLIS Unit Linz, LIT AI Lab, Institute for Machine Learning, Linz, AT 4040, Austria;* orcid.org/0000-0001-9387-1427

Complete contact information is available at: https://pubs.acs.org/10.1021/acs.jcim.4c00519

Author Contributions

P.R. designed the study, incorporating input from G.K. and S.L. P.R. executed the experiments and analyzed the results. P.R., G.K., and S.L. contributed to writing the manuscript, with P.R. taking the lead. All authors reviewed and approved the manuscript.

Notes

The authors declare no competing financial interest.

■ ACKNOWLEDGMENTS

The ELLIS Unit Linz, the LIT AI Lab, and the Institute for Machine Learning, are supported by the Federal State Upper Austria. We thank the projects Medical Cognitive Computing Center (MC3), INCONTROL-RL (FFG-881064), PRIMAL (FFG-873979), S3AI (FFG-872172), DL for GranularFlow (FFG-871302), EPILEPSIA (FFG-892171), AIRI FG 9-N (FWF-36284, FWF-36235), AI4GreenHeatingGrids (FFG-899943), INTEGRATE (FFG-892418), ELISE (H2020-ICT-2019-3 ID: 951847), Stars4Waters (HORIZON-CL6-2021-CLIMATE-01-01). We thank NXAI GmbH, Audi. JKU Deep Learning Center, TGW LOGISTICS GROUP GMBH, Silicon Austria Laboratories (SAL), FILL Gesellschaft mbH, Anyline GmbH, Google, ZF Friedrichshafen AG, Robert Bosch GmbH, UCB Biopharma SRL, Merck Healthcare KGaA, Verbund AG, GLS (Univ. Waterloo), Software Competence Center Hagenberg GmbH, Borealis AG, TÜV Austria, Frauscher Sensonic, TRUMPF and the NVIDIA Corporation.

■ REFERENCES

(1) Schneider, G. *De Novo Molecular Design*; Wiley-VCH Verlag GmbH & Co., 2013.
(2) Walters, W. P. Virtual Chemical Libraries. *J. Med. Chem.* **2019**, *62*, 1116−1124.
(3) Sanchez-Lengeling, B.; Aspuru-Guzik, A. Inverse Molecular Design Using Machine Learning: Generative Models for Matter Engineering. *Science* **2018**, *361*, 360−365.
(4) Elton, D. C.; Boukouvalas, Z.; Fuge, M. D.; Chung, P. W. Deep Learning for Molecular Design—a Review of the State of the Art. *Mol. Syst. Des. Eng.* **2019**, *4*, 828−849.
(5) Luukkonen, S.; van den Maagdenberg, H. W.; Emmerich, M. T. M.; van Westen, G. J. P. Artificial intelligence in multi-objective drug design. *Curr. Opin. Struct. Biol.* **2023**, *79*, No. 102537.
(6) Fromer, J. C.; Coley, C. W. Computer-aided multi-objective optimization in small molecule discovery. *Patterns* **2023**, *4*, No. 100678.
(7) Renz, P.; Van Rompaey, D.; Wegner, J. K.; Hochreiter, S.; Klambauer, G. On Failure Modes in Molecule Generation and Optimization. *Drug Discovery Today: Technol.* **2019**, *32*−*33*, 55−63.
(8) Martin, Y. C. Diverse Viewpoints on Computational Aspects of Molecular Diversity. *J. Comb. Chem.* **2001**, 3, 231−250.
(9) Seneci, P. Trends in Drug Research III. In *Pharmacochemistry Library*; van der Goot, H., Ed.; 2002; Vol. *32*, pp 147−160.
(10) Angeli, P.; Gaviraghi, G. *Pharmacochemistry Library* **2002**, *32*, 95−96.
(11) Gorse, A.-D. Diversity in Medicinal Chemistry Space. *Curr. Top. Med. Chem.* **2006**, *6*, 3−18.
(12) Benhenda, M. ChemGAN Challenge for Drug Discovery: Can AI Reproduce Natural Chemical Diversity? *arXiv*, **2017**.
(13) Shimizu, Y.; Ohta, M.; Ishida, S.; Terayama, K.; Osawa, M.; Honma, T.; Ikeda, K. AI-driven molecular generation of not-patented pharmaceutical compounds using world open patent data. *J. Cheminformatics* **2023**, *15*, 120.
(14) 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.; Nikolenko, S.; Aspuru-Guzik, A.; Zhavoronkov, A. Molecular Sets (MOSES): A Benchmarking Platform for Molecular Generation Models. *Front. Pharmacol.* **2020**, *11*, 565644.
(15) 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*, 1736−1741.
(16) Blaschke, T.; Engkvist, O.; Bajorath, J.; Chen, H. Memory-Assisted Reinforcement Learning for Diverse Molecular de Novo Design. *J. Cheminformatics* **2020**, *12*, 68.
(17) Brown, N.; Fiscato, M.; Segler, M. H.; Vaucher, A. C. GuacaMol: Benchmarking Models for de Novo Molecular Design. *J. Chem. Inf. Model.* **2019**, *59*, 1096−1108.
(18) Rupakheti, C.; Virshup, A.; Yang, W.; Beratan, D. N. Strategy To Discover Diverse Optimal Molecules in the Small Molecule Universe. *J. Chem. Inf. Model.* **2015**, *55*, 529−537.
(19) Liu, X.; Ye, K.; van Vlijmen, H. W. T.; IJzerman, A. P.; van Westen, G. J. P. An Exploration Strategy Improves the Diversity of de Novo Ligands Using Deep Reinforcement Learning: A Case for the Adenosine A2A Receptor. *J. Cheminformatics* **2019**, *11*, 35.
(20) Chen, B.; Wang, T.; Li, C.; Dai, H.; Song, L. Molecule Optimization by Explainable Evolution. *ICLR*, 2020.
(21) Bengio, E.; Jain, M.; Korablyov, M.; Precup, D.; Bengio, Y. Flow Network Based Generative Models for Non-Iterative Diverse Candidate Generation. *NeurIPS*, **2021**.
(22) Pereira, T.; Abbasi, M.; Ribeiro, B.; Arrais, J. P. Diversity Oriented Deep Reinforcement Learning for Targeted Molecule Generation. *J. Cheminformatics* **2021**, *13*, 21.
(23) Bjerrum, E. J.; Margreitter, C.; Blaschke, T.; Kolarova, S.; de Castro, R. L.-R. Faster and More Diverse de Novo Molecular Optimization with Double-Loop Reinforcement Learning Using Augmented SMILES. *J. Comput. Aided. Mol. Des.* **2023**, *37*, 373−394.
(24) Waldman, M.; Li, H.; Hassan, M. Novel algorithms for the optimization of molecular diversity of combinatorial libraries. *J. Mol. Graph. Model.* **2000**, *18*, 412−426.
(25) Xie, Y.; Xu, Z.; Ma, J.; Mei, Q. How Much Space Has Been Explored? Measuring the Chemical Space Covered by Databases and Machine-Generated Molecules. *ICLR*, **2023**.
(26) Bemis, G. W.; Murcko, M. A. The Properties of Known Drugs. 1. Molecular Frameworks. *J. Med. Chem.* **1996**, *39*, 2887−2893.
(27) Gillet, V. J.; Willett, P. Dissimilarity-Based Compound Selection for Library Design. In *Combinatorial Library Design and Evaluation: Principles, Software, Tools, and Applications in Drug Discovery*; Ghose, A., Viswanadhan, V., Eds.; CRC Press: Boca Raton, FL, 2001; Chapter 13.
(28) Thomas, M.; Smith, R. T.; O'Boyle, N. M.; de Graaf, C.; Bender, A. Comparison of Structure- and Ligand-Based Scoring Functions for Deep Generative Models: A GPCR Case Study. *J. Cheminformatics* **2021**, *13*, 39.
(29) Guo, J.; Janet, J. P.; Bauer, M. R.; Nittinger, E.; Giblin, K. A.; Papadopoulos, K.; Voronov, A.; Patronov, A.; Engkvist, O.; Margreitter, C. DockStream: a docking wrapper to enhance de novo molecular design. *J. Cheminformatics* **2021**, *13*, 89.
(30) Goel, M.; Raghunathan, S.; Laghuvarapu, S.; Priyakumar, U. D. MoleGuLAR: Molecule Generation Using Reinforcement Learning with Alternating Rewards. *J. Chem. Inf. Model.* **2021**, *61*, 5815−5826.
(31) Elend, L.; Jacobsen, L.; Cofala, T.; Prellberg, J.; Teusch, T.; Kramer, O.; Solov'yov, I. A. Design of SARS-CoV-2 Main Protease Inhibitors Using Artificial Intelligence and Molecular Dynamic Simulations. *Molecules* **2022**, *27*, 4020.

(32) Gao, W.; Fu, T.; Sun, J.; Coley, C. W. Sample Efficiency Matters: A Benchmark for Practical Molecular Optimization. *NeurIPS* **2022**, 1.
(33) Li, Y.; Zhang, L.; Liu, Z. Multi-Objective de Novo Drug Design with Conditional Graph Generative Model. *J. Cheminformatics* **2018**, *10*, 33.
(34) Breiman, L. Random Forests. *Mach. Learn.* **2001**, *45*, 5−32.
(35) Thomas, M.; O'Boyle, N. M.; Bender, A.; De Graaf, C. Re-Evaluating Sample Efficiency in de Novo Molecule Generation. *arXiv* **2022**, 2212.01385.
(36) Segler, M. H. S.; Kogej, T.; Tyrchan, C.; Waller, M. P. Generating Focused Molecule Libraries for Drug Discovery with Recurrent Neural Networks. *ACS Cent. Sci.* **2018**, *4*, 120−131.
(37) Neil, D.; Segler, M.; Guasch, L.; Ahmed, M.; Plumbley, D.; Sellwood, M.; Brown, N. Exploring Deep Recurrent Models with Reinforcement Learning for Molecule Design. 2018; https://openreview.net/forum?id=HkcTe-bR-.
(38) Olivecrona, M.; Blaschke, T.; Engkvist, O.; Chen, H. Molecular De-Novo Design through Deep Reinforcement Learning. *J. Cheminformatics* **2017**, *9*, 48.
(39) Williams, R. J. Simple Statistical Gradient-Following Algorithms for Connectionist Reinforcement Learning. *Mach Learn* **1992**, *8*, 229−256.
(40) Thomas, M.; O'Boyle, N. M.; Bender, A.; de Graaf, C. Augmented Hill-Climb Increases Reinforcement Learning Efficiency for Language-Based de Novo Molecule Generation. *J. Cheminformatics* **2022**, *14*, 68.
(41) Guo, J.; Schwaller, P. Augmented Memory: Capitalizing on Experience Replay to Accelerate De Novo Molecular Design. *arXiv* **2023**, 2305.16160.
(42) Atance, S. R.; Diez, J. V.; Engkvist, O.; Olsson, S.; Mercado, R. De Novo Drug Design Using Reinforcement Learning with Graph-Based Deep Generative Models. *J. Chem. Inf. Model.* **2022**, *62*, 4863−4872.
(43) Jensen, J. H. A Graph-Based Genetic Algorithm and Generative Model/Monte Carlo Tree Search for the Exploration of Chemical Space. *Chem. Sci.* **2019**, *10*, 3567−3572.
(44) Yoshikawa, N.; Terayama, K.; Sumita, M.; Homma, T.; Oono, K.; Tsuda, K. Population-Based de Novo Molecule Generation, Using Grammatical Evolution. *Chem. Lett.* **2018**, *47*, 1431.
(45) Nigam, A.; Pollice, R.; Krenn, M.; Gomes, G. d. P.; Aspuru-Guzik, A. Beyond Generative Models: Superfast Traversal, Optimization, Novelty, Exploration and Discovery (STONED) Algorithm for Molecules Using SELFIES. *Chem. Sci.* **2021**, *12*, 7079−7090.
(46) Krenn, M.; Häse, F.; Nigam, A.; Friederich, P.; Aspuru-Guzik, A. Self-Referencing Embedded Strings (SELFIES): A 100% Robust Molecular String Representation. *Mach. Learn.: Sci. Technol.* **2020**, *1*, 045024.
(47) Xie, Y.; Shi, C.; Zhou, H.; Yang, Y.; Zhang, W.; Yu, Y.; Li, L. MARS: Markov Molecular Sampling for Multi-objective Drug Discovery. *ICLR*, **2021**.
(48) Fu, T.; Xiao, C.; Li, X.; Glass, L. M.; Sun, J. MIMOSA: Multi-constraint Molecule Sampling for Molecule Optimization. *AAAI*, **2021**.
(49) Sayle, R. *2D Similarity, Diversity and Clustering in RDKit*. 2019; https://www.nextmovesoftware.com/talks/Sayle_2DSimilarityDiversityAndClusteringInRdkit_RDKITUGM_201909.pdf.

https://doi.org/10.1021/acs.jcim.4c00519
J. Chem. Inf. Model. XXXX, XXX, XXX−XXX

Supporting Information
Diverse Hits in de novo Molecule Design: Diversity-based Comparison of Goal-directed Generators

Philipp Renz[1], Sohvi Luukkonen[2], and Günter Klambauer*[2]

[1]Johannes Kepler University Linz, Altenbergerstraße 69, Linz, AT 4040
[2]Johannes Kepler University Linz, ELLIS Unit Linz, LIT AI Lab, Institute for Machine Learning, Altenbergerstraße 69, Linz, AT 4040
*klambauer@ml.jku.at

S1 Diversity metric details

S1.1 Internal diversity

The internal diversity of a set of molecules is calculated as the average pairwise distance between all molecules in the set. For a set of molecules $\mathcal{M}$ it is calculated as,

$$\text{IntDiv}(\mathcal{M}) = \frac{1}{|\mathcal{M}|(|\mathcal{M}|-1)} \sum_{m_i \in \mathcal{M}} \sum_{m_j \in \mathcal{M}} \text{distance}(m_i, m_j). \tag{1}$$

As discussed in the main text, the internal diversity is a weak indicator of a set's diversity in the context of *de novo* molecule design. For instance, given a set of generated molecules, it can be maximized by selecting the two molecules with the highest pairwise distance. Additionally, internal diversity can be misleadingly high if the set consists of a few dense clusters of molecules that are far apart in molecular space. This scenario does not align with the goal of identifying many structurally diverse molecules.

When such a set is submitted to a biological assay, the information retrieved is highly redundant, making the assay costs unjustifiable relative to the information gained. Therefore we use the #Circles metric as the primary diversity metric in this study.

S1.2 #Circles metrics

The #Circles metric quantifies the diversity of a set of molecules. Given some distance metric in molecular space, such as the Tanimoto distance between fingerprint vectors, the #Circles metric is given by the maximal number of molecules with pairwise distances greater than a given threshold. This is equivalent to determining the packing number of the set of molecules in the distance metric space.

Alternatively, the #Circles metric can be interpreted as the size of a graph's Maximum Independent Set (MIS). In this context, given a set of molecules and a distance threshold, a graph is constructed where each molecule corresponds to a node, and edges connect nodes if the distance between the respective molecules is below the threshold. The MIS of this graph is the largest subset of nodes with no two nodes connected by an edge. Thus, the #Circles metric equals the size of the graph's MIS.

Calculating the packing number or MIS is generally NP-hard but can be approximated efficiently using a greedy algorithm. One simple, linear-time greedy algorithm is the sphere exclusion method. This algorithm iterates through the set of molecules, adding a molecule to the selected set if it is not within the distance threshold of any already selected molecule. This algorithm iterates through the set of molecules and adds a molecule to the set of selected molecules if it is not within the distance threshold of any of the already selected molecules.

The MaxMin algorithm offers a sub-quadratic time approximation for the packing number. Instead of iterating through the list in order, it selects the molecule that is furthest from the already selected molecules, specifically the one with the greatest minimum distance to the selected

set. This process can be further accelerated from quadratic to sub-quadratic time using an Alpha-beta pruning-like strategy.

S1.3 Diverse hits

To compute the number of diverse hits, we use the score threshold of $S = 0.5$, the Tanimoto distance between Morgan fingerprints (radius=2, size=2048) as the distance metric, and a distance threshold of $D = 0.7$, which aligns with the sharp drop in the probability of similar bioactivities beyond this value[1–3]. We efficiently approximate #Circles using the MaxMin algorithm[2].

S2 Scoring function details

S2.1 Bioactivity prediction models

For each of the datasets, we partition the data into training and test sets using a random 75/25% split and build a Random Forrest classifier[4] with the sklearn (v1.3.0) default parameters on Morgan fingerprints (radius=2, size=2048)[5].

In Table S1 we show the classification performance of the predictive models on the respective test sets. We evaluated the classification performance for the classifiers using the Area under the Receiver Operating Characteristic Curve (ROCAUC) and Average Precision (AP) metrics, which show good performance for all three datasets. We also report the Precision and Recall at the score threshold, $S = 0.5$, used in the molecule optimization tasks. Table S1 also shows the number of diverse hits in the training set giving a lower bound on the number of diverse hits recoverable by the generators.

We establish a score threshold of $S = 0.5$ for the optimization tasks. Precision values are high at this threshold, indicating that compounds scoring above this mark are very likely to be true actives. At the same time, the recall values are not excessively low, indicating that not too many true actives are rejected. Increasing the score threshold to a higher value like 0.9 would result in marginally higher precision values but drastically reduced recall, and would result in the discarding of many potentially active compounds. Furthermore using a high score threshold biases the optimization process towards recovering active compounds in the training set[6].

Table S1: Performance Metrics for JNK3, GSK3β and DRD2 activity prediction models. The table shows the ROCAUC, Average Precision (AP), Precision, and Recall at a 0.5 threshold, average scores of train and test actives, the number of samples, and the number of diverse hits in the training sets .

Target	ROCAUC	AP	Prec@0.5	Rec@0.5	Prec@0.9	Rec@0.9	#Samples	DivHits
JNK3	0.96	0.86	0.97	0.62	0.98	0.21	50390	220
GSK3	0.98	0.93	0.98	0.72	0.99	0.32	52802	643
DRD2	1.00	0.91	0.89	0.79	0.97	0.30	102981	229

S2.2 Property filters

Generative models have been observed to frequently produce compounds that feature atypical substructures or yield compounds with exceptionally high molecular weight (MW) or water-octanol partition coefficients (logP) values[6,7]. To enable a meaningful comparison we constrain the optimization objective to ensure that the generated molecules have properties within the range of those in a reference set. We use the ChEMBL subset provided in GuacaMol[8] as a reference set. In line with[7], we use relatively lenient property constraints to ensure that the optimization objective is not overly restrictive but ensures that compounds with strongly atypical properties are not rewarded. For the molecular weight (MW) and water-octanol partition coefficient (logP), we determine two-sided quantile-based lower and upper bounds, ensuring that 99% of the compounds in the ChEMBL dataset fall within these limits. For MW this results in a permissible range of $[157, 761]$ Da, and for logP this range is $[-2.0, 8.3]$.

We adopt the methodology outlined by[7] to address the challenge of uncommon substructures. We first partition the GuacaMol data set into a reference set and a calibration set, comprising 1.3M and 300K compounds, respectively. Then, we compute the unfolded ECFP4 fingerprints[5] for all compounds within the reference set, and determine the set of all occurring hash values. This

gives a reference of substructures typically found in drug-like molecules. We then compute the fingerprints for each compound in the calibration set and calculate the proportion of substructures absent in the reference set. We determine a threshold for this proportion such that 99% of the compounds in the calibration set fall below it, which evaluates to 0.08.

S2.3 Diversity filter

The diversity filter (DF) algorithm is given in Algorithm 1. The DF is initialized using an empty list M. Whenever a compound c is generated, it is compared with all compounds in M. If its distance to any compound in M is less than D_{DF} the compound does not pass the diversity filter and its DF-score is set to zero. If a compound passes the diversity filter, its DF-score is set to one. If a compound passes the diversity filter and $s(c) > s_{\mathrm{DF}}$ it is added to M.

For the experiments in this study, we set $D_{\mathrm{DF}} = 0.7$ and $s_{\mathrm{DF}} = 0.5$. We do not make explicit use of the bucket mechanism originally proposed[9]. Instead, we set the bucket size to one, as this resulted in quicker reorientation and faster exploration in preliminary experiments.

Algorithm 1: Computation of DF-score

Input : Generated compound c, score of compound $s(c)$, Filter list M, DF score threshold s_{DF}, DF distance threshold D_{DF}
Output: DF score s_{DF}, Updated filter list M

```
passes ← True ;                                  // Initialize pass status to True
for each compound m in M do
    if distance(c, m) < D_DF then
        passes ← False ;                         // Set pass status to False
        break
if passes and s(c) > S then
    Add c to M ;                                 // Add compound to filter list
return passes, M
```

S3 Generative model details

The tested models include a range of auto-regressive models operating on SMILES strings[10]. **LSTM-HC**[11] uses a hill-climb algorithm for fine-tuning a pre-trained LSTM model. **Reinvent**[12] optimizes a recurrent neural network using the REINFORCE algorithm[13] combined with prior regularization. **AugmentedHC**[14] forms a hybrid between the Reinvent and LSTM-HC models, by only using the Reinvent loss of the k top-scoring compounds to update the model. **AugMemory**[15] extends Reinvent, adding experience replay and data augmentation to increase sample efficiency. It makes use of selective memory purge to make experience replay compatible with the diversity filter described below. The BestAgentReminder (**BAR**) method[16] also extends the Reinvent algorithm, by keeping track of the best agent found so far and intersperses samples from the best agent with samples from the current model to stabilize training. **LSTM-PPO**[17] makes use of the popular PPO reinforcement learning algorithm to tune the model[18].

We test three genetic algorithms making use of mutations of different molecular representations. **GraphGA**[19] is a graph-based genetic algorithm that operates on the graph representation of molecules, and has shown competitive performance in previous benchmarks[8,20]. **SmilesGA**[21] generates novel molecules by encoding SMILES strings[10] into production rules of a context-free grammar and randomly inserting mutations into these rules. **Stoned**[22] generates molecules by introducing point mutations into the SELFIES[23] representation of molecules. All three genetic algorithms used randomly selected starting compounds from the Guacamol dataset[8].

We further test a range of models that generate molecules via sequential graph edits. **Mars**[24] makes use of Markov chain Monte Carlo sampling to generate molecules and achieved the best performance in a previous diverse optimization evaluation[25]. **Mimosa**[26] also operates on the graph representation of molecules and evolves molecules by applying a sequence of graph edits. **GFlowNet**[27] similarly sequentially builds molecules by graph edits but uses a specialized learning objective to enable diverse candidate generation.

While a range of strategies to promote diversity has been suggested[9,25,28–31], the diversity filter proposed in[9] is emerging as a standard approach employed in many studies[7,14,15,29] as it is a natural solution in line with our optimization objective, and is easy to combine with different generative models. Therefore we focus on the use of the DF for inducing diversity into the generated molecules. As GFlowNet is designed for diverse optimization we test it both with and without the diversity filter.

We do not test some other methods designed for diverse optimization, as they are conceptually similar to prior regularization used in Reinvent and its derivatives. We provide a detailed discussion in Section S3.2.

S3.1 Virtual screening baselines

We also test two virtual screening (VS) baselines that sample molecules the from the Guacamol dataset[8]. VS methods are deemed inefficient as they ignore feedback from already scored molecules, but serve as a valuable baseline. **VS Random** scores this library in random order.

We also test **VS MaxMin** which is adapted to diverse optimization. This algorithm operates on the screening library sorted by the MaxMin algorithm[2]. This algorithm first selects a random compound and then it selects the compound with the largest distance from the first one. The algorithm next selects the compound that is maximally distant from already selected ones. It does so by selecting the compound with the maximal minimum distance to already selected ones. This promotes diversity by ensuring that molecules screened first have as large pairwise distances as possible and prevents evaluating redundant molecules.

S3.2 Choice of tested algorithms

We did not test the following algorithms that have been reported to help improve the diversity of generated molecules. We did not include double-loop reinforcement learning[29] as it is conceptually very similar to the Augmented Memory algorithm[15]. Both algorithms use augmented versions of previously generated compounds to update the generative model multiple times.

We did not test the exploration approaches taken in[28,30] as they are conceptually similar to the prior regularization approaches that are used in Reinvent[12] and its descendants Augmented Hill-Climb[14] and Augmented Memory[15]. We also did not include the the method proposed in[32] in our experiments, as it is similar to the genetic algorithm equipped with the diversity filter.

Active learning methods have been shown to be effective in increasing the sample efficiency in optimization tasks[33,34]. Testing these methods is beyond the scope of this study, as implementation

and tuning is non-trivial. However, learning a fast proxy scoring function effectively reduces the cost of scoring molecules. Therefore the time constraint experiments give an indication of the performance of active learning methods. In principle, all the methods tested in this study can be combined with active learning methods, and pose an interesting avenue for future research.

S4 Hyperparameter optimization

For each generative model, we performed hyperparameter optimization to identify the best-performing hyperparameters for each combination of a generative algorithm, scoring function, and the used compute constraint. For each combination, we performed 15 runs with independently sampled hyperparameters. The hyperparameter distributions used for the random search and the selected parameters are given in Table S2.

In principle, the performance of the Reinvent derivatives AugmentedHC, AugmentedMem, and BAR could match that of Reinvent, when selecting the right hyperparameters. However, in this comparison, we restricted the parameter ranges to ensure non-trivial differences between the algorithms. This allows us to analyze the impact of the modifications in this setting.

Table S2: Hyperparameter ranges for the tested optimizers. We executed a random search using the distributions specified in this table. The selected hyperparameters for the respective constraint settings and targets are shown in the last six columns.

		Limit	Samples			Time		
Optimizer	Parameter	Search Space	DRD2	GSK3β	JNK3	DRD2	GSK3β	JNK3
AugHC	batch_size	RandInt(128, 512)	482	305	510	440	487	321
	learning_rate	LogUniform($10^{-4}, 10^{-3}$)	3.55e−4	2.89e−4	3.39e−4	2.24e−4	2.65e−4	1.90e−4
	sigma	Uniform(100.0, 500.0)	432.90	412.21	468.84	201.75	358.36	183.90
	topk	Uniform(0.15, 0.35)	0.16	0.17	0.17	0.20	0.24	0.17
AugMemory	augmentation_rounds	RandInt(1, 7)	6	6	2	4	1	1
	batch_size	RandInt(32, 128)	110	126	41	114	125	91
	learning_rate	LogUniform($10^{-4}, 10^{-3}$)	1.56e−4	1.25e−4	2.00e−4	1.61e−4	7.55e−4	3.90e−4
	replay_buffer_size	RandInt(32, 128)	107	82	111	111	74	61
	sigma	Uniform(100.0, 500.0)	409.80	369.46	332.04	493.91	490.70	261.11
BestAgentReminder	alpha	Uniform(0.3, 0.7)	0.67	0.42	0.34	0.45	0.42	0.53
	batch_size	RandInt(16, 256)	95	221	177	111	221	179
	learning_rate	LogUniform($10^{-4}, 10^{-3}$)	2.07e−4	2.58e−4	8.64e−4	3.12e−4	2.58e−4	1.25e−4
	sigma	Uniform(100.0, 500.0)	430.72	370.18	476.31	160.91	370.18	493.97
GA	mutation_rate	LogUniform($10^{-3}, 10^{-1}$)	3.96e−3	2.13e−3	1.86e−2	1.97e−2	4.91e−3	1.86e−2
	offspring_size	RandInt(50, 500)	160	426	230	245	130	230
	population_size	RandInt(50, 500)	217	397	409	444	496	409
Gflownet	learning_rate	LogUniform($10^{-5}, 10^{-3}$)	5.98e−5	1.72e−4	3.52e−4	1.17e−5	1.72e−4	3.33e−5
	momentum	Uniform(0.5, 0.9)	0.72	0.72	0.68	0.82	0.72	0.70
	sampling_tau	Uniform(0.8, 0.99)	0.87	0.96	0.83	0.97	0.96	0.93
GflownetDF	learning_rate	LogUniform($10^{-5}, 10^{-3}$)	8.25e−4	9.30e−4	1.37e−4	8.25e−4	3.75e−5	2.71e−5
	momentum	Uniform(0.5, 0.9)	0.83	0.78	0.60	0.83	0.77	0.57
	sampling_tau	Uniform(0.8, 0.99)	0.98	0.83	0.91	0.98	0.81	0.98
LSTM-HC	mols_to_sample	RandInt(8, 2048)	174	245	463	1695	859	1503
	optimize_n_epochs	RandInt(1, 6)	4	1	1	1	1	1
LSTM-PPO	batch_size	RandInt(64, 1024)	508	179	401	948	508	401
	clip_param	Uniform(0.1, 0.6)	0.11	0.19	0.41	0.12	0.22	0.41
	entropy_weight	Uniform(0.01, 1.0)	0.49	0.48	0.25	0.16	0.14	0.25
	episode_size	RandInt(64, 4096)	2277	1913	1121	2661	1932	1121
	kl_div_weight	RandInt(1, 10)	5	3	6	2	3	6
Mars	batch_size	RandInt(64, 512)	434	418	129	434	86	129
	n_layers	RandInt(1, 4)	1	3	2	1	1	2
	num_mols	RandInt(32, 512)	248	83	371	248	239	371
Mimosa	lamb	Uniform(0.1, 10.0)	7.07	5.84	2.03	7.07	5.84	1.73
	population_size	RandInt(50, 200)	150	199	87	150	199	85
	train_epoch	RandInt(1, 10)	2	2	9	2	2	8
Reinvent	batch_size	RandInt(256, 512)	260	288	417	462	454	464
	experience_replay	RandInt(0, 64)	49	34	54	38	12	50
	learning_rate	LogUniform($10^{-5}, 10^{-2}$)	1.00e−3	1.56e−3	3.25e−4	2.65e−4	2.24e−3	2.15e−4
	sigma	Uniform(100.0, 600.0)	582.10	540.86	296.22	357.38	244.20	283.52
SmilesGA	gene_size	RandInt(100, 600)	370	590	361	370	374	361
	n_mutations	RandInt(100, 300)	214	165	176	214	253	176
	population_size	RandInt(50, 200)	101	169	111	101	89	111
Stoned	generation_size	RandInt(50, 1000)	461	872	980	461	872	980

Table S3: Performance for the tested methods under a sample limit. Performance is given by the number of hits, diverse hits (DivHits), novel diverse hits (NDivHits), and internal diversity (IntDiv). The internal diversity is calculated on the discovered hits.

	DRD2			GSK3β			JNK3		
	DivHits	NDivHits	IntDiv	DivHits	NDivHits	IntDiv	DivHits	NDivHits	IntDiv
AugMemory	$81^{\pm 19\%}$	$9^{\pm 75\%}$	$0.76^{\pm 0.01}$	$636^{\pm 6\%}$	$507^{\pm 12\%}$	$0.82^{\pm 0.00}$	$176^{\pm 11\%}$	$104^{\pm 13\%}$	$0.77^{\pm 0.00}$
AugmentedHC	$66^{\pm 11\%}$	$3^{\pm 44\%}$	$0.77^{\pm 0.01}$	$674^{\pm 11\%}$	$533^{\pm 11\%}$	$0.84^{\pm 0.00}$	$111^{\pm 27\%}$	$63^{\pm 41\%}$	$0.79^{\pm 0.01}$
LSTM-HC	$62^{\pm 16\%}$	$8^{\pm 37\%}$	$0.76^{\pm 0.01}$	$456^{\pm 9\%}$	$231^{\pm 16\%}$	$0.84^{\pm 0.01}$	$103^{\pm 13\%}$	$36^{\pm 17\%}$	$0.78^{\pm 0.00}$
BAR	$49^{\pm 11\%}$	$1^{\pm 56\%}$	$0.77^{\pm 0.02}$	$361^{\pm 8\%}$	$156^{\pm 11\%}$	$0.85^{\pm 0.00}$	$69^{\pm 20\%}$	$20^{\pm 19\%}$	$0.79^{\pm 0.00}$
Reinvent	$41^{\pm 25\%}$	$3^{\pm 53\%}$	$0.74^{\pm 0.02}$	$198^{\pm 18\%}$	$135^{\pm 24\%}$	$0.81^{\pm 0.01}$	$35^{\pm 11\%}$	$6^{\pm 68\%}$	$0.75^{\pm 0.01}$
GraphGA	$21^{\pm 31\%}$	$7^{\pm 55\%}$	$0.75^{\pm 0.04}$	$115^{\pm 14\%}$	$78^{\pm 15\%}$	$0.84^{\pm 0.00}$	$24^{\pm 37\%}$	$10^{\pm 61\%}$	$0.79^{\pm 0.01}$
VSRandom	$21^{\pm 12\%}$	$0^{\pm 0\%}$	$0.82^{\pm 0.01}$	$93^{\pm 6\%}$	$7^{\pm 12\%}$	$0.87^{\pm 0.00}$	$15^{\pm 13\%}$	$0^{\pm 0\%}$	$0.83^{\pm 0.01}$
LSTM-PPO	$14^{\pm 32\%}$	$0^{\pm 0\%}$	$0.81^{\pm 0.02}$	$108^{\pm 9\%}$	$16^{\pm 26\%}$	$0.87^{\pm 0.00}$	$13^{\pm 18\%}$	$1^{\pm 71\%}$	$0.81^{\pm 0.02}$
VSMaxMin	$19^{\pm 0\%}$	$0^{\pm 0\%}$	$0.88^{\pm 0.00}$	$68^{\pm 0\%}$	$8^{\pm 0\%}$	$0.89^{\pm 0.00}$	$9^{\pm 0\%}$	$0^{\pm 0\%}$	$0.88^{\pm 0.00}$
Mimosa	$6^{\pm 47\%}$	$0^{\pm 224\%}$	$0.80^{\pm 0.06}$	$23^{\pm 33\%}$	$8^{\pm 62\%}$	$0.84^{\pm 0.02}$	$8^{\pm 37\%}$	$3^{\pm 49\%}$	$0.78^{\pm 0.07}$
Mars	$3^{\pm 62\%}$	$0^{\pm 224\%}$	$0.39^{\pm 0.26}$	$39^{\pm 45\%}$	$36^{\pm 49\%}$	$0.81^{\pm 0.02}$	$4^{\pm 64\%}$	$2^{\pm 84\%}$	$0.61^{\pm 0.09}$
SmilesGA	$3^{\pm 80\%}$	$0^{\pm 224\%}$	$0.64^{\pm 0.36}$	$27^{\pm 12\%}$	$14^{\pm 26\%}$	$0.85^{\pm 0.01}$	$4^{\pm 83\%}$	$2^{\pm 87\%}$	$0.58^{\pm 0.34}$
GflownetDF	$0^{\pm 224\%}$	$0^{\pm 224\%}$	$0.00^{\pm 0.00}$	$77^{\pm 60\%}$	$73^{\pm 61\%}$	$0.81^{\pm 0.00}$	$0^{\pm 224\%}$	$0^{\pm 224\%}$	$0.00^{\pm 0.00}$
Stoned	$3^{\pm 41\%}$	$0^{\pm 0\%}$	$0.62^{\pm 0.15}$	$13^{\pm 19\%}$	$1^{\pm 64\%}$	$0.79^{\pm 0.06}$	$4^{\pm 37\%}$	$0^{\pm 224\%}$	$0.56^{\pm 0.15}$
Gflownet	$1^{\pm 122\%}$	$0^{\pm 0\%}$	$0.15^{\pm 0.33}$	$67^{\pm 75\%}$	$64^{\pm 77\%}$	$0.81^{\pm 0.01}$	$0^{\pm 0\%}$	$0^{\pm 0\%}$	$0.00^{\pm 0.00}$

Table S4: Performance for the tested methods under a time limit. Performance is given by the number of hits, diverse hits (DivHits), novel diverse hits (NDivHits), and internal diversity (IntDiv). The internal diversity is calculated on the discovered hits.

	DRD2			GSK3β			JNK3		
	DivHits	NDivHits	IntDiv	DivHits	NDivHits	IntDiv	DivHits	NDivHits	IntDiv
LSTM-HC	$544^{\pm 7\%}$	$154^{\pm 15\%}$	$0.80^{\pm 0.00}$	$2620^{\pm 7\%}$	$2045^{\pm 8\%}$	$0.84^{\pm 0.00}$	$708^{\pm 13\%}$	$487^{\pm 13\%}$	$0.81^{\pm 0.00}$
AugmentedHC	$214^{\pm 31\%}$	$50^{\pm 42\%}$	$0.80^{\pm 0.01}$	$2543^{\pm 9\%}$	$2251^{\pm 10\%}$	$0.85^{\pm 0.00}$	$433^{\pm 14\%}$	$291^{\pm 15\%}$	$0.81^{\pm 0.00}$
Reinvent	$221^{\pm 5\%}$	$48^{\pm 16\%}$	$0.79^{\pm 0.01}$	$1315^{\pm 4\%}$	$1100^{\pm 5\%}$	$0.84^{\pm 0.00}$	$318^{\pm 11\%}$	$184^{\pm 17\%}$	$0.79^{\pm 0.01}$
BAR	$126^{\pm 4\%}$	$7^{\pm 30\%}$	$0.80^{\pm 0.01}$	$1469^{\pm 6\%}$	$1022^{\pm 10\%}$	$0.85^{\pm 0.00}$	$252^{\pm 7\%}$	$132^{\pm 15\%}$	$0.80^{\pm 0.01}$
VSMaxMin	$155^{\pm 0\%}$	$0^{\pm 0\%}$	$0.83^{\pm 0.00}$	$643^{\pm 0\%}$	$88^{\pm 0\%}$	$0.87^{\pm 0.00}$	$131^{\pm 0\%}$	$3^{\pm 0\%}$	$0.83^{\pm 0.00}$
AugMemory	$82^{\pm 11\%}$	$7^{\pm 44\%}$	$0.75^{\pm 0.01}$	$753^{\pm 9\%}$	$615^{\pm 12\%}$	$0.82^{\pm 0.01}$	$163^{\pm 26\%}$	$77^{\pm 33\%}$	$0.78^{\pm 0.01}$
GraphGA	$102^{\pm 27\%}$	$50^{\pm 21\%}$	$0.78^{\pm 0.01}$	$774^{\pm 10\%}$	$699^{\pm 11\%}$	$0.85^{\pm 0.00}$	$111^{\pm 56\%}$	$70^{\pm 68\%}$	$0.80^{\pm 0.01}$
VSRandom	$134^{\pm 3\%}$	$0^{\pm 0\%}$	$0.82^{\pm 0.00}$	$540^{\pm 2\%}$	$86^{\pm 4\%}$	$0.87^{\pm 0.00}$	$125^{\pm 4\%}$	$4^{\pm 12\%}$	$0.83^{\pm 0.00}$
LSTM-PPO	$39^{\pm 18\%}$	$0^{\pm 0\%}$	$0.81^{\pm 0.01}$	$308^{\pm 25\%}$	$69^{\pm 48\%}$	$0.87^{\pm 0.00}$	$30^{\pm 28\%}$	$2^{\pm 130\%}$	$0.82^{\pm 0.00}$
Mimosa	$5^{\pm 34\%}$	$0^{\pm 0\%}$	$0.82^{\pm 0.04}$	$26^{\pm 33\%}$	$8^{\pm 68\%}$	$0.84^{\pm 0.02}$	$6^{\pm 50\%}$	$1^{\pm 71\%}$	$0.74^{\pm 0.09}$
SmilesGA	$4^{\pm 35\%}$	$0^{\pm 224\%}$	$0.76^{\pm 0.11}$	$17^{\pm 17\%}$	$8^{\pm 33\%}$	$0.84^{\pm 0.04}$	$4^{\pm 59\%}$	$2^{\pm 82\%}$	$0.42^{\pm 0.40}$
Stoned	$3^{\pm 34\%}$	$0^{\pm 0\%}$	$0.54^{\pm 0.30}$	$15^{\pm 18\%}$	$1^{\pm 96\%}$	$0.76^{\pm 0.06}$	$4^{\pm 50\%}$	$0^{\pm 224\%}$	$0.61^{\pm 0.17}$
Gflownet	$0^{\pm 224\%}$	$0^{\pm 224\%}$	$0.00^{\pm 0.00}$	$112^{\pm 70\%}$	$108^{\pm 73\%}$	$0.80^{\pm 0.01}$	$0^{\pm 224\%}$	$0^{\pm 224\%}$	$0.00^{\pm 0.00}$
GflownetDF	$0^{\pm 137\%}$	$0^{\pm 224\%}$	$0.00^{\pm 0.00}$	$87^{\pm 53\%}$	$84^{\pm 51\%}$	$0.81^{\pm 0.01}$	$0^{\pm 224\%}$	$0^{\pm 224\%}$	$0.00^{\pm 0.00}$
Mars	$2^{\pm 100\%}$	$0^{\pm 0\%}$	$0.37^{\pm 0.34}$	$15^{\pm 36\%}$	$10^{\pm 45\%}$	$0.75^{\pm 0.03}$	$2^{\pm 122\%}$	$0^{\pm 224\%}$	$0.19^{\pm 0.19}$

S5 Extended results

S5.1 Additional metrics

Tables S3 and S4 give extended results for both constraints settings, including the number of hits, novel diverse hits, and internal diversity of the found hits. The novel diverse hits are calculated as follows: First, we take the hits found by the generative model and from these remove all compounds that have a distance of less than 0.7 to any active compound in the training set. Then we calculate the #Circles metric on this reduced set. While the number of novel diverse hits is in general highly correlated with the number of diverse hits, the virtual screening methods generate relatively few novel diverse hits. This is because the virtual screening methods are biased toward finding compounds that are similar to the training set, which can be seen in similar distributions of the molecular properties of the generated molecules and the training set (see Figures S6 & S7).

Figure S1 shows the correlation between different diversity metrics, namely the number of hits, diverse hits, the number of novel diverse hits, and the internal diversity of the hits. We can see that the number of diverse hits is strongly correlated with the number of novel diverse hits. The internal diversity is only weakly correlated with these metrics. This means internal diversity values reported in previous studies are not a good indicator of the number of diverse (novel) hits found.

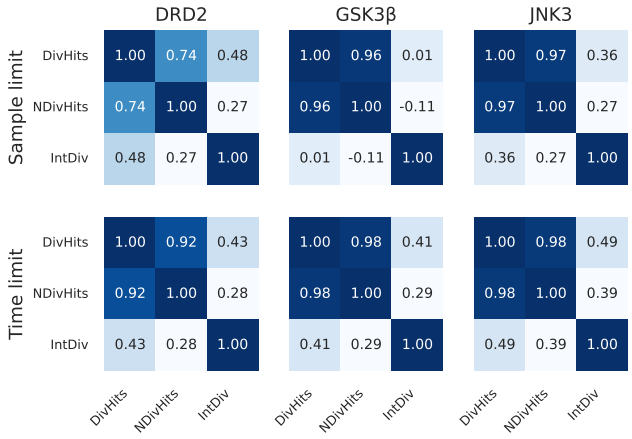


Figure S1: Correlation between different diversity metrics. The number of diverse hits is correlated with the number of novel diverse hits. The internal diversity is only weakly correlated with the other metrics.

S5.2 Optimization curves

Figures S2 and S3 show the number of diverse hits found by the tested methods over the number of scoring function evaluations and the time elapsed. Most methods show no sign of saturating performance within the chosen limits. This shows that the comparison of the methods is only meaningful under standardized computing constraints. The method ranking remains somewhat constant throughout the optimization. Only single curves, like LSTM-HC on DRD2 in the sample-constrained setting, show a significant increase in performance towards the end of the optimization. Similarly, this holds for AugmentedHC in the time-constrained setting.

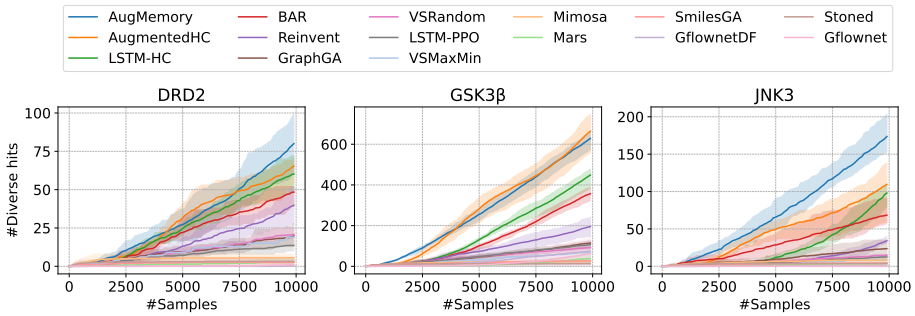


Figure S2: Number of diverse hits found by the tested methods over the number of scoring function evaluations.

S5.3 Cost to fixed performance

In Figure S4 we show the incurred compute costs to reach a fixed number of diverse hits. We chose this number such that most methods were able to reach them within the given compute constraints. This resulted in 20/100 diverse hits for the sample/time limit respectively.

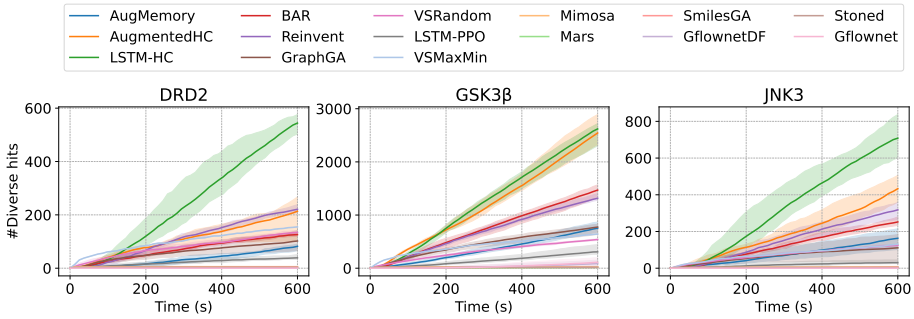


Figure S3: Number of diverse hits found by the tested methods over the elapsed time.

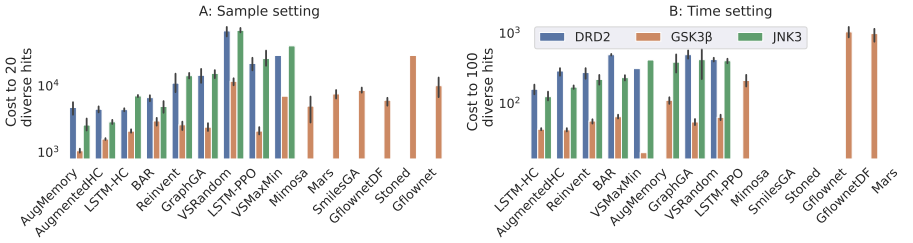


Figure S4: Compute cost to reach a fixed number of diverse hits. The cost is given in terms of the number of scoring function evaluations / time. Missing bars indicate that the method did not reach the threshold within the given compute constraints.

The difference in costs to reach the same number of diverse hits is very substantial with virtual screening methods requiring up to 10x the number of scoring function evaluations compared to the best performing methods in the sample-constrained setting. The differences are less pronounced in the time-constrained setting, where the factors are in the range from 3-5x.

The missing bars indicate that methods did not reach the threshold within the given compute constraints. These ommisions were necessary to keep the thresholds high enough for the results to be informative, for the rest of the methods. Running these remaining methods until they reach the threshold would have resulted in an excessive run time, as the limits are quite ambitious for the lower performing methods. The maximum values in this comparison serve as a lower bound for the missing values.

S5.4 Model ranking

To determine an overall ranking, we computed a combined ranking incorporating model performance in all six settings (2 compute limits × 3 tasks). In Figure S5 we show the average rank and the mean normalized performance for each algorithm. The mean normalized performance is computed as an average of an algorithm's performance divided by the best performance for the respective task and compute limit.

LSTM-HC demonstrates the highest mean normalized performance obtaining 84%, followed by AugmentedHC at 74% and AugMemory at 60%. This makes LSTM-HC the most versatile of the optimizers. However, it is crucial to ensure that the hyperparameters for LSTM-HC are appropriately adapted to the compute budget. Hyperparameters optimized for the sample-constrained setting will perform poorly in the time-constrained setting, and vice versa.

S5.5 Molecular property distributions

In this section, we show distributions of the molecular weight (MW), the water-octanol partition coefficient (logP), the fraction of *de novo* ECFP4 bits, the synthetic accessibility (SA), the quantitative estimate of drug-likeness (QED), and the length of the SMILES strings of the generated molecules. Figure S6 shows the distributions for the sample constrained setting, and Figure S7

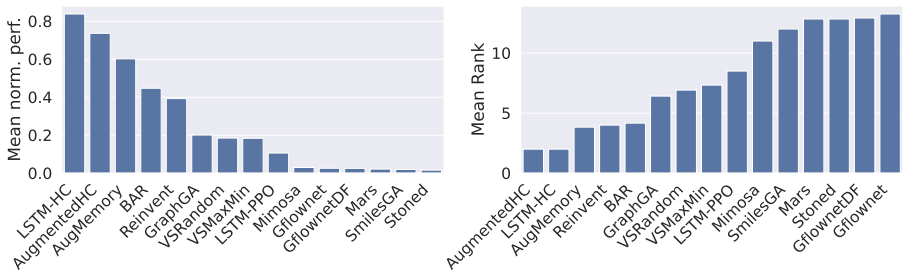


Figure S5: Model ranking in terms of mean normalized performance and average rank across all six tasks. LSTM-HC and AugmentedHC perform best and are most suitable as one-size-fits-all algorithms

shows the distributions for the time-constrained setting. For the properties used in the property filter, we also show the used thresholds.

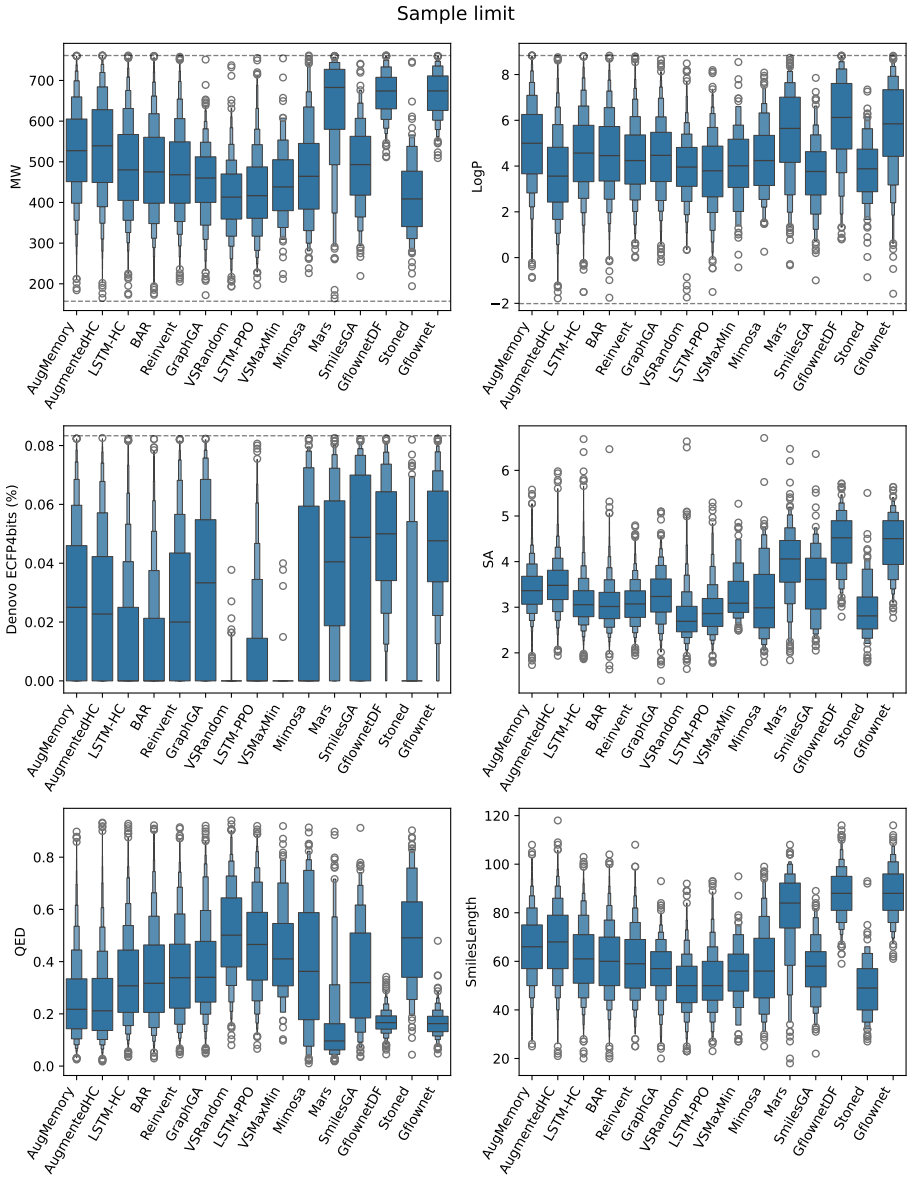


Figure S6: Distributions of the molecular properties of the generated molecules for the sample-constrained setting. The dashed lines indicate the thresholds used in the property filter. Results are aggregated over the three optimization tasks.

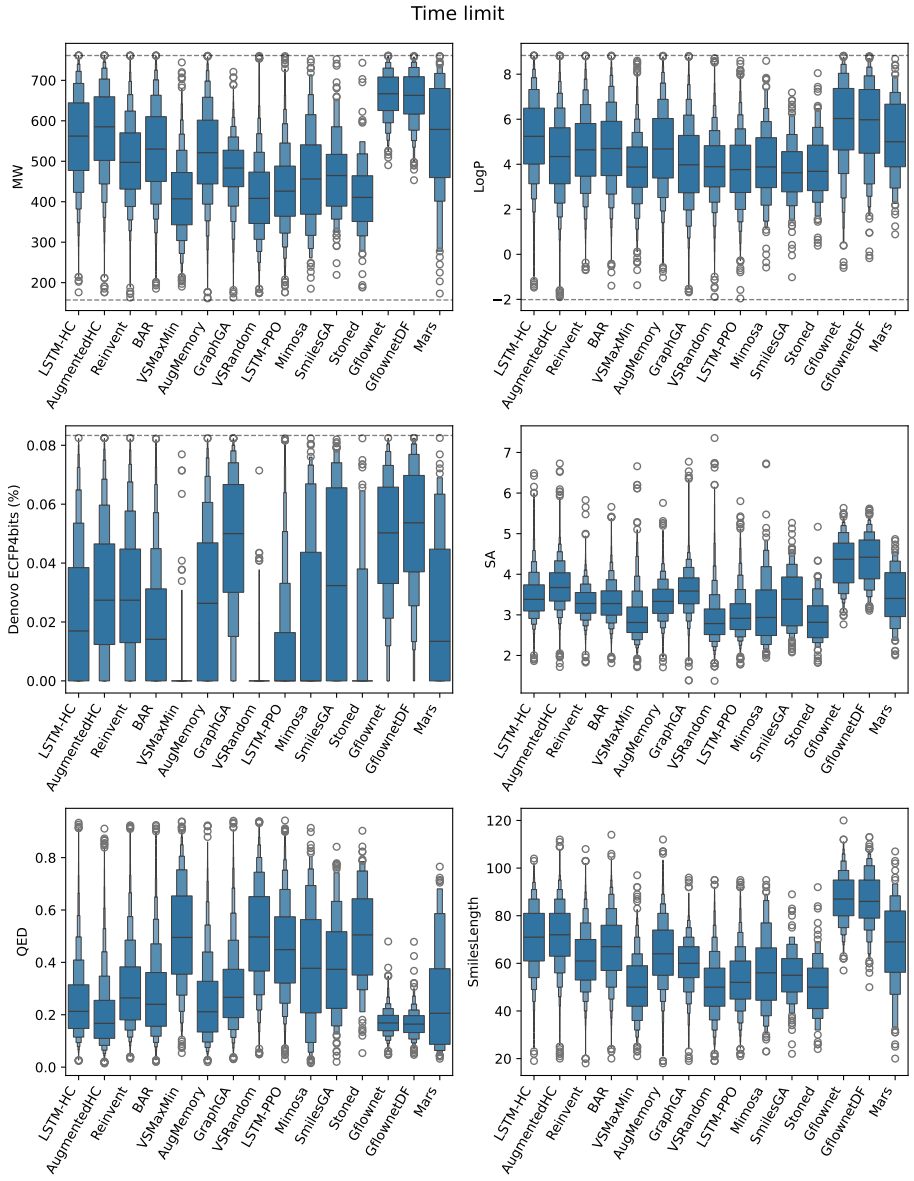


Figure S7: Distributions of the molecular properties of the generated molecules for the time-constrained setting. The dashed lines indicate the thresholds used in the property filter. Results are aggregated over the three optimization tasks.

References

[1] Jasial, S., Hu, Y., Vogt, M., Bajorath, J. "Activity-Relevant Similarity Values for Fingerprints and Implications for Similarity Searching". In: *F1000Res* 5 (2016), Chem Inf Sci–591. DOI: 10.12688/f1000research.8357.2.

[2] Sayle, R. *2D Similarity, Diversity and Clustering in RDKit*. 2019. URL: https://www.nextmovesoftware.com/talks/Sayle%5C%5C_2DSimilarityDiversityAndClusteringInRdkit%5C%5C_RDKITUGM%5C%5C_201909.pdf (visited on 11/17/2023).

[3] Landrum, G. *A New "Lessel and Briem like" Dataset*. URL: http://rdkit.blogspot.com/2019/10/a-new-lessel-and-briem-like-dataset.html (visited on 2023-10-26).

[4] Breiman, L. "Random Forests". In: *Mach. Learn.* 45.1 (2001), pp. 5–32. DOI: 10.1023/A:1010933404324.

[5] Rogers, D., Hahn, M. "Extended-Connectivity Fingerprints". In: *J. Chem. Inf. Model.* 50.5 (2010), pp. 742–754. DOI: 10.1021/ci100050t.

[6] Renz, P., Van Rompaey, D., Wegner, J. K., Hochreiter, S., Klambauer, G. "On Failure Modes in Molecule Generation and Optimization". In: *Drug Discov. Today: Technol.* Artificial Intelligence 32–33 (2019), pp. 55–63. DOI: 10.1016/j.ddtec.2020.09.003.

[7] Thomas, M., O'Boyle, N. M., Bender, A., De Graaf, C. "Re-Evaluating Sample Efficiency in de Novo Molecule Generation". In: (2022). DOI: 10.48550/arXiv.2212.01385.

[8] Brown, N., Fiscato, M., Segler, M. H., Vaucher, A. C. "GuacaMol: Benchmarking Models for de Novo Molecular Design". In: *J. Chem. Inf. Model.* 59.3 (2019), pp. 1096–1108. DOI: 10.1021/acs.jcim.8b00839.

[9] Blaschke, T., Engkvist, O., Bajorath, J., Chen, H. "Memory-Assisted Reinforcement Learning for Diverse Molecular de Novo Design". In: *J. Cheminformatics* 12.1 (2020), p. 68. DOI: 10.1186/s13321-020-00473-0.

[10] Weininger, D. "SMILES, a Chemical Language and Information System". In: *J. Chem. Inf. Comput. Sci.* 28.1 (1988), pp. 31–36. DOI: 10.1021/ci00057a005.

[11] Segler, M. H. S., Kogej, T., Tyrchan, C., Waller, M. P. "Generating Focused Molecule Libraries for Drug Discovery with Recurrent Neural Networks". In: *ACS Cent. Sci.* 4.1 (2018), pp. 120–131. DOI: 10.1021/acscentsci.7b00512.

[12] Olivecrona, M., Blaschke, T., Engkvist, O., Chen, H. "Molecular De-Novo Design through Deep Reinforcement Learning". In: *J. Cheminformatics* 9.1 (2017), p. 48. DOI: 10.1186/s13321-017-0235-x.

[13] Williams, R. J. "Simple Statistical Gradient-Following Algorithms for Connectionist Reinforcement Learning". In: *Mach Learn* 8.3-4 (1992), pp. 229–256. DOI: 10.1007/BF00992696.

[14] Thomas, M., O'Boyle, N. M., Bender, A., de Graaf, C. "Augmented Hill-Climb Increases Reinforcement Learning Efficiency for Language-Based de Novo Molecule Generation". In: *J. Cheminformatics* 14.1 (2022), p. 68. DOI: 10.1186/s13321-022-00646-z. (Visited on 09/04/2023).

[15] Guo, J., Schwaller, P. "Augmented Memory: Capitalizing on Experience Replay to Accelerate De Novo Molecular Design". In: (2023). DOI: 10.48550/arXiv.2305.16160. arXiv: 2305.16160.

[16] Atance, S. R., Diez, J. V., Engkvist, O., Olsson, S., Mercado, R. "De Novo Drug Design Using Reinforcement Learning with Graph-Based Deep Generative Models". In: *J. Chem. Inf. Model.* 62.20 (2022), pp. 4863–4872. DOI: 10.1021/acs.jcim.2c00838.

[17] Neil, D., Segler, M., Guasch, L., Ahmed, M., Plumbley, D., Sellwood, M., Brown, N. *Exploring Deep Recurrent Models with Reinforcement Learning for Molecule Design*. 2018. URL: https://openreview.net/forum?id=HkcTe-bR- (visited on 05/06/2019).

[18] Schulman, J., Wolski, F., Dhariwal, P., Radford, A., Klimov, O. "Proximal Policy Optimization Algorithms". In: (2017). DOI: 10.48550/arXiv.1707.06347.

[19] Jensen, J. H. "A Graph-Based Genetic Algorithm and Generative Model/Monte Carlo Tree Search for the Exploration of Chemical Space". In: *Chem. Sci.* 10.12 (2019), pp. 3567–3572. DOI: 10.1039/C8SC05372C.

[20] Gao, W., Fu, T., Sun, J., Coley, C. W. "Sample Efficiency Matters: A Benchmark for Practical Molecular Optimization". In: *NeurIPS*. 2022.

[21] Yoshikawa, N., Terayama, K., Honma, T., Oono, K., Tsuda, K. "Population-Based de Novo Molecule Generation, Using Grammatical Evolution". In: *Chem. Lett.* (2018). DOI: 10.1246/cl.180665.

[22] Nigam, A., Pollice, R., Krenn, M., Gomes, G. d. P., Aspuru-Guzik, A. "Beyond Generative Models: Superfast Traversal, Optimization, Novelty, Exploration and Discovery (STONED) Algorithm for Molecules Using SELFIES". In: *Chem. Sci.* 12.20 (2021), pp. 7079–7090. DOI: 10.1039/D1SC00231G.

[23] Krenn, M., Häse, F., Nigam, A., Friederich, P., Aspuru-Guzik, A. "Self-Referencing Embedded Strings (SELFIES): A 100% Robust Molecular String Representation". In: *Mach. Learn.: Sci. Technol.* (2020). DOI: 10.1088/2632-2153/aba947.

[24] Xie, Y., Shi, C., Zhou, H., Yang, Y., Zhang, W., Yu, Y., Li, L. "MARS: Markov Molecular Sampling for Multi-objective Drug Discovery". In: *ICLR*. 2021.

[25] Xie, Y., Xu, Z., Ma, J., Mei, Q. "How Much Space Has Been Explored? Measuring the Chemical Space Covered by Databases and Machine-Generated Molecules". In: *ICLR*. 2023.

[26] Fu, T., Xiao, C., Li, X., Glass, L. M., Sun, J. "MIMOSA: Multi-constraint Molecule Sampling for Molecule Optimization". In: *AAAI*. 2021.

[27] Bengio, E., Jain, M., Korablyov, M., Precup, D., Bengio, Y. "Flow Network Based Generative Models for Non-Iterative Diverse Candidate Generation". In: *NeurIPS*. 2021.

[28] Pereira, T., Abbasi, M., Ribeiro, B., Arrais, J. P. "Diversity Oriented Deep Reinforcement Learning for Targeted Molecule Generation". In: *J. Cheminformatics* 13.1 (2021), p. 21. DOI: 10.1186/s13321-021-00498-z.

[29] Bjerrum, E. J., Margreitter, C., Blaschke, T., de Castro, R. L.-R. "Faster and More Diverse de Novo Molecular Optimization with Double-Loop Reinforcement Learning Using Augmented SMILES". In: *J. Comput. Aided. Mol. Des.* 37.8 (2023), pp. 373–394. DOI: 10.1007/s10822-023-00512-6.

[30] Liu, X., Ye, K., van Vlijmen, H. W. T., IJzerman, A. P., van Westen, G. J. P. "An Exploration Strategy Improves the Diversity of de Novo Ligands Using Deep Reinforcement Learning: A Case for the Adenosine A2A Receptor". In: *J. Cheminformatics* 11.1 (2019), p. 35. DOI: 10.1186/s13321-019-0355-6.

[31] Chen, B., Wang, T., Li, C., Dai, H., Song, L. "Molecule Optimization by Explainable Evolution". In: *ICLR*. 2020.

[32] Rupakheti, C., Virshup, A., Yang, W., Beratan, D. N. "Strategy To Discover Diverse Optimal Molecules in the Small Molecule Universe". In: *J. Chem. Inf. Model.* 55.3 (2015), pp. 529–537. DOI: 10.1021/ci500749q.

[33] Graff, D. E., Shakhnovich, E. I., Coley, C. W. "Accelerating High-Throughput Virtual Screening through Molecular Pool-Based Active Learning". In: *Chem. Sci.* 12.22 (2021), pp. 7866–7881. DOI: 10.1039/D0SC06805E.

[34] Tripp, A., Simm, G. N. C., Hernández-Lobato, J. M. "A Fresh Look at De Novo Molecular Design Benchmarks". In: *NeurIPS-AI4Science*. 2021.

2.3 Improving Few- and Zero-Shot Reaction Template Prediction Using Modern Hopfield Networks

JCIM JOURNAL OF CHEMICAL INFORMATION AND MODELING

pubs.acs.org/jcim Article

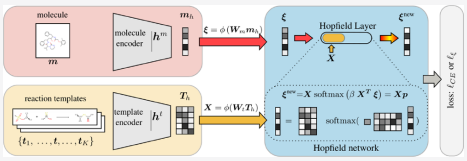
Improving Few- and Zero-Shot Reaction Template Prediction Using Modern Hopfield Networks

Philipp Seidl, Philipp Renz, Natalia Dyubankova, Paulo Neves, Jonas Verhoeven, Jörg K. Wegner, Marwin Segler, Sepp Hochreiter, and Günter Klambauer*

Cite This: J. Chem. Inf. Model. 2022, 62, 2111−2120 Read Online

ACCESS | Metrics & More | Article Recommendations | Supporting Information

ABSTRACT: Finding synthesis routes for molecules of interest is essential in the discovery of new drugs and materials. To find such routes, computer-assisted synthesis planning (CASP) methods are employed, which rely on a single-step model of chemical reactivity. In this study, we introduce a template-based single-step retrosynthesis model based on Modern Hopfield Networks, which learn an encoding of both molecules and reaction templates in order to predict the relevance of templates for a given molecule. The template representation allows generalization across different reactions and significantly improves the performance of template relevance prediction, especially for templates with few or zero training examples. With inference speed up to orders of magnitude faster than baseline methods, we improve or match the state-of-the-art performance for top-k exact match accuracy for $k \geq 3$ in the retrosynthesis benchmark USPTO-50k. Code to reproduce the results is available at github.com/ml-jku/mhn-react.



INTRODUCTION

The design of a new molecule starts with an initial idea of a chemical structure with hypothesized desired properties.[1] Desired properties might be the inhibition of a disease or a virus in drug discovery or thermal stability in material science.[2,3] From the design idea of the molecule, a virtual molecule is constructed, the properties of which can then be predicted by means of computational methods.[4,5] However, to test its properties and to finally make use of it, the molecule must be made physically available through chemical synthesis. Finding a synthesis route for a given molecule is a multistep process that is considered highly complex.[6,7]

To aid in finding synthesis routes, chemists have resorted to computer-assisted synthesis planning (CASP) methods.[6,8] Chemical synthesis planning is often viewed in the retrosynthesis setting in which a molecule of interest is recursively decomposed into less complex molecules until only readily available precursor molecules remain.[9] Such an approach relies on a single-step retrosynthesis model, which, given a product, tries to propose sets of reactants from which it can be synthesized. Early methods modeled chemical reactivity using rule-based expert systems.[8] These methods, however, require extensive manual curation.[9−11] Recently, there have been increased efforts to model chemical reactivity from reaction databases using machine learning methods.[9,12−15]

These efforts to model chemical reactions encompass a variety of different approaches. In one line of methods,[14,16−20] the simplified molecular-input line-entry system (SMILES) representation[21] of the reactants given that of the product is predicted, using architectures initially proposed for the translation between natural languages.[22,23] Others exploit the graph structure of molecules and model the task using graph neural networks.[24,25] A prominent line of work makes use of *reaction templates* which are graph transformation rules that encode connectivity changes between atoms during a chemical reaction.

In a template-based approach, reaction templates are first extracted from a reaction database or hand-coded by a chemist. If the product side of a template is a subgraph of a molecule, the template is called applicable to the molecule and can be used to transform it to a reactant set. However, even if a template can be applied to a molecule, the resulting reaction might not be viable in the laboratory.[11] Hence, a core task, which we refer to as template-relevance prediction, in such an approach is to predict with which templates a molecule can be combined with to yield a viable reaction. In prior work, this problem has often been tackled using machine learning methods that are trained at this task on a set of recorded reactions.[9,11,26−32]

Template-based methods usually view the problem as a classification task in which the templates are modeled as distinct categories. However, this can be problematic as automatic template extraction leads to many templates that are

Special Issue: From Reaction Informatics to Chemical Space

Received: September 6, 2021
Published: January 15, 2022

https://doi.org/10.1021/acs.jcim.1c01065
J. Chem. Inf. Model. 2022, 62, 2111−2120

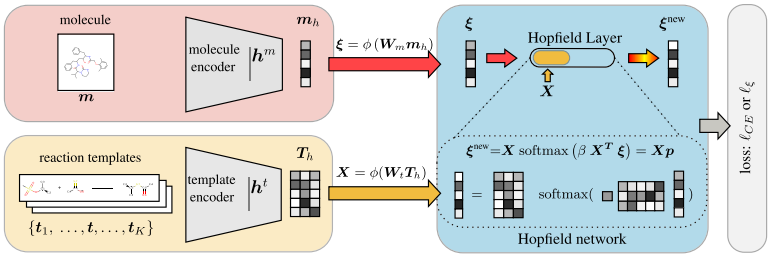


Figure 1. Simplified depiction of our approach. Standard approaches only encode the molecule and predict a fixed set of templates. In our modern Hopfield network (MHN)-based approach, the templates are also encoded and transformed to stored patterns via the template encoder. The Hopfield layer learns to associate the encoded input molecule, the state pattern $\boldsymbol{\xi}$, with the memory of encoded templates, the stored patterns $\mathbf{X}$. Multiple Hopfield layers can operate in parallel or can be stacked using different encoders.

represented by few training samples,[9,26] Somnath et al.[25] argued that template-based approaches suffer from bad performance, particularly for rare reaction templates. Segler[9] and Struble et al.[10] noted that machine learning (ML) has not been applied successfully for CASP in low-data regimes. To address the low-data issue, Fortunato et al.[26] pretrained their template-relevance model to predict which templates are applicable and then fine-tuned it on recorded reactions in a database. This improved template-relevance prediction, especially for rare templates, as well as the average applicability of the top-ranked templates. Overall, a challenge of template-based methods arises from modeling reaction templates as distinct categories, which leads to many classes with few examples (see the section entitled "Methods").

To avoid the above-mentioned problems, we propose a new model that does not consider templates as distinct categories, but can leverage structural information about the template. This allows for generalization over templates and improves performance in the tasks defined in ref 26, especially for templates with few training samples and even for unseen templates. This model learns to associate relevant templates to product molecules using a modern Hopfield network (MHN).[33,34] To this end, we adapted MHNs to associate objects of different modalities, namely input molecules and reaction templates. A depiction of our approach is illustrated in Figure 1.

In contrast to popular ML approaches, in which variable or input-dependent subsets of the data are associated,[22,33,35,36] our architecture maintains a fixed set of representations, considered as a static memory independent of the input.

In this study, we propose a template-based method, which are often reported to be computationally expensive, because of the NP-complete subgraph-isomorphism calculations involved in template execution.[24−26,28] To address this issue Fortunato et al.,[26] Bjerrum et al.[28] trained neural networks to predict which templates are applicable, given a molecule to filter inapplicable templates during inference. We find that using a substructure screen, i.e., a fast check of a necessary condition for a graph to be a subgraph of another improves inference speed, which may also be of interest for other template-based methods.

The main advance of our model over Fortunato et al.,[26] Hasic and Ishida,[37] or other template-based methods, is that by representing and encoding reaction templates we are able to predict relevant templates, even if few training data is available, which is a common issue in reaction datasets.

This work is structured as follows: In the "Methods" section, we propose a template relevance model that predicts template relevance by applying a multimodal learning approach using a modern Hopfield network. In the sections entitled "Template Prediction" and "Single-Step Retrosynthesis", we demonstrate that our architecture improves predictive performance for template relevance prediction and single-step retrosynthesis. In the section enetitled "Inference Speed", we show that our method is several times faster than baseline methods.

SINGLE-STEP RETROSYNTHESIS

The goal of *single-step retrosynthesis* is to predict sets of molecules that react to a given product.[7,38] Since a molecule can be synthesized in various ways, this represents a one-to-many task. Performance in this setting is usually measured by *reactant top-k accuracy* using a reaction database. This metric measures the fraction of samples for which, given the product of a recorded reaction, the recorded reactants are among the top-k predictions. Given the one-to-many setting, small values of k might not be an optimal choice as there might exist scenarios where a good model receives low scores. Choosing a large k might result in a metric that is overly easy to optimize.

Template-based approaches predict reactant sets via reaction templates. A reaction template encodes atom connectivity changes during a chemical reaction and can be used to transform a product molecule to reactants, $\boldsymbol{m} \xrightarrow{\boldsymbol{t}} \boldsymbol{r}$, where $\boldsymbol{m}$ is a product molecule, $\boldsymbol{r}$ represents a set of reactants and $\boldsymbol{t}$ a reaction template. The product side of a template encodes at which position in a molecule the template can be applied. A necessary condition for this is that the product side of the template is a substructure of the molecule of interest. If this is the case, a template is said to be *applicable* to the molecule. The product subgraph is then transformed according to the reactant side of the template and an atom-mapping between the two sides. Templates can be either hand-coded or automatically extracted from reaction databases, which yields an ordered set of K unique templates $\boldsymbol{T} = \{\boldsymbol{t}^k\}_{k=1}^K$.

The aim of *template-relevance prediction* is to predict which templates result in a feasible reaction given a product. If this is the case, we say that a template is *relevant* to a molecule. While applicability is a necessary condition for relevance, it ignores the context of the whole molecule and thus substructures that might conflict with the encoded reaction (see Figure 1 in

https://doi.org/10.1021/acs.jcim.1c01065
J. Chem. Inf. Model. 2022, 62, 2111−2120

Segler and Waller[11]). In practice, applicability gives poor performance at relevance prediction (see Table 1, presented later in this work). To evaluate template-relevance predictions, we use *template top-k accuracy*, which given the product of a recorded reaction measures the fraction of samples for which the template extracted from the recorded reaction is among the top-k predicted ones.

Given relevance predictions for a product, reactant sets are obtained by executing top-scoring templates. We do not permit relevance prediction to rely on applicability calculations, because it is relatively slow to compute. Via this constraint, template top-k accuracy also incorporates information about the models ability to filter out nonapplicable templates. This information might be lost in reactant accuracy as template execution relies on a check for applicability. Other differences between the reactant/template accuracy can arise from multiple locations in which the correct template may be applied or incorrect templates leading to the correct reactants.

Multistep retrosynthesis can be achieved by applying single-step retrosynthesis recursively. One can decompose the desired molecule into less-complex molecules until only readily available precursor molecules remain.

METHODS

Motivation. Many template-based methods[9,11,26−28] consider a classification problem and predict templates using

$$\hat{\boldsymbol{y}} = \text{softmax}(\boldsymbol{W}\boldsymbol{h}^m(\boldsymbol{m})) \tag{1}$$

where $\boldsymbol{h}^m(\boldsymbol{m})$ is a neural network (NN) that maps a molecule representation to a vector of size d_m, which we call *molecule encoder*. $\boldsymbol{W}$ is a randomly initialized matrix, the last layer of the NN mapping from the molecule encoder to the predictive score of the template classes. Multiplication with $\boldsymbol{W} \in \mathbb{R}^{K\times d_m}$ yields a score for each template $\boldsymbol{t}_1, \cdots, \boldsymbol{t}_K$. These scores are then normalized using the softmax function, which yields the vector $\hat{\boldsymbol{y}} \in \mathbb{R}^K$. In this setting, different templates are viewed as distinct categories or classes, which makes the model ignorant of similarities between classes, which prevents generalization over templates. The high fraction of samples in reaction datasets that have a unique template can be problematic because they cannot contribute to performance. This problem might also appear for templates occurring only a few times, but to a lesser extent.

Instead of learning the rows of $\boldsymbol{W}$ independently, one could map each template to a vector of size d_t using a *template encoder*, $\boldsymbol{h}^t$, and concatenate them row-wise to obtain $\boldsymbol{T}_h = \boldsymbol{h}^t(\boldsymbol{T}) \in \mathbb{R}^{K\times d_t}$. If $d_m = d_t$, replacing $\boldsymbol{W}$ in the equation above yields

$$\hat{\boldsymbol{y}} = \text{softmax}(\boldsymbol{T}_h\boldsymbol{h}^m(\boldsymbol{m})) \tag{2}$$

which associates the molecule m with each template via the dot product of their representations. This allows generalization across templates, because the structure of the template is used to represent the class and the model can leverage similarities between templates. We adapt modern Hopfield networks[33] to generalize this association of the two modalities, molecules and reaction templates.

Modern Hopfield Networks. By going from eq 1 to eq 2, we have recast the problem of classifying a given molecule into a reaction template class into a content-based retrieval problem. Given a molecule $\boldsymbol{m}$, the correct address, or index, of the molecule's associated template $\boldsymbol{t}$ in a database of templates $\boldsymbol{T}$ must be retrieved based on the chemical structure of the molecule. Such content-addressable, so-called associative memory systems realized as neural networks are called Hopfield networks.[39,40] Their storage capacity can be considerably increased by polynomial terms in the energy function.[41−48] In contrast to these binary memory networks, we use continuous associative memory networks with very high storage capacity. These modern Hopfield networks for deep learning architectures have an energy function with continuous states and can retrieve samples with only one update.[33,49]

For tackling retrieval problems, modern Hopfield networks perform several operations with so-called patterns, i.e., vector representations of the data points. A retrieval model based on a modern Hopfield network can be considered as a function g that returns the position $\hat{\boldsymbol{y}}$ of the retrieved pattern

$$\hat{\boldsymbol{y}} = g(\boldsymbol{h}^m(\boldsymbol{m}), \boldsymbol{h}^t(\boldsymbol{T})) \tag{3}$$

The structure of the function g can be relatively complex,[33,34] but consists of two main components: (a) a mapping to a d-dimensional associative space using linear embeddings $\boldsymbol{W}_m$ and $\boldsymbol{W}_t$ followed by a a non/linear activation ϕ. With these mappings, the state pattern $\boldsymbol{\xi} = \phi(\boldsymbol{W}_m\boldsymbol{h}^m(\boldsymbol{m}))$ and stored patterns $\boldsymbol{X} = \phi(\boldsymbol{W}_t\boldsymbol{h}^t(\boldsymbol{T}))$ are obtained. (b) An update function that performs the following operation,

$$\boldsymbol{\xi}^{\text{new}} = \boldsymbol{X}\boldsymbol{p} = \boldsymbol{X}\ \text{softmax}(\beta\boldsymbol{X}^{\text{T}}\boldsymbol{\xi}) \tag{4}$$

where $\boldsymbol{\xi}^{\text{new}}$ is the retrieved pattern and the stochastic vector $\mathbf{p}$ associates the state pattern with the stored patterns and $\beta > 0$ is a scaling parameter (inverse temperature). The stored patterns $\boldsymbol{X}$ can be considered a memory of reaction templates. Other components, such multiple mappings to associative spaces in parallel, so-called heads, and iterative refinement of the retrieved patterns across multiple layers in the form of stacking are suggested by the powerful transformer architectures,[22] which are also based on Hopfield networks.[33,34] Multiple layers of parallel heads have been shown to be necessary for high predictive quality at natural language processing tasks.[22] This design of the function g allows one to build a DL architecture that is potentially able to retrieve a correct reaction template from an arbitrary set of templates given a molecule.

All these above-mentioned operations and architectural components are implemented in the so-called "Hopfield layer",[33] whose design we use in our model and whose concrete settings are determined during hyperparameter selection. The matrices $\boldsymbol{W}_t$ and $\boldsymbol{W}_m$ that associate molecules and templates are learned during training of the model. In the following, we provide details on the architecture.

Model Architecture. Our model architecture consists of three main parts: (a) a molecule encoder, (b) a reaction template encoder, and (c) one or more stacked or parallel Hopfield layers. First, we use a molecule encoder function that learns a relevant representation for the task at hand. For this, we use a fingerprint-based, e.g., extended connectivity fingerprint (ECFP),[50] fully connected NN, $\boldsymbol{h}^m_{\boldsymbol{w}}(\boldsymbol{m})$ with weights $\boldsymbol{w}$. The molecule encoder maps a molecule to a representation $\boldsymbol{m}_h = \boldsymbol{h}^m_{\boldsymbol{w}}(\boldsymbol{m})$ of dimension d_m.

Second, we use the reaction template encoder $\boldsymbol{h}^t_{\boldsymbol{v}}$ with parameters $\boldsymbol{v}$ to learn relevant representations of templates. Here, we also use a fully connected NN with *template fingerprints* as input. These fingerprints are described in Section S3 in the Supporting Information. This function is

https://doi.org/10.1021/acs.jcim.1c01065
J. Chem. Inf. Model. 2022, 62, 2111−2120

applied to all templates T and the resulting vectors are concatenated column-wise into a matrix $T_h = h_v^t(T)$ with shape (d_v, K).

Finally, we use a single or several stacked or parallel Hopfield layers $g(.,.)$ to associate a molecule with all templates in the memory. Hopfield layers consist of the option of layer normalization[51] for ξ and X, which is included as a hyperparameter. We also consider the scaling parameter β as a hyperparameter. The Hopfield layer then employs the update rule described by eq 4 through which the updated representation of the product molecule ξ^{new} and the vector of associations p is obtained. If multiple Hopfield layers are stacked, ξ^{new} enters the next Hopfield layer, for which additional template encoders supply the template representations. Parallel Hopfield layers use the same template encoder, but learn different projections W_v, W_m, which is analogous to the heads in Transformer networks.

The simple model (eq 2) is a special case of our MHN and recovered if (a) W_t and W_m are the identity matrices and $d_t = d_m = d$, (b) the Hopfield network is constrained to a single update, (c) Hopfield networks are not stacked, i.e., there is only a single Hopfield layer, (d) the scaling parameter $\beta = 1$, (e) layer norm learns zero mean and unit variance and does not use its adaptive parameters, and (f) the activation function ϕ is the identity. The standard deep neural network (DNN) model (eq 1) is recovered if additionally the reaction templates are one-hot encoded, and the template encoder is linear.

In this study, we tested fingerprint-based fully connected networks for the molecule and template encoder. In principle, one could use any mapping from molecules/templates to vector-valued representations for these components, for example, raw fingerprints, graph neural networks[52] or SMILES arbitrary target specification (SMARTS)-based RNNs,[53] or Transformers.[22]

Loss Function and Optimization. Given a training pair (m,t) and the set of all templates T, the model should assign high probability to t, based on m and T. We encode this objective by the negative log-likelihood: $-\log p(t|m,T)$. The probability of each template in T is encoded by the corresponding element of the vector of associations p of the last Hopfield layer. In the simple case of a single correct template, this is equivalent to the cross-entropy loss $l_{CE}(y,p)$ between the one-hot encoded label vector y and the predictions p. In the case of multiple parallel Hopfield layers, we use average pooling across the vectors p supplied from each layer. We provide a general definition of the loss in terms of retrieved patterns and details in the section entitled "Related Work".

The parameters of the model are adjusted on a training set using stochastic gradient descent on the loss with respect to W_t, W_m, w, v via the AdamW optimizer.[54] We train our model for a maximum of 100 epochs and then select the best model with respect to the minimum cross-entropy loss in the case of template-relevance prediction or maximum top-1 accuracy for single-step retrosynthesis on the validation set.

We use dropout regularization in the molecule encoder h^m for the template encoder h^t, as well as for the representations in the Hopfield layers. We employ L2 regularization on the parameters. A detailed list of considered and selected hyperparameters is given in Tables S2 and S3 in the Supporting Information.

We added a computationally inexpensive fingerprint-based substructure screen as a post-processing step that can filter out a part of the nonapplicable templates. For each product and the product side of each template, we calculated a bit-vector using the "PatternFingerprint" function from RDKit.[55] Each bit set in this vector specifies the presence of a substructure. For a template to be applicable, every bit set in the template fingerprint also must be set in the product fingerprint, which is a necessary condition for subgraphs to match. We chose a fingerprint size of 4096, as we did not observe significant performance gains for larger sizes, as can be seen in Figure S2 in the Supporting Information.

Related Work. From the perspective of attention-based machine learning,[22] our model can be seen as an attention mechanism that learns to attend elements or patterns of an external memory. The model proposed in Dai et al.[29] is similar to our approach, because it also makes use of the templates' structures and could even be seen as a special case, in which the memory of reaction templates is assembled based on logical operators. There are further restrictions on the structures of the encoder networks, which our work demonstrates are unnecessary. Because of the fact that representations of data from two different modalities, reactions and molecules, are learned, our approach also resembles the contrastive learning approaches taken in CLIP[36] and ConVIRT,[56] in which associated pairs of images and texts are contrasted against nonassociated pairs. Our adaption of MHN to maintain a static memory complements previous contrastive learning[35,57] approaches using a memory.[58−61] To embed molecule and reaction representations in the same latent space by maximizing the cosine similarity of reactions relevant to a given molecule has also been suggested by Segler.[9] We also considered a contrastive learning setting using the InfoNCE loss;[35] however, this led to slightly inferior results (see the Supporting Information).

Data and Preprocessing. All datasets used in this study are derived from the United States Patent and Trademark Office (USPTO) dataset, extracted from the U.S. patent literature between 1976 and September 2016 by Lowe.[62] This dataset contains 1.8 million text-mined reaction equations in SMILES notation[21] and consists of reactions recorded in the years from 1976 to 2016. Reaction conditions and process actions are not included. For evaluating template relevance prediction, we use the preprocessing procedure described in ref 26. Templates are extracted using rdchiral.[63] This results in two datasets: *USPTO-sm*, which is based on USPTO-50k,[64] and *USPTO-lg*, which is based on *USPTO-410k*.[65] *USPTO-50k* contains only the 50 most populated reaction types, yielding a much simpler dataset than *USPTO-lg*, which is more diverse and entails multireactant reactions, which leads to many different templates

For evaluating single-step retrosynthesis, we use *USPTO-50k* as preprocessed in ref 31. For this set, we also extract templates using rdchiral,[63] but only for the train and validation split, to prevent test data leakage. Figure 2 displays the fraction of samples for different template frequencies for *USPTO-sm*/*USPTO-lg*. A detailed description of the datasets and their preprocessing can be found in Section S3 in the Supporting Information.

EXPERIMENTS AND RESULTS

Template Prediction. In this section, we evaluate different models in the setup by Fortunato et al.[26] Here, the aim is to predict the correct reaction templates, with template top-k accuracy used as a metric. In contrast to reactant prediction,

https://doi.org/10.1021/acs.jcim.1c01065
J. Chem. Inf. Model. 2022, 62, 2111−2120

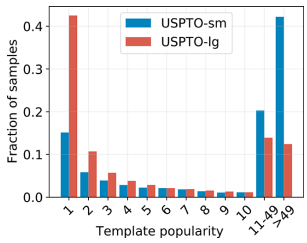


Figure 2. Histogram showing the fraction of samples for different template frequencies. The leftmost red bar indicates that over 40% of chemical reactions of *USPTO-lg* have a unique reaction template. The majority of reaction templates are rare.

this allows a more fine-grained analysis of the template ranking obtained by the models, because it ignores errors stemming from multiple potential application locations. The evaluation of our model in the full reactant prediction task is delayed to the next section. In their study, Fortunato et al.[26] mainly compared two models. First, a fully connected network with a softmax output in which each output unit corresponds to a reaction template, conceptually similar to the model introduced in ref 11. We refer to this model as DNN. The second method is identical to the above, but instead of randomly initializing the weights, pretraining on a template applicability task (DNN +pretrain) is done. We extend this model by the addition of a template encoder and the MHN to associate the entities. We refer to models of the latter type as MHN while calling the former DNN. We also introduced the fingerprint filter (FPF) as a post-processing step. The choice of (a) model type, (b) the use of the FPF, and (c) the pretraining results in $2^3 = 8$ model variants. For all model variants, hyperparameters were adjusted on the validation set, as described in Section S3. We start with a general performance analysis and then investigate how rare templates affect the performance.

Overall Performance. The upper section of Table 1 shows the performance of these eight variants on *USPTO-sm* and *USPTO-lg*. Overall, it can be seen that the use of MHN and FPF yields large performance improvements over the methods evaluated in ref 26. Most notably, the top-100 accuracy increases by 10% on *USTPO-sm* and 18% on *USPTO-lg*. The plain MHN model without both FPF or pretraining has higher top-k accuracy for most values of k and both datasets, except for top-1 accuracy on *USPTO-lg*, showing the isolated performance gains by the model type. We will further investigate where these performance differences stem from below. Furthermore, the FPF yields non-negligible accuracy improvements for all models. Interestingly, pretraining and the FPF seem to complement each other in predicting applicability for the DNN models, rather than one of them being redundant. While pretraining yields non-negligible performance increases for the DNN models, the effect on the MHN model performance seems rather limited.

In the lower part of Table 1, we show the performance of a simple popularity baseline. This baseline predicts templates based on their occurrence in the training set. The last row shows that a plain applicability check is not sufficient for high performance. We include additional results in Section S3.

Rare Templates: Few- and Zero-Shot Learning. Given the propensity of rare template samples in the used datasets, we next show how the predictive performance varies with template popularity. Figure 3 shows the top-100 accuracy for different subsets of the test set, which were grouped according to the number of training samples with the same template. For improved clarity, we only include four of the above methods: DNN, DNN+pretrain, MHN+FPF, and the popularity baseline. Figure S1 in the Supporting Information shows all eight model variants. Especially for samples with rare templates, the performance gap between our method and the compared ones is large. As expected, in the experiments on template relevance prediction (see the section entitled "Template Prediction" and Figure 3), the DNN models and the popularity baseline perform poorly for samples with templates not seen during training. The MHN model, on the other hand, achieves far above random accuracy on these samples by generalizing over templates. Because of the large fraction of rare template samples, the overall performance is strongly dependent on these.

Table 1. Template Top-k Accuracy (%) of Different Method Variants on *USPTO-sm* and *USPTO-lg**

				USPTO-sm			USPTO-lg		
Ref.	Model	Filter	Pretrain	Top-1	Top-10	Top-100	Top-1	Top-10	Top-100
[11]	DNN	-	no	38.1[a]	64.1[a]	76.5[a]	16.0[b]	35.7[b]	50.7[b]
[26]	DNN	-	yes	38.5[a]	69.1[a]	85.8[a]	20.8[b]	41.7[b]	54.2[b]
	DNN	FPF	no	39.0[a]	67.6[a]	84.6[a]	17.1[b]	38.1[b]	53.6[b]
	DNN	FPF	yes	38.9[a]	71.2[a]	90.6[a]	**21.5**[b]	43.0[b]	56.0[b]
	MHN	-	no	39.9[a]	75.7[a]	91.9[a]	16.7[b]	43.6[b]	71.4[b]
	MHN	-	yes	**40.4**[a]	76.2[a]	91.8[a]	16.7[b]	43.5[b]	71.4[b]
(ours)	MHN	FPF	no	**40.5**[a]	**78.7**[a]	**95.9**[a]	16.9[b]	**44.2**[b]	**72.4**[b]
	MHN	FPF	yes	**41.3**[a]	**78.8**[a]	**95.7**[a]	17.0[b]	**44.1**[b]	**72.3**[b]
	Pop	-	no	0.0[a]	8.6[a]	28.9[a]	0.1[b]	0.8[b]	3.5[b]
	Pop	FPF	no	1.5[a]	17.6[a]	53.1[a]	0.3[b]	1.9[b]	7.5[b]
	Pop	App[c]	no	9.4[a]	39.6[a]	80.3[a]	1.1[b]	5.1[b]	16.5[b]

*"Model" indicates how the templates were ranked. "Filter" specifies if and how templates were excluded from the ranking via FPF or an applicability check (App). Pre-train indicates whether a model was pre-trained on the applicability task. Error bars represent confidence intervals on binomial proportions. The gray rows indicate methods specifically proposed here or in prior work. [a]Width of 95% confidence interval <1.3%. [b]Width of 95% confidence interval <0.4%. [c]Note that the applicability filter violates modeling constraints from the section entitled "Single-Step Retrosynthesis".

https://doi.org/10.1021/acs.jcim.1c01065
J. Chem. Inf. Model. 2022, 62, 2111−2120

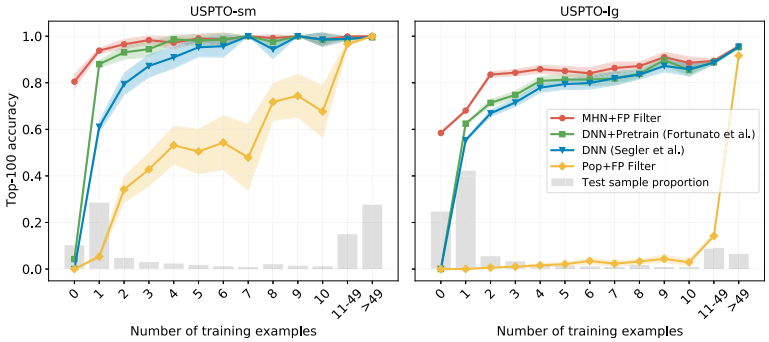


Figure 3. Top-100 accuracy for different template popularity on the *USPTO-sm*/*USPTO-lg* datasets. The gray bars represent the proportion of samples in the test set. Error bars represent 95% confidence intervals on binomial proportion. Our method performs especially well on samples with reaction templates with few training examples.

Table 2. Reactant Top-*k* Accuracy (%) on *USPTO-50k* Retrosynthesis[a]

Abbr.	Ref.	Cat.	Top-1	Top-3	Top-5	Top-10	Top-20	Top-50
MHNreact	ours	tb	$51.8^{\pm.2}$	$\mathbf{74.6}^{\pm.3}$	$\mathbf{81.2}^{\pm.2}$	$\mathbf{88.1}^{\pm.2}$	$\mathbf{92.0}^{\pm.1}$	$\mathbf{94.0}^{\pm.0}$
Neuralsym	11	tb	$45.2^{\pm.2}$	$67.9^{\pm.5}$	$75.8^{\pm.2}$	$83.5^{\pm.2}$	$89.1^{\pm.1}$	$93.5^{\pm.1}$
Pop		tb	18.4	38.7	48.6	63.0	75.8	89.8
Transformer		tf	43.7	59.7	65.1	70.1	73.5	75.0
Dual-TB	66	tb	**55.2**	**74.6**	80.5	86.9		
Dual-TF	66	tf	53.3	69.7	73.0	75.0		
ATx100	20	tf	53.5		81.0	85.7		
GLN	29	tb	52.5	69.0	75.6	83.7	89.0	92.4
RetroPrime	67	tf	51.4	70.8	74.0	76.1		
G2G	24	tf	48.9	67.6	72.5	75.5		
MEGAN	68	tf	48.1	70.7	78.4	86.1	90.3	93.2
GET-LT1	69	tf	44.9	58.8	62.4	65.9		
Neuralsym	11,29	tb	44.4	65.3	72.4	78.9	82.2	83.1
GOPRO	70	tf	43.8	57.2	61.4	66.6		
SCROP	71	tf	43.7	60.0	65.2	68.7		
LV-Trans	72	tf	40.5	65.1	72.8	79.4		
Trans	19	tf	37.9	57.3	62.7			
Retrosim	31	tb	37.3	54.7	63.3	74.1	82.0	85.3

[a]Data taken from refs 11, 19, 20, 24, 29, 31, and 66−72. Bold values indicate values within 0.1 of the maximum value, green denotes a value within 1 percentage point of the maximum value, and yellow denotes a value within 3 percentage points to the maximum value. Error bars represent standard deviations across five reruns. Category ("Cat.") indicates whether a method is template-based (tb) or template-free (tf). Methods in the upper part have been (re-)implemented in this work.

Learning from Rare Templates. Next, we analyze the effect on performance of rare template samples in the training set, as opposed to those in the test set. In a classification setting, it is only useful to include classes if they are recurring, i.e., represented by more than one sample. However, in the *USPTO-sm*/*USPTO-lg* datasets, many templates occur only once (see Figure 2). If the templates are modeled as categories, as done in the DNN approach, a large fraction of samples cannot contribute to performance. However, this does not hold for models that can generalize across templates, as our MHN model is able to do. To show the effect of the rare template samples on learning, we use the following experiment on *USPTO-sm*: We removed all samples with templates that are *exactly once* in the training set and *not* in the test set and retrain the best DNN and MHN models of the template relevance prediction experiment. After removal of these samples, the top-10 accuracy rose from 71.2 ± 0.2 to 72.3 ± 0.2 for the DNN +pretrain model and dropped from 78.8 ± 0.4 to 73.7 ± 0.3 for the MHN model. As expected, the performance does not drop for the DNN model, but even improved marginally, which we attribute to the model knowing which templates do not occur in the test set. In contrast, the performance for the MHN model decreased. This shows that the increased performance of our approach is in part caused by the larger fraction of data that can be leveraged for learning.

Single-Step Retrosynthesis. Next, we compare our method to previously suggested ones in the single-step retrosynthesis task using the *USPTO-50k* and *USPTO-full* datasets. We followed the preprocessing procedure of ref 13 and used rdchiral[63] to extract reaction templates. Following ref

https://doi.org/10.1021/acs.jcim.1c01065
J. Chem. Inf. Model. 2022, 62, 2111−2120

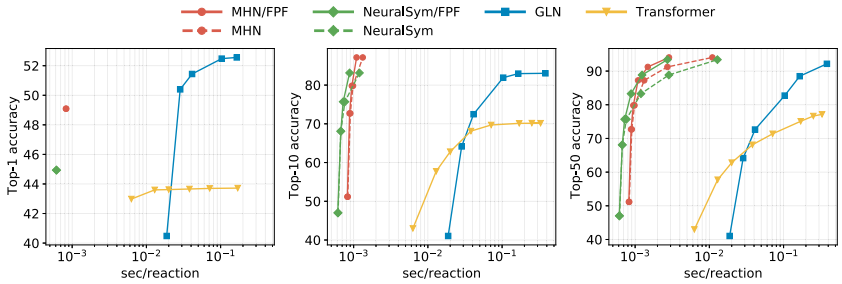


Figure 4. Reactant top-*k* accuracy versus inference speed for different values of *k*. Upper left is better. For Transformer/GLN, the points represent different beam sizes. For MHN/NeuralSym, the points reflect different numbers of generated reactant sets, namely, {1, 3, 5, 10, 20, 50}. In case of a Transformer, the points depict different beam sizes: {1, 3, 5, 10, 20, 50, 75, 100}, from left to right.

13, we shuffled the data and assigned 80%/10%/10% of the samples in each reaction class into train/validation/test set, respectively. This is similar to USPTO-sm above but varies in details discussed in section S3. Also, in contrast to template relevance, we optimize for maximum top-1 accuracy, which results in different selected hyperparameters. In addition, we report results of a single run on the *USPTO-full* dataset preprocessed by Tetko et al.[20] (see Table S4 in the Supporting Information). We first compare the predictive performance of our method to previous ones and then investigate its inference speed.

Predictive Performance. Table 2 shows the reactant top-*k* accuracies on *USPTO-50k* for different methods. These methods include, among others, transformer-based,[20,68] graph-to-graph[67] or template-based ones.[29] Some methods[18,25,37,73−77] that also report results on *USPTO-50k* have been omitted here, either because of test set leakage or different evaluation conditions, as detailed in Section S3. We reimplemented and improved the NeuralSym method as described in Section S3 and added the popularity baseline described in the section entitled "Template Prediction". Hyperparameter selection on the validation set returned an MHN model with two stacked Hopfield layers, which we refer to as *MHNreact* (see Section S3). We ranked reactant sets by the score of the template used to produce them. If a template execution yielded multiple results, all were included in the prediction in random order. Our method achieved state-of-the-art performance for $k \geq 3$ and approaches it for $k = 1, 3$. Together with Dual-TB,[66] this puts template-based methods ahead of other approaches at all considered values of *k*.

Without the canonicalization procedure of the product-SMILES from the mapped reaction smiles, we obtained a significant increase in performance (Top-1 accuracy of 59.04%). This might suggest leakage, as observed in ref 73, or signal getting lost from canonicalization procedure. This is apparent when using mini-hash fingerprint (MHFP),[78] a SMILES-based fingerprint. For all our experiments, we canonicalize the product-SMILES.

Inference Speed. Aside from predictive performance, inference speed is also vital for retrosynthesis methods. Therefore, CASP methods are often evaluated by their ability to find a route in a given time.[9,28,79] Template-based methods are sometimes reported to be slow;[24,68] however, we found that inference speed was not reported in mentioned studies and generally are seldom reported, despite their importance. Accuracy can be traded for inference speed for many models. For some, this tradeoff is achieved by varying the beam size.[20,29] In template-based approaches, the number of executed templates can be varied and traded off against speed. We compared inference speed of our MHN method with the following baselines. We obtained results for a graph logic network (GLN) from their paper.[24] We trained a Transformer baseline using the code of ref 14, as a representative of transformer-based methods.[19,20,72] In addition, we also include the NeuralSym[11] model that we implemented in the comparison. The results are displayed in Figure 4. At comparable or better performance, our method achieves inference speed of up to two magnitudes faster, compared to the Transformer and GLN. While NeuralSym is faster than our model for some fixed values of accuracy, MHN yields better maximum accuracy with comparable speed.

Computation Time and Resources. All experiments were run on different servers with diverse Nvidia GPUs (Titan V 12GB, P40 24 GB, V100 16GB, A100 20GB MIG), using PyTorch 1.6.0.[80] We estimate the total run time, for all experiments we performed for this study, including baselines, to be ∼1000 GPU hours. A single MHN model can be trained on *USPTO-50k* within ∼5 min on a V100.

■ DISCUSSION AND CONCLUSION

In this work, we have reformulated the problem of template-based reaction and retrosynthesis prediction as a retrieval problem. We introduced a deep learning architecture for reaction template prediction, based on using modern Hopfield networks. The proposed architecture consists of a molecule encoder, a reaction template encoder network, and Hopfield layers. The best network architectures that were found during hyperparameter selection are typically relatively lightweight, with one or two stacked Hopfield layers, compared to the sizable Transformer architectures.

The retrieval approach enables generalization across templates, which makes zero-shot learning possible and improves few-shot learning. On the single-step retrosynthesis benchmark *USPTO-50k*, our MHN model reaction reaches the state-of-the-art at top-*k* accuracy for $k \geq 3$. Furthermore, we falsify the common claim of template-based methods being slow.

We note that the current *USPTO-50k* benchmark and its great emphasis on top-1 accuracy for single-step retrosynthesis might only reflect part of what is needed for single-step

https://doi.org/10.1021/acs.jcim.1c01065
J. Chem. Inf. Model. 2022, 62, 2111−2120

retrosynthesis. In cases where a molecule can be made by multiple routes, top-1 accuracy might be too ambiguous; however, the evaluation unrealistically expects to use the one that is present in the dataset. We further argue that there is a tradeoff between having diverse results and having accurate results.[81,82] Unfortunately, it is currently hard to measure diversity, because of not having multiple correct ground-truth answers per product molecule.

Our experiments are currently limited by several factors. We did not investigate the importance of radius around the reaction center used for template extraction. We currently do not rerank reactants based on a secondary model, such as an in-scope filter[9] or dual models,[66] which could increase performance. There would also be several other hyperparameters to be explored, such as the template encoding, whose exploration could lead to an improvement of our method. The results for inference speed are dependent highly on implementation and may potentially be improved by relatively simple means, which was not the primary focus and is left for future work.

Nevertheless, we envision that our approach will be used to improve CASP systems or synthesis-aware generative models.[83−87]

■ ASSOCIATED CONTENT

Supporting Information

The Supporting Information is available free of charge at https://pubs.acs.org/doi/10.1021/acs.jcim.1c01065.

> Additional background information and related work (Section S2); details on experiments (Section S3); and additional results for USPTO-full, as well as, e.g., results for a chronological split for USPTO-50k (Section S4); visualizations on a down projection of the association space of reaction templates (Section S5); showcasing of several exemplary test-set samples of USPTO-50k and a reactant ranking comparison, to illustrate differences between methods (Section S6); an alternative view on the loss and an extended formulation of the algorithm as pseudocode (Section S7); details on experiments (e.g., dataset preprocessing, training, and hyperparameter selection, feature extraction from molecules, as well as details on template-fingerprint featurization can be found in the Supporting Information document (PDF)

■ AUTHOR INFORMATION

Corresponding Author

Günter Klambauer − *ELLIS Unit Linz, LIT AI Lab, Institute for Machine Learning, Johannes Kepler University Linz, Linz, Austria 4040*; Email: klambauer@ml.jku.at

Authors

Philipp Seidl − *ELLIS Unit Linz, LIT AI Lab, Institute for Machine Learning, Johannes Kepler University Linz, Linz, Austria 4040*; orcid.org/0000-0003-4333-2040

Philipp Renz − *ELLIS Unit Linz, LIT AI Lab, Institute for Machine Learning, Johannes Kepler University Linz, Linz, Austria 4040*; orcid.org/0000-0002-3323-7632

Natalia Dyubankova − *Janssen Pharmaceutica NV, High Dimensional Biology and Discovery Data Sciences, Janssen Research & Development, Beerse, Belgium 2340*

Paulo Neves − *Janssen Pharmaceutica NV, High Dimensional Biology and Discovery Data Sciences, Janssen Research & Development, Beerse, Belgium 2340*

Jonas Verhoeven − *Janssen Pharmaceutica NV, High Dimensional Biology and Discovery Data Sciences, Janssen Research & Development, Beerse, Belgium 2340*

Jörg K. Wegner − *Janssen Research & Development, LLC, In-Silico Discovery and External Innovation (ISD&EI), Cambridge, Massachusetts 02142, United States;* orcid.org/0000-0002-1852-9434

Marwin Segler − *Microsoft Research, Cambridge, United Kingdom CB1 2FB;* orcid.org/0000-0001-8008-0546

Sepp Hochreiter − *ELLIS Unit Linz, LIT AI Lab, Institute for Machine Learning, Johannes Kepler University Linz, Linz, Austria 4040; Institute of Advanced Research in Artificial Intelligence, Wien, Austria 1030*

Complete contact information is available at: https://pubs.acs.org/10.1021/acs.jcim.1c01065

Funding

This work was supported by Flanders Innovation & Entrepreneurship (VLAIO) with the project grant (No. HBC.2018.2287) (MaDeSMart).

Notes

The authors declare no competing financial interest.

Data and Software Availability: Python code and instructions to reproduce the predictive results of template prediction as well as single-step retrosynthesis are available at https://github.com/ml-jku/mhn-react under the BSD 2-Clause license. Instructions to prepare the programming-environment, as well as to download the data and run the training, inference, and evaluation procedure can be found there. The code was run on different servers with diverse Nvidia GPUs using PyTorch 1.6.0.[80]

■ ACKNOWLEDGMENTS

The ELLIS Unit Linz, the LIT AI Lab, the Institute for Machine Learning, are supported by the Federal State Upper Austria. IARAI is supported by Here Technologies. We thank the projects AI-MOTION (No. LIT-2018-6-YOU-212), AI-SNN (No. LIT-2018-6-YOU-214), DeepFlood (No. LIT-2019-8-YOU-213), Medical Cognitive Computing Center (MC3), PRIMAL (No. FFG-873979), S3AI (No. FFG-872172), DL for granular flow (No. FFG-871302), ELISE (No. H2020-ICT-2019-3, ID: 951847), AIDD (No. MSCA-ITN-2020, ID: 956832). We thank Janssen Pharmaceutica, Audi.JKU Deep Learning Center, TGW LOGISTICS GROUP GMBH, Silicon Austria Laboratories (SAL), FILL Gesellschaft mbH, Anyline GmbH, Google, ZF Friedrichshafen AG, Robert Bosch GmbH, UCB Biopharma SRL, Merck Healthcare KGaA, Software Competence Center Hagenberg GmbH, TÜV Austria, and the NVIDIA Corporation.

■ REFERENCES

(1) Lombardino, J. G.; Lowe, J. A. The Role of the Medicinal Chemist in Drug Discovery — Then and Now. *Nat. Rev. Drug Discovery* **2004**, *3*, 853−862.

(2) Lu, Z.; Chen, X.; Liu, X.; Lin, D.; Wu, Y.; Zhang, Y.; Wang, H.; Jiang, S.; Li, H.; Wang, X.; Lu, Z. Interpretable machine-learning strategy for soft-magnetic property and thermal stability in Fe-based metallic glasses. *npj Comput. Mater.* **2020**, *6*, 1−9.

(3) Mayr, A.; Klambauer, G.; Unterthiner, T.; Steijaert, M.; Wegner, J. K.; Ceulemans, H.; Clevert, D.-A.; Hochreiter, S. Large-scale

comparison of machine learning methods for drug target prediction on ChEMBL. *Chem. Sci.* **2018**, *9*, 5441−5451.
(4) McCammon, J. A. Computer-Aided Molecular Design. *Science* **1987**, *238*, 486−491.
(5) Ng, L. Y.; Chong, F. K.; Chemmangattuvalappil, N. G. Challenges and Opportunities in Computer-Aided Molecular Design. *Comput. Chem. Eng.* **2015**, *81*, 115−129.
(6) Corey, E. J.; Wipke, W. T. Computer-Assisted Design of Complex Organic Syntheses. *Science* **1969**, *166*, 178−192.
(7) Strieth-Kalthoff, F.; Sandfort, F.; Segler, M. H.; Glorius, F. Machine learning the ropes: principles, applications and directions in synthetic chemistry. *Chem. Soc. Rev.* **2020**, *49*, 6154−6168.
(8) Szymkuc, S.; Gajewska, E. P.; Klucznik, T.; Molga, K.; Dittwald, P.; Startek, M.; Bajczyk, M.; Grzybowski, B. A. Computer-Assisted Synthetic Planning: The End of the Beginning. *Angew. Chem., Int. Ed.* **2016**, *55*, 5904−5937.
(9) Segler, M. H. S.; Preuss, M.; Waller, M. P. Planning Chemical Syntheses with Deep Neural Networks and Symbolic AI. *Nature* **2018**, *555*, 604−610.
(10) Struble, T. J.; Alvarez, J. C.; Brown, S. P.; Chytil, M.; Cisar, J.; DesJarlais, R. L.; Engkvist, O.; Frank, S. A.; Greve, D. R.; Griffin, D. J.; Hou, X.; Johannes, J. W.; Kreatsoulas, C.; Lahue, B.; Mathea, Miriam; Mogk, G.; Nicolaou, C. A.; Palmer, A. D.; Price, D. J.; Robinson, R. I.; Salentin, S.; Xing, L.; Jaakkola, T.; Green, W. H.; Barzilay, R.; Coley, C. W.; Jensen, K. F. Current and Future Roles of Artificial Intelligence in Medicinal Chemistry Synthesis. *J. Med. Chem.* **2020**, *63*, 8667−8682.
(11) Segler, M. H. S.; Waller, M. P. Neural-Symbolic Machine Learning for Retrosynthesis and Reaction Prediction. *Chemistry* **2017**, *23*, 5966−5971.
(12) Coley, C. W.; Green, W. H.; Jensen, K. F. Machine Learning in Computer-Aided Synthesis Planning. *Acc. Chem. Res.* **2018**, *51*, 1281−1289.
(13) Coley, C. W.; Rogers, L.; Green, W. H.; Jensen, K. F. Computer-Assisted Retrosynthesis Based on Molecular Similarity. *ACS Cent. Sci.* **2017**, *3*, 1237−1245.
(14) Schwaller, P.; Laino, T.; Gaudin, T.; Bolgar, P.; Hunter, C. A.; Bekas, C.; Lee, A. A. Molecular Transformer: A Model for Uncertainty-Calibrated Chemical Reaction Prediction. *ACS Cent. Sci.* **2019**, *5*, 1572−1583.
(15) Klambauer, G.; Hochreiter, S.; Rarey, M. Machine Learning in Drug Discovery. *J. Chem. Inf. Model.* **2019**, *59*, 945.
(16) Nam, J.; Kim, J. Linking the neural machine translation and the prediction of organic chemistry reactions. *arXiv (Machine Learning)*, **2016**. 1612.09529.
(17) Schwaller, P.; Gaudin, T.; Lányi, D.; Bekas, C.; Laino, T. Found in Translation": Predicting Outcomes of Complex Organic Chemistry Reactions Using Neural Sequence-to-Sequence Models. *Chem. Sci.* **2018**, *9*, 6091−6098.
(18) Liu, B.; Ramsundar, B.; Kawthekar, P.; Shi, J.; Gomes, J.; Luu Nguyen, Q.; Ho, S.; Sloane, J.; Wender, P.; Pande, V. Retrosynthetic Reaction Prediction Using Neural Sequence-to-Sequence Models. *ACS Cent. Sci.* **2017**, *3*, 1103−1113.
(19) Karpov, P.; Godin, G.; Tetko, I. V. A transformer model for retrosynthesis. *Int. Conf. on Artif. Neur. Netw.* **2019**, *11731*, 817−830.
(20) Tetko, I. V.; Karpov, P.; Van Deursen, R.; Godin, G. State-of-the-Art Augmented NLP Transformer Models for Direct and Single-Step Retrosynthesis. *Nat. Commun.* **2020**, *11*, 5575.
(21) Weininger, D. SMILES, a Chemical Language and Information System. *J. Chem. Inf. Model.* **1988**, *28*, 31−36.
(22) Vaswani, A.; Shazeer, N.; Parmar, N.; Uszkoreit, J.; Jones, L.; Gomez, A. N.; Kaiser, Ł.; Polosukhin, I. Attention is all you need. *Adv. Neural Inf. Process. Syst.* **2017**, 5998−6008.
(23) Sutskever, I.; Vinyals, O.; Le, Q. V. Sequence to sequence learning with neural networks. *Adv. Neural Inf. Process. Syst.* **2014**, 3104−3112.
(24) Shi, C.; Xu, M.; Guo, H.; Zhang, M.; Tang, J. A graph to graphs framework for retrosynthesis prediction. *Int. Conf. Mach. Learn.* **2020**, 8818−8827.
(25) Somnath, V. R.; Bunne, C.; Coley, C. W.; Krause, A.; Barzilay, R. Learning graph models for template-free retrosynthesis. *arXiv (Machine Learning)*, **2020**. 2006.07038.
(26) Fortunato, M. E.; Coley, C. W.; Barnes, B. C.; Jensen, K. F. Data Augmentation and Pretraining for Template-Based Retrosynthetic Prediction in Computer-Aided Synthesis Planning. *J. Chem. Inf. Model.* **2020**, *60*, 3398−3407.
(27) Wei, J. N.; Duvenaud, D.; Aspuru-Guzik, A. Neural Networks for the Prediction of Organic Chemistry Reactions. *ACS Cent. Sci.* **2016**, *2*, 725−732.
(28) Bjerrum, E. J.; Thakkar, A.; Engkvist, O. Artificial Applicability Labels for Improving Policies in Retrosynthesis Prediction. *ChemRxiv*, **2020**. DOI: 10.26434/chemrxiv.12249458.v1.
(29) Dai, H.; Li, C.; Coley, C.; Dai, B.; Song, L. Retrosynthesis Prediction with Conditional Graph Logic Network. *Adv. Neural Inf. Process. Syst.* **2019**, *32*, 8872−8882.
(30) Baylon, J. L.; Cilfone, N. A.; Gulcher, J. R.; Chittenden, T. W. Enhancing Retrosynthetic Reaction Prediction with Deep Learning Using Multiscale Reaction Classification. *J. Chem. Inf. Model.* **2019**, *59*, 673.
(31) Coley, C. W.; Rogers, L.; Green, W. H.; Jensen, K. F. Computer-Assisted Retrosynthesis Based on Molecular Similarity. *ACS Cent. Sci.* **2017**, *3*, 1237−1245.
(32) Ishida, S.; Terayama, K.; Kojima, R.; Takasu, K.; Okuno, Y. Prediction and Interpretable Visualization of Retrosynthetic Reactions Using Graph Convolutional Networks. *J. Chem. Inf. Model.* **2019**, *59*, 5026−5033.
(33) Ramsauer, H.; Schäfl, B.; Lehner, J.; Seidl, P.; Widrich, M.; Gruber, L.; Holzleitner, M.; Adler, T.; Kreil, D.; Kopp, M. K.; Klambauer, G.; Brandstetter, J.; Hochreiter, S. Hopfield Networks is All You Need. *Int. Conf. Learn. Rep.*, **2021**. https://openreview.net/forum?id=tL89RnzIiCd.
(34) Widrich, M.; Schäfl, B.; Pavlovic, M.; Ramsauer, H.; Gruber, L.; Holzleitner, M.; Brandstetter, J.; Sandve, G. K.; Greiff, V.; Hochreiter, S.; Klambauer, G. Modern Hopfield Networks and Attention for Immune Repertoire Classification. *Adv. Neural Inf. Process. Syst.* **2020**, *33*, 18832−18845.
(35) Chen, T.; Kornblith, S.; Norouzi, M.; Hinton, G. A simple framework for contrastive learning of visual representations. *Int. Conf. Mach. Learn.* **2020**, 1597−1607.
(36) Radford, A.; Kim, J. W.; Hallacy, C.; Ramesh, A.; Goh, G.; Agarwal, S.; Sastry, G.; Askell, A.; Mishkin, P.; Clark, J.; Krueger, G.; Sutskever, I. Learning transferable visual models from natural language supervision. *arXiv (Machine Learning)*, **2021**. 2103.00020.
(37) Hasic, H.; Ishida, T. Single-Step Retrosynthesis Prediction Based on the Identification of Potential Disconnection Sites Using Molecular Substructure Fingerprints. *J. Chem. Inf. Model.* **2021**, *61*, 641−652.
(38) Segler, M. H. World programs for model-based learning and planning in compositional state and action spaces. *arXiv (Machine Learning)*, **2019**. 1912.13007.
(39) Hopfield, J. J. Neural networks and physical systems with emergent collective computational abilities. *Proc. Natl. Acad. Sci. U. S. A.* **1982**, *79*, 2554−2558.
(40) Graves, A.; Wayne, G.; Danihelka, I. Neural Turing Machines. *arXiv (Machine Learning)*, **2014**. 1410.5401.
(41) Chen, H. H.; Lee, Y. C.; Sun, G. Z.; Lee, H. Y.; Maxwell, T.; Giles, C. L. High order correlation model for associative memory. *AIP Conf. Proc.* **1986**, *151*, 86−99.
(42) Psaltis, D.; Park, C. H. Nonlinear discriminant functions and associative memories. *AIP Conf. Proc.* **1986**, *151*, 370−375.
(43) Baldi, P.; Venkatesh, S. S. Number of stable points for spin-glasses and neural networks of higher orders. *Phys. Rev. Lett.* **1987**, *58*, 913−916.
(44) Gardner, E. Multiconnected neural network models. *J. Phys. A* **1987**, *20*, 3453−3464.
(45) Abbott, L. F.; Arian, Y. Storage capacity of generalized networks. *Phys. Rev. A* **1987**, *36*, 5091−5094.

(46) Horn, D.; Usher, M. Capacities of multiconnected memory models. *J. Phys. (Paris)* **1988**, *49*, 389−395.
(47) Caputo, B.; Niemann, H. Storage Capacity of Kernel Associative Memories. *Proceedings of the International Conference on Artificial Neural Networks (ICANN)*. Berlin, Heidelberg, 2002; p 51−56.
(48) Krotov, D.; Hopfield, J. J. Dense associative memory for pattern recognition. *Adv. Neural Inf. Process. Syst.* **2016**, *29*, 1172−1180.
(49) Ramsauer, H.; Schäfl, B.; Lehner, J.; Seidl, P.; Widrich, M.; Adler, T.; Gruber, L.; Holzleitner, M.; Pavlovic, M.; Sandve, G. K.; Greiff, V.; Kreil, D.; Kopp, M.; Klambauer, G.; Brandstetter, J.; Hochreiter, S. Hopfield networks is all you need. *arXiv (Machine Learning)*, **2020**. 2008.02217.
(50) Rogers, D.; Hahn, M. Extended-Connectivity Fingerprints. *J. Chem. Inf. Model.* **2010**, *50*, 742−754.
(51) Ba, J. L.; Kiros, J. R.; Hinton, G. E. Layer normalization. *arXiv (Machine Learning)*, **2016**. 1607.06450.
(52) Gilmer, J.; Schoenholz, S. S.; Riley, P. F.; Vinyals, O.; Dahl, G. E. Neural message passing for quantum chemistry. *Int. Conf. Mach. Learn.* **2017**, 1263−1272.
(53) Mayr, A.; Klambauer, G.; Unterthiner, T.; Steijaert, M.; Wegner, J. K.; Ceulemans, H.; Clevert, D.-A.; Hochreiter, S. Large-Scale Comparison of Machine Learning Methods for Drug Target Prediction on ChEMBL. *Chem. Sci.* **2018**, *9*, 5441.
(54) Loshchilov, I.; Hutter, F. Decoupled weight decay regularization. *arXiv (Machine Learning)*, **2017**. 1711.05101.
(55) Landrum, G. *RDKit: Open-Source Cheminformatics*. 2006, accessed on Jan. 1, 2020.
(56) Zhang, Y.; Jiang, H.; Miura, Y.; Manning, C. D.; Langlotz, C. P. Contrastive learning of medical visual representations from paired images and text. *arXiv (Machine Learning)*, **2020**. 2010.00747.
(57) Hadsell, R.; Chopra, S.; LeCun, Y. Dimensionality reduction by learning an invariant mapping. *Proc. IEEE* **2006**, *2*, 1735−1742.
(58) Wu, Z.; Xiong, Y.; Yu, S. X.; Lin, D. Unsupervised Feature Learning via Non-parametric Instance Discrimination. *Proc. IEEE* **2018**, 3733−3742.
(59) Misra, I.; van der Maaten, L. Self-supervised learning of pretext-invariant representations. In *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, Seattle, WA, 2020; pp 6707−6717, .
(60) He, K.; Fan, H.; Wu, Y.; Xie, S.; Girshick, R. Momentum contrast for unsupervised visual representation learning. In *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, Seattle, WA2020; pp 9726−9735, .
(61) Tian, Y.; Krishnan, D.; Isola, P. Contrastive multiview coding. *Comp. Vision ECCV* **2020**, *12356*, 776−794.
(62) Lowe, D. M. Extraction of chemical structures and reactions from the literature. Ph.D. Thesis, University of Cambridge, 2012.
(63) Coley, C. W.; Green, W. H.; Jensen, K. F. RDChiral: An RDKit Wrapper for Handling Stereochemistry in Retrosynthetic Template Extraction and Application. *J. Chem. Inf. Model.* **2019**, *59*, 2529−2537.
(64) Schneider, N.; Stiefl, N.; Landrum, G. A. What's What: The (Nearly) Definitive Guide to Reaction Role Assignment. *J. Chem. Inf. Model.* **2016**, *56*, 2336−2346.
(65) Coley, C. W.; Jin, W.; Rogers, L.; Jamison, T. F.; Jaakkola, T. S.; Green, W. H.; Barzilay, R.; Jensen, K. F. A Graph-Convolutional Neural Network Model for the Prediction of Chemical Reactivity. *Chem. Sci.* **2019**, *10*, 370−377.
(66) Sun, R.; Dai, H.; Li, L.; Kearnes, S.; Dai, B. Energy-based View of Retrosynthesis. *arXiv (Machine Learning)*, **2020**. 2007.13437.
(67) Wang, X.; Li, Y.; Qiu, J.; Chen, G.; Liu, H.; Liao, B.; Hsieh, C.-Y.; Yao, X. RetroPrime: A Diverse, Plausible and Transformer-Based Method for Single-Step Retrosynthesis Predictions. *Chem. Eng. J.* **2021**, *420*, 129845.
(68) Sacha, M.; Błaz, M.; Byrski, P.; Dabrowski-Tumanski, P.; Chrominski, M.; Loska, R.; Włodarczyk-Pruszynski, P.; Jastrzebski, S. Molecule edit graph attention network: modeling chemical reactions as sequences of graph edits. *J. Chem. Inf. Model.* **2021**, *61*, 3273−3284.
(69) Mao, K.; Zhao, P.; Xu, T.; Rong, Y.; Xiao, X.; Huang, J. Molecular Graph Enhanced Transformer for Retrosynthesis Prediction. *bioRxiv*, **2020**. DOI: 10.1101/2020.03.05.979773.
(70) Mann, V.; Venkatasubramanian, V. Retrosynthesis Prediction Using Grammar-Based Neural Machine Translation: An Information-Theoretic Approach. *ChemRxiv*, **2021**. DOI: 10.26434/chemrxiv.14410442.v1.
(71) Zheng, S.; Rao, J.; Zhang, Z.; Xu, J.; Yang, Y. Predicting Retrosynthetic Reactions Using Self-Corrected Transformer Neural Networks. *J. Chem. Inf. Model.* **2020**, *60*, 47−55.
(72) Chen, B.; Shen, T.; Jaakkola, T. S.; Barzilay, R. Learning to Make Generalizable and Diverse Predictions for Retrosynthesis. *arXiv (Machine Learning)*, **2019**. 1910.09688.
(73) Yan, C.; Ding, Q.; Zhao, P.; Zheng, S.; Yang, J.; Yu, Y.; Huang, J. RetroXpert: Decompose Retrosynthesis Prediction like a Chemist. *arXiv (Machine Learning)*, **2020**. 2011.02893.
(74) Lee, H.; Ahn, S.; Seo, S.-W.; Song, Y. Y.; Hwang, S.-J.; Yang, E.; Shin, J. RetCL: A Selection-Based Approach for Retrosynthesis via Contrastive Learning. *arXiv (Machine Learning)*, **2021**. 2105.00795.
(75) Guo, Z.; Wu, S.; Ohno, M.; Yoshida, R. A Bayesian Algorithm for Retrosynthesis. *J. Chem. Inf. Model.* **2020**, *60*, 4474−4486.
(76) Ishiguro, K.; Ujihara, K.; Sawada, R.; Akita, H.; Kotera, M. Data Transfer Approaches to Improve Seq-to-Seq Retrosynthesis. *arXiv (Machine Learning)*, **2020**. 2010.00792.
(77) Ucak, U. V.; Kang, T.; Ko, J.; Lee, J. Substructure-based neural machine translation for retrosynthetic prediction. *J. Cheminf.* **2021**, *13*, 4.
(78) Probst, D.; Reymond, J.-L. A probabilistic molecular fingerprint for big data settings. *J. Cheminf.* **2018**, *10*, 1−12.
(79) Chen, B.; Li, C.; Dai, H.; Song, L. Retro*: learning retrosynthetic planning with neural guided A* search. *Int. Conf. Mach. Learn.* **2020**, 1608−1616.
(80) Paszke, A.; Gross, S.; Massa, F.; Lerer, A.; Bradbury, J.; Chanan, G.; Killeen, T.; Lin, Z.; Gimelshein, N.; Antiga, L.; Desmaison, A.; Köpf, A.; Yang, E.; DeVito, Z.; Raison, M.; Tejani, A.; Chilamkurthy, S.; Steiner, B.; Fang, L.; Bai, J.; Chintala, S. Pytorch: An imperative style, high-performance deep learning library. *Adv. Neural Inf. Process. Syst.* **2019**, *32*, 8026−8037.
(81) Kunaver, M.; Pozrl, T. Diversity in Recommender Systems, A Survey. *Knowl.-Based Syst.* **2017**, *123*, 154−162.
(82) Isufi, E.; Pocchiari, M.; Hanjalic, A. Accuracy-diversity trade-off in recommender systems via graph convolutions. *Inf. Process. Manage.* **2021**, *58*, 102459.
(83) Bradshaw, J.; Paige, B.; Kusner, M. J.; Segler, M.; Hernández-Lobato, J. M. A Model to Search for Synthesizable Molecules. *Adv. Neural Inf. Process. Syst.* **2019**, *32*, 7937−7949.
(84) Gottipati, S. K.; Sattarov, B.; Niu, S.; Pathak, Y.; Wei, H.; Liu, S.; Thomas, K. M. J.; Blackburn, S.; Coley, C. W.; Tang, J.; Chandar, S.; Bengio, Y. Learning To Navigate The Synthetically Accessible Chemical Space Using Reinforcement Learning. *arXiv (Machine Learning)*, **2020**. 2004.12485.
(85) Horwood, J.; Noutahi, E. Molecular Design in Synthetically Accessible Chemical Space via Deep Reinforcement Learning. *ACS Omega* **2020**, *5*, 32984−32994.
(86) Renz, P.; Van Rompaey, D.; Wegner, J. K.; Hochreiter, S.; Klambauer, G. On Failure Modes in Molecule Generation and Optimization. *Drug Discovery Today: Technol.* **2019**, *32*−*33*, 55−63.
(87) Bradshaw, J.; Paige, B.; Kusner, M. J.; Segler, M. H.; Hernández-Lobato, J. M. Barking up the right tree: an approach to search over molecule synthesis dags. *arXiv Preprint* **2020**, arXiv:2012.11522.

Supporting Information for Improving few- and zero-shot reaction template prediction using modern Hopfield networks

Philipp Seidl,[†] Philipp Renz,[†] Natalia Dyubankova,[‡] Paulo Neves,[‡] Jonas Verhoeven,[‡] Jörg K. Wegner,[¶] Marwin Segler,[§] Sepp Hochreiter,[†,‖] and Günter Klambauer[*,†]

[†]*Johannes Kepler University Linz, ELLIS Unit Linz, LIT AI Lab, Institute for Machine Learning, Altenbergerstraße 69, Linz, AT 4040*

[‡]*Janssen Pharmaceutica NV, High Dimensional Biology and Discovery Data Sciences, Janssen Research & Development, Turnhoutseweg 30, Beerse, BE 2340*

[¶]*Janssen Research Development, LLC, In-Silico Discovery and External Innovation (ISDEI), 1 Cambridge Center, 255 Main St, Cambridge, MA 02142, USA*

[§]*Microsoft Research, 21 Station Road, Cambridge, UK CB1 2FB*

[‖]*Institute of Advanced Research in Artificial Intelligence, Landstraßer Hauptstraße 5, Wien, AT 1030*

E-mail: klambauer@ml.jku.at

This document provides more related work (see Section S2), then details on the experiments (see Section S3), additional results (see Section S4), visualization (see Section S5), comparison of different methods for selected examples for single-step retrosynthesis (see Section S6)), as well as an alternative view on the loss and an extended formulation of the algorithm as pseudo-code (see Section S7).

Contents

S1 Notation

Table S1: Symbols and notations used in this paper.

Definition	Symbol/Notation	Dimension
set of reaction templates	$\boldsymbol{T}$	set of size K
encoded set of reaction template	$\boldsymbol{T}_h$	$d_t \times K$
reactant molecules	$\boldsymbol{r}$	
product molecule	$\boldsymbol{m}$	
encoded molecule	$\boldsymbol{m}_h$	d_m
reaction template	$\boldsymbol{t}$ or $\boldsymbol{t}_k$	
training set pair	$(\boldsymbol{m}, \boldsymbol{t})$	
state pattern	$\boldsymbol{\xi}$	d
stored pattern	$\boldsymbol{x}_k$	d
stored pattern matrix	$\boldsymbol{X}$	$d \times K$
associations	$\boldsymbol{p}$	K
update function of modern Hopfield network (MHN)	f	
molecule encoder	$\boldsymbol{h}^m$	
reaction template encoder	$\boldsymbol{h}^t$	
model function	g	
network parameters of $\boldsymbol{h}^m$	$\boldsymbol{w}$	
network parameters of $\boldsymbol{h}^t$	$\boldsymbol{v}$	
parameters of Hopfield layer $\boldsymbol{h}^m$	$\boldsymbol{W}_m, \boldsymbol{W}_t$	$d \times$ undef
association activation function	ϕ	
number of templates	K	
dimension of association space	d	

S2 Further related work

Here we provide a broader view on works that have addressed common issues with template relevance prediction. In prior work[1–6], template relevance prediction is often viewed as a multi-class classification task, where, given a product, an machine learning (ML) model is trained to predict which of the templates extracted from a reaction database are most relevant. Automatic extraction of templates leads to many rare templates, which poses a problem in the classification task as it leads to many classes with few training samples[3,5]. In earlier work[5], rare templates were excluded from training. Baylon et al.[1] proposed a hierarchical grouping

of reaction templates and trained a separate NN for each group of templates. Fortunato et al.[3] pre-trained their template scoring model to predict which templates are applicable and observed that it improves template-relevance predictions, especially for rare templates.[2] used a graph-NN to predict the relevant templates. Bjerrum et al.[6] trained two separate NNs. The first NN is trained on the applicability matrix and serves for pre-filtering reaction templates. The second is trained on the reaction dataset and ranks the reaction templates according to their relevance. Dai et al.[7] make use of the template structures. However, they factorize the predicted probabilities into multiple functions, which might not be suited to the problem. The in-scope filter of the computer-assisted synthesis planning (CASP) system by Segler et al.[5] also make use of template structure. Sun et al.[8] apply all templates and uses a model to rank the resulting reactants, which achieves the best top-1 accuracy but is computationally costly (Dual-TB in Table 2).

S3 Details on Experiments

S3.1 Template relevance prediction

S3.1.1 Datasets and preprocessing

For preprocessing USPTO-sm and USPTO-lg, we followed the implementation of Fortunato et al.[3]. The templates were extracted from the mapped reactions using RDChiral[9] and subsequently filtered according to symmetry, validity, and by checking if the application of the template yielded the result as in the reaction the template originated from. Despite adhering to the original implementation by the authors, our preprocessing resulted in different dataset sizes. The roughly 2 million starting reactions decreased to 443,763 samples and 236,053 reaction templates for USPTO-lg (compared to 669,683 samples and 186,822 reaction templates in Fortunato et al.[3]), and to 40,257 samples and 9,162 reaction templates for USPTO-sm (compared to 32,099 samples and 7,765 reaction templates in Fortunato et al.[3]).

It can be seen that 17% of samples occur in a class that has only one sample in USPTO-sm and 43% in USPTO-lg. 66% of reaction templates in USPTO-sm and 80% in USPTO-lg occur only in a single reaction.

To allow for pre-training of the methods, we calculated the applicability matrix, i.e., which templates in a dataset are applicable to which molecules. Fortunato et al.[3] reported that this would take ∼36 CPU-hours for USPTO-sm and ∼330 CPU-days for USPTO-lg on a single core of a Xeon(R) Gold 6154 CPU@3 GHz. We found that using a substructure screen could speed up this procedure. The following values were obtained using an AMD EPYC 7542. For USPTO-sm it only takes us 3.3 CPU-minutes to achieve the same result. Using 16 CPU-cores this reduces to $\sim 14s$. Using 32 CPU-cores applicability calculation time for USPTO-lg reduces to $\sim 50m$ which corresponds to 27 CPU-hours compared to ∼8000 CPU-hours for the original implementation. These comparisons should be taken with a grain of salt because of the slightly different dataset sizes and hardware used. For USPTO-lg $443\,763 \cdot 236\,053 \sim 10^{11}$ pairs have to be checked, and our code relies on a python loop. Using a compiled language could probably further speed up the procedure.

S3.1.2 Data splits

We split the data into a training, validation, and test set following Fortunato et al.[3]. Here, a stratified split was used to ensure that templates are more equally represented across the splits. Concretely, in Fortunato et al.[3], the split proportions were 80/10/10% except for templates with fewer than 10 samples, where one random sample was put into the test set, one into the validation set, and the rest into the training set. If only two samples were present, one was put into the test and one into the training set. Finally, if only a single sample was available for a template, it was randomly placed into the train/validation/test set with an 80/10/10% chance.

S3.1.3 Feature extraction

Molecule fingerprints. The source molecules are represented as simplified molecular-input line-entry system (SMILES). We extract a fingerprint representation of the molecules. A molecular fingerprint represents the absence and presence of substructures within a molecule. The way those subgraphs or substructures are selected is very much different and depends on the chosen method as well as their parameters. For example, when using morgan fingerprints [10], the algorithm iterates through each atom in the graph, and notes all substructure up to a certain radius. All substructures are hashed to each map to an integer. All integers are folded (modulo operation) to a fixed length. The longer the less collisions will appear. The final vector is a rather efficient indication of how many substructures two molecules share, and therefore one way to measure their similarity.

We tried out different fingerprint types, e.g. folded Morgan fingerprints with chirality [10] and the hyperparameter selection procedure (see Table S2) selected Morgan fingerprint with radius of 2 folded to 4096 features.

Template fingerprints. For the template representation, a similar procedure has been applied. A template consists of multiple enumerated SMILES arbitrary target specification (SMARTS)-strings. The fingerprint type for the templates was set to `rdk`-fingerprint or `pattern`-fingerprint for template relevance prediction and calculated for each molecule that the pattern represents. We experimented with multiple ways of combining not just the product side, but also the reactant side to this representation. We found the following to perform best among the considered variants. The fingerprints were calculated for each molecular pattern, and a disjunction over reactants as well as products was calculated. The product minus half of the reactant side results in the template fingerprint as input for the template encoder. We also tried the `structuralFingerprintForReaction` function provided by RDKit [11], which concatenates the disjunction of each side of the reaction, but found the weighted combination to perform better.

Consider a template consisting of reactant substructures s_r and product substructure

s_p. Using a fingerprint-function f_p produces a fingerprint-vector $x_{r_1} \dots x_{r_n}$ and x_p. The final template fingerprint x_t is computed using the following form:

$$x_t = f_p(s_p) - \cdot 0.5\text{pool}(f_p(s_{r_1}), ..., f_p(s_{r_n}))$$

where pool is a pooling function. A few things need to be considered for the implementation as can be found in the linked github repository. To compute the fingerprints for each of the substructures the function `getFingerprint` from `molutils` is used and `issmarts` is set to True. By default the fingerprint-size, -type, or -radius (if applicable) is defined by the hyperparameters. Sanitization is an important step and done by default. Here the `SanitizeMol` function from RDKit[11] is employed setting `catchErrors` parameter to True. The next step is using the `FastFindRings` function to providing ring information if applicable. Further `UpdatePropertyCache(strict=False)` to correcting valence information is used. Max-pooling is used as pooling operation. Another thing to note is that, e.g. number of explicit hydrogens, direct bonds or charge, may be present in a SMARTS-template, but will not be represented by the fingerprint in RDKit. Nevertheless, the resulting representation allows for a learnable measure of similarity between templates, but for some templates the representation turned out to be equal, and therefore indistinguishable. A simple, yet effective approach is to add an additional template embedding, which adds more flexibility and has less inductive bias.

The randomly initialized embedding was added to the representation of frequent templates in order to help to differentiate frequent templates with a high fingerprint similarity. Templates are classified as frequent if they appear at least a certain number of times in the training-set, which is determined by the hyperparameter `random template threshold` (see Table S2).

S3.1.4 Training

All models were trained for a maximum of 100 epochs on a Titan V with 12 GB RAM or a P40 with 24 GB RAM using PyTorch 1.6.0[12]. In the case of deep neural network (DNN), only the molecule encoder was trained, and a linear layer, projecting from the last hidden layer to the number of templates was added. For pre-training on the applicability task we changed the loss function to the mean of binary cross-entropy for each output (template). We also experimented with Information Noise-Contrastive Estimation (InfoNCE)-loss[13] on representations in Hopfield space (see S7). Because of fast convergence and slightly better performance, and because for the United States patent and trademark office (USPTO) data sets only a single template is correct for each molecule, we use our proposed loss, which in this case is equivalent to CE-loss.

S3.1.5 Hyperparameter selection and model architecture

Hyperparameters were explored via automatic Bayesian optimization for USPTO-sm, as well as manual hyperparameter-tuning. In the former, early stopping was employed. The range of values was selected based on prior knowledge. Additional manual hyperparameter-tuning resulted in better predictive performance on the validation set. Some of the important hyperparameters are the beta scaling factor of the Hopfield layer β, the dimension of the association space d, as well as the association-activation function, or if the association space should be normalized via layer-norm[14]. An overview of considered and selected hyperparameters is given in Tab. S2. All models were trained if applicable for a maximum of 100 epochs using AdamW[15] (`betas`=(0.9, 0.999), `eps`=1e−8, `weight_decay`=1e−2, `amsgrad`=False). Hyperparameters were selected based on the minimal CE-loss on the validation set.

Table S2: Hyperparameter search-space for template relevance prediction. The rows are subdivided into five modules: overall parameters, the molecule encoder, the template encoder, the Hopfield layer, and setting specific hyperparameters that were only used if explicitly stated. The column values show the range of the explored parameters. If multiple Hopfield layers were used, the same hyperparameters were used for all layers. A `random template threshold` of -1 corresponds to not adding noise. The fingerprint size for the molecule encoder was always the same as the template encoder. The pre-training learning rate was also defined by the learning rate, the optimizer remained the same, but the loss-function changed to binary-cross-entropy loss.

Hyperparam	**Values**	**MHN selected (Sm/Lg)**	**DNN selected (Sm/Lg)**
learning rates	{1e-4, 2e-4, 5e-4, 1e-3}	5e-4 / 1e-4	5e-4 / 2e-4
batch-size	{32, 128, 256, 1024}	1024	256 / 1024
dropout	[0.0, 0.6]	0.2	0.15
molecule encoder			
fingerprint type	{morgan, rdk}	morgan	morgan
fingerprint size	{1024, 2048, 4096}	4096	4096
number of layers	{0, 1, 2}	0	1
layer-dimension	{1024, 2048, 4096}	-	2048
activation-function	{None, SELU, ReLU}	None	ReLU
template encoder			
number of layers	{0, 1, 2}	0	
template fingerprint type	{pattern, rdk}	rdk	
random template thresh.	-1, 2, 5, 10, 50	2	
Hopfield layer			
beta	[0.01, 0.3]	0.03	
association af	{None, SELU, GeLU, Tanh}	None / Tanh	
normalize pattern	{False, True}	False	
normalize projection	{False, True}	True	
learnable stored-pattern	{False, True}	False	
hopf-num-layers	{1, 2, 3}	1	
hopf-num-Wm	{1, 2, 3}	1	
hopf-num-Wt	{1, 2, 3}	1	
hopf-FF-activation	{None, SELU, ReLU}	None	
association-dimension d	{32, 64, 512, 1024}	1024	
hopf-num-heads	{1, 6, 12}	1	
Setting-specific-hps			
pre-training epochs	{0,5,10,15,20,25}	10	25 / 5

S3.2 Single-step retrosynthesis

S3.2.1 Datasets and preprocessing

For single-step retrosynthesis, we used the preprocessed version and splitting-procedure from Coley et al.[16]. The dataset originated from USPTO-50k by Schneider et al.[17]. It is different in details from USPTO-sm and does not contain a filtering step, whereby samples are excluded if extracted and applied templates don't yield the reactants. A further difference is the split. For USPTO-50k, we shuffle the samples and further split it according to the procedure by Coley et al.[16], randomly splitting within the reaction types, to obtain 40008 train- 5001 validation- and 5007 test-samples (80/10/10). We computed the reaction templates only from the train- and validation-set.

We additionally report results for USPTO-full which was preprocessed and split randomly by Tetko et al.[18]. The dataset contains almost 1M samples (train: 761.335, valid: 95.181, test: 96.027), and extracting templates from the train and validation set lead to 257.340 templates. Note that there are less test samples as reported in Dai et al.[7] due to the different pre-processing and removal of duplicate reactions.

S3.2.2 Feature extraction

For this experiment, we additionally explored Mixed-Fingerprint (MxFP), which is a mixture of multiple folded, counted (where applicable) fingerprints: molecular access system (MACCS), Morgan, ErG, AtomPair, TopolocialTorsion, mini-hash fingerprint (MHFP)[19] and RDK. We additionally scale the counts by $\log(1+x)$[5].

Template-representation. We compute fingerprints for each subgraph-pattern in the reaction template. Again we use a mixture of multiple unfolded RDKit fingerprints. For pooling the reactant fingerprints, we additionally experimented with different pooling operations. The main idea is to avoid that different sets are identical after pooling and thus to increase the expressivity of the pooling operation. Lgamma pooling is a novel pooling operation that

uses the log of the gamma-function.

$$\mathrm{lgp}(\boldsymbol{x}) = \log\left(\Gamma\left(\sum_{i=0}^{n} x_i + 2\right)\right) - \sum_{i=0}^{n} \log(\Gamma(x_i + 1)), \tag{1}$$

where the $\boldsymbol{x}$ contains a single feature of the elements of the set that is pooled. The use of this pooling function provided a small performance increase over max-pooling.

S3.2.3 Hyperparameters and model architecture

Hyperparameters were tuned manually and selected based on top-1 accuracy on the validation set. The explored parameters, as well as the selected hyperparameters, can be found in Table S3. Models were also trained if applicable for a maximum of 100 epochs using AdamW [15] (betas=(0.9,0.999), eps=1e-8, weight_decay=1e-2, amsgrad=False). As input, MxFP was selected with a fingerprint size of 30k. It consists of two layers, where the input for the second layer is $\boldsymbol{\xi}^{\mathrm{new}} + \boldsymbol{\xi}$, a skip connection from the first layers input plus the output of the first layer. The first layers' memory is comprised of MxFP-template fingerprints with lgamma-pooled reactants (see Section S3.2.2). The second layer uses a different template representation: RDK-template fingerprint with additional random noise for all templates which appear more than once in the training set. The final prediction is computed by a weighted average of the individual layers' $\boldsymbol{p}$.

The NeuralSym baseline was trained as follows: As a model architecture, we used a feed-forward neural network with a single hidden layer of size 4096 and self-normalizing linear unit (SELU) activation function. [20] The inputs to this network were extended connectivity fingerprint (ECFP)-fingerprints [21] with radius 2 and size 4096. The model was trained using AdamW [15] with learning rate 1e-3 and weight-decay of 1e-3. The model was trained for 7 epochs with a batch size of 512.

Table S3: Hyperparameter search-space for single-step retrosynthesis. The layout and specifics are equivalent to Table S2 but differ in the explored values and architectural choices. The hyperparameters for the Hopfield layer remain the same among layers, with individually initialized weight parameters. The input for layer 1 is given by "template encoder 1" and vice versa for layer 2. The column MHN (50k/full) corresponds to the results of MHNreact and DNN (50k) to the first mention of Neuralsym in Table 2.

Hyperparam	**Values**	**MHN (50k/full)**	**DNN (50k)**
learning rates	{1e-4, 2e-4, 5e-4, 1e-3}	5e-4 / 1e-4	1e-4
batch-size	{32, 128, 256, 1024}	1024	256
dropout	[0.0, 0.6]	0.4/0.2	0.15
molecule encoder			
fingerprint type	{morgan, rdk, MxFP}	MxFP	morgan
fingerprint size	{4096, ..., 40e3}	3e4/2.4e4	4096
fingerprint radius	{2, ..., 6}	-	2
number of layers	{0, 1, 2}	0	1
layer-dimension	{1024, 2048, 4096}	-	4096
activation-function (af)	{None, SELU, ReLU}	None	SELU
template encoder 1			
number of layers	{0, 1, 2}	0	
template fingerprint type	{rdk, rdkc, MxFP}	MxFP	
random template threshold	-1, 2, 5, 10, 50	-1/2	
reactant pooling	{max, sum, mean, lgamma}	lgamma	
template encoder 2			
number of layers	{0, 1, 2}	0 / -	
template fingerprint type	{rdk, MxFP}	rdk/-	
random template threshold	-1, 2, 5, 10, 50	2/-	
Hopfield layer 1 and 2			
beta	[0.01, 0.3]	0.03	
association af	{None, Tanh}	None	
normalize input pattern	{False, True}	True	
normalize association proj.	{False, True}	True	
learnable stored-pattern	{False}	False	
hopf-num-layers	{1, 2}	2/1	
hopf-num-Wm	{1, 2}	1	
hopf-num-Wt	{1, 2}	1	
hopf-FF-af	{None, SELU, ReLU}	None	
association-dimension d	{64, ..., 2048}	1024/1500	
hopf-num-heads	{1, 6, 12}	1	

S3.2.4 Methods omitted from comparison

We omitted some studies from the comparison in Table 2, despite them reporting performance on USPTO-50k. We found that the experimental settings or reported metrics in these studies differed from ours. While these reported values are not per se flawed, we think that inclusion in the comparison may be misleading. We list specifics below:

- Yan et al.[22] reported (`https://github.com/uta-smile/RetroXpert`) that their model used information in the atom-mappings about where the reaction center is. This information relies on the knowledge of the reactants. As the reactants are to be predicted in this task this is considered test set leakage.
- The reported values in[23] are also based on unintentional use of information about the reaction center, similar to above[1].
- The method proposed in[24] selects reactants from a candidate set. Since this candidate set is a superset of the reactants in the USPTO-50k, it might contain information about the test data. Indeed we found that we could augment the performance of our method by a process of elimination, i.e., discarding reactant sets from the predictions if they are not in the candidate set.
- The method proposed in[25] also relies on a candidate set that we suspect to contain information about the test set. However, the description of the method is not very detailed.
- Guo et al.[26] use reactants from USPTO-stereo as described in[27] as a candidate set. We found that, given this set, we could augment the performance of our method by removing reactant sets not in this set from our predictions.
- Ishiguro et al.[28] propose a pre-training step on a larger data set which does not conform to the setting in most prior work and is therefor excluded.

[1]Personal communication with the authors

- Ishida et al.[2] use a different subset of USPTO-50k to train their model and report different metrics.
- Ucak et al.[29] also make use of a different subset and also do not report reactant top-k accuracy.
- Liu et al.[30] only provide results for the special case where the type of reaction is provided to the model.

S3.2.5 Inference speed

We investigated the speed/performance trade-off for multiple methods. Firstly, we trained a Transformer baseline using the code and settings provided by[27], except for setting the batch size to 8192, warm-up steps to 6k, and train steps to 50k. We evaluated the predictions of this model when run with beam sizes $\{1, 3, 5, 10, 20, 50, 75, 100\}$. While performance increases with larger beam size, the model also gets slower. This model outperforms the model suggested in[31], but could not reach the performance of[18]. Model training took about six hours on an Nvidia V100.

For MHN and NeuralSym, the inference procedure contains the following steps. First fingerprints for the given products have are generated. Then the model is used to predict template relevance. For each product templates are executed in the order of their score until a fixed number of reactant sets are obtained. To optimize top-k accuracy, it does not help to generate more than k reactant sets. Therefore we set the number of reactant sets to generate to $\{1, 3, 5, 10, 20, 50\}$ optimize speed without loss of top-k accuracy for the respective k and measured inference time. Speeds for the Transformer, NeuralSym, and MHN models have been measured using an Nvidia Tesla T4 and 16 cores of an AMD EPYC 7542. We also tested stopping template execution based on the cumulative probability of already executed ones as done in[5], however found that it did not improve upon stopping after a certain number of reactants have been retrieved.

S4 Additional Results

S4.1 Template relevance prediction

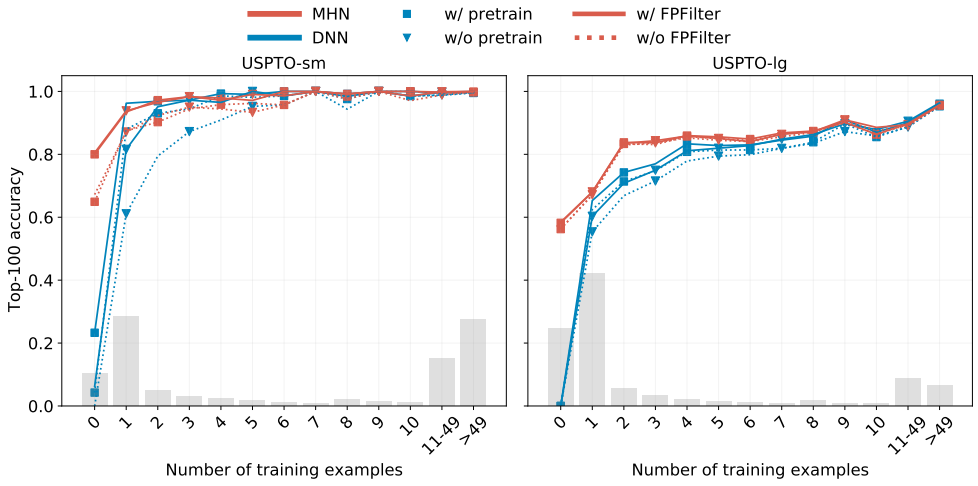


Figure S1: Results of methods with different design elements on the USPTO-sm and USPTO-lg datasets. Each method consists of a combination of the following elements: a) a network MHN or DNN (blue or red line), b) whether pre-training is applied (squares or triangles), and c) whether fingerprint filter (FPF) is applied for postprocessing (solid or dashed line). These eight possible combinations are displayed as lines with their top-100 accuracy on the y-axis and the different template frequency categories on the x-axis.

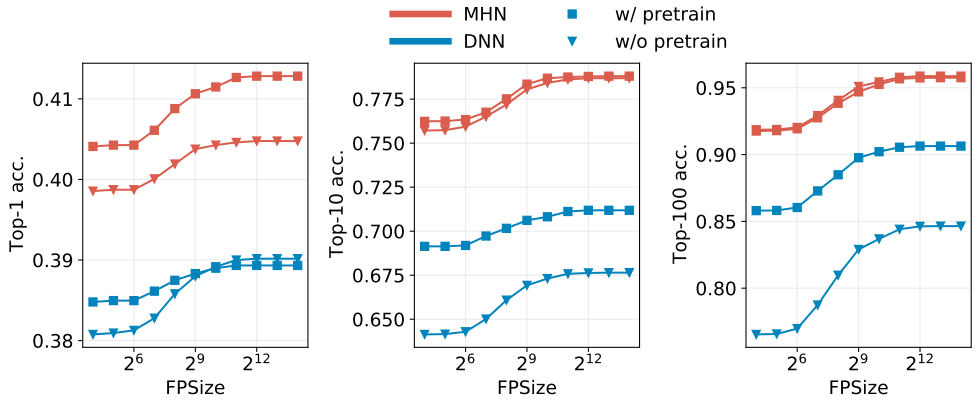


Figure S2: Predictive performance of different methods in dependency of the fingerprint size. Each method consists of a combination of the following elements: a) a network MHN or DNN (blue or red line), and b) whether pre-training is applied (squares or triangles). These four possible combinations are displayed as lines with their top-1, top-10 and top-100 accuracy. Performance saturates at a fingerprint size of about 2^{12}=4096, and we therefore choose this value for the other experiments.

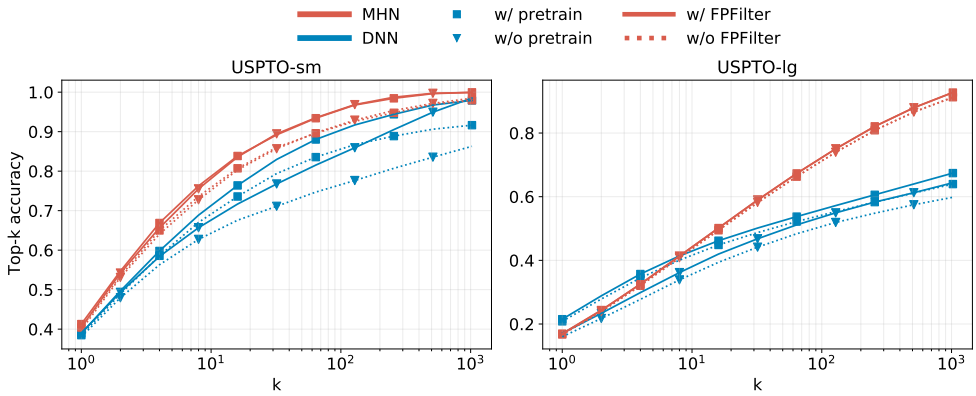


Figure S3: Comparison of methods with respect to their top-k accuracies. Each method consists of a combination of the following elements: a) a network MHN or DNN (blue or red line), b) whether pre-training is applied (squares or triangles), and c) whether FPF is applied for postprocessing (solid or dashed line). These eight possible combinations are displayed as lines with their k parameter on the x-axis and their top-k accuracy on the y-axis. MHNs provide the best top-k accuracy with k larger or equal 10.

S4.2 Single-step retrosynthesis

S4.2.1 USPTO-full

Table S4: Reactant top-k accuracy (%) on USPTO-full retrosynthesis. Bold values indicate values within 0.1, green 1 and yellow within 3 percentage points to the maximum value. Category ("Cat.") indicates whether a method is template-based (tb) or template-free (tf). Methods in the upper part evalute on a version of USPTO-full with pre-processing according to Tetko et al.[18], whereas the lower part has a larger test set (some duplicates), different pre-processing and split as described in Dai et al.[7].

Abbr.	Ref.	Cat.	Top-1	Top-2	Top-3	Top-5	Top-10	Top-20	Top-50
MHNreact	ours	tb	45.5	**57.2**	**62.5**	**67.4**	71.9	**74.8**	**77.1**
ATx100	[18]	tf	**46.2**	**57.2**			**73.3**		
RetroPrime	[32]	tf	44.1		59.1	62.8	68.5		
GLN	[7]	tb	39.3				63.7		
MEGAN	[33]	tf	33.6				63.9		74.1
Neuralsym	[4,7]	tb	35.8				60.8		
Retrosim	[16]	tb	32.8				56.1		

S4.2.2 USPTO-50k further splits

Table S5: Different split-scenarios for reactant top-k accuracy (%) on USPTO-50k retrosynthesis for MHNreact. Chronological split refers to a split where one trains on data <=2012, validations on 2013 and tests on samples >2013 on a proportion of approx. 78:7:15. Dates have been obtained where possible, through matching USPTO-50k reaction-id's with the original dataset[34] (For 19 samples the year could not be determined). Note that the hyperparameters have been determined on the random split, and not on each split individually.

Split	Top-1	Top-3	Top-5	Top-10	Top-20	Top-50
default random split	$51.8^{\pm.2}$	$74.6^{\pm.3}$	$81.2^{\pm.2}$	$88.1^{\pm.2}$	$92.0^{\pm.1}$	$94.0^{\pm.0}$
10-fold cross-validation	$50.7^{\pm.6}$	$73.2^{\pm.6}$	$80.7^{\pm.4}$	$87.2^{\pm.5}$	$91.1^{\pm.5}$	$93.5^{\pm.5}$
chronological split	43.3	65.9	74.9	82.9	88.4	92.2

S4.2.3 Reactant ranking comparison

In order to illustrate and compare the novelty and effectiveness of our method, we showcase results of different methods, for the same synthesis goal. The ranking, and therefore preferred synthesis-starting point is shown for Top-1 in the left column from Figure S5 to Figure

S12. The results of 3 Methods are being compared: MHNreact, DNN corresponding to Segler and Waller [4] trained using the MHNreact framework with the same input and our re-implementation of a Transformer for single-step retrosynthesis [31].

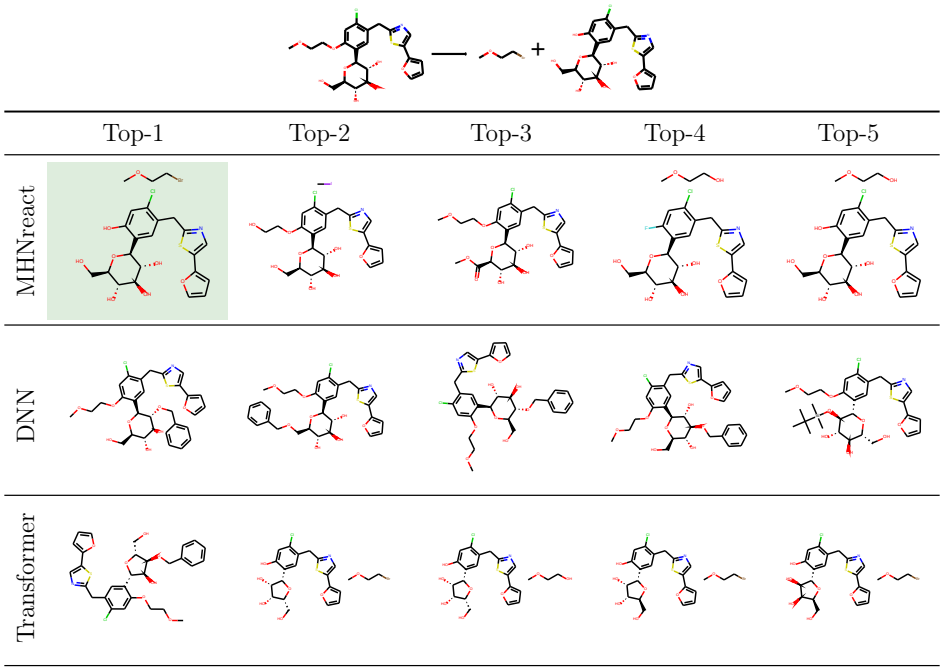


Figure S4: Sample #1117 from the test-set: Top-ranked molecules of different approaches. On the bottom, the ground truth reaction from the dataset is shown, and the matching predicted reactants are highlighted in green. Note that the transformer model makes a copy error, and changes the tetrahydropyran in the start molecule into a tetrahydrofuran.

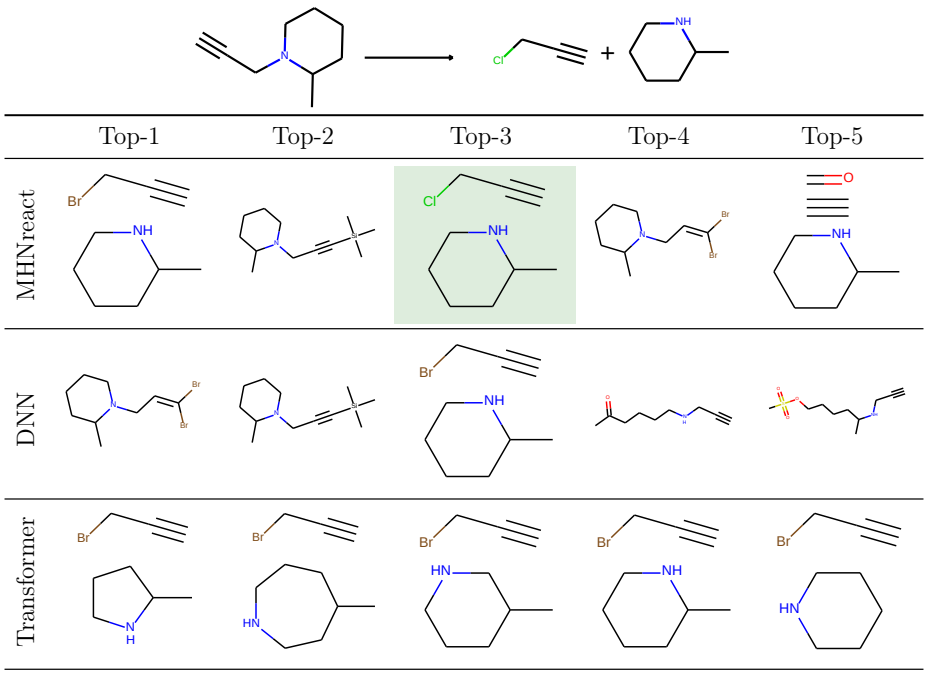


Figure S5: Sample #29 from the test-set. The transformer model exhibits copy mistakes, where the ring size of the piperidine 6-membered ring is changed to a pyrrolidine (5-membered) or azepane (7-membered) ring, or shifts the relative position of the methyl group around.

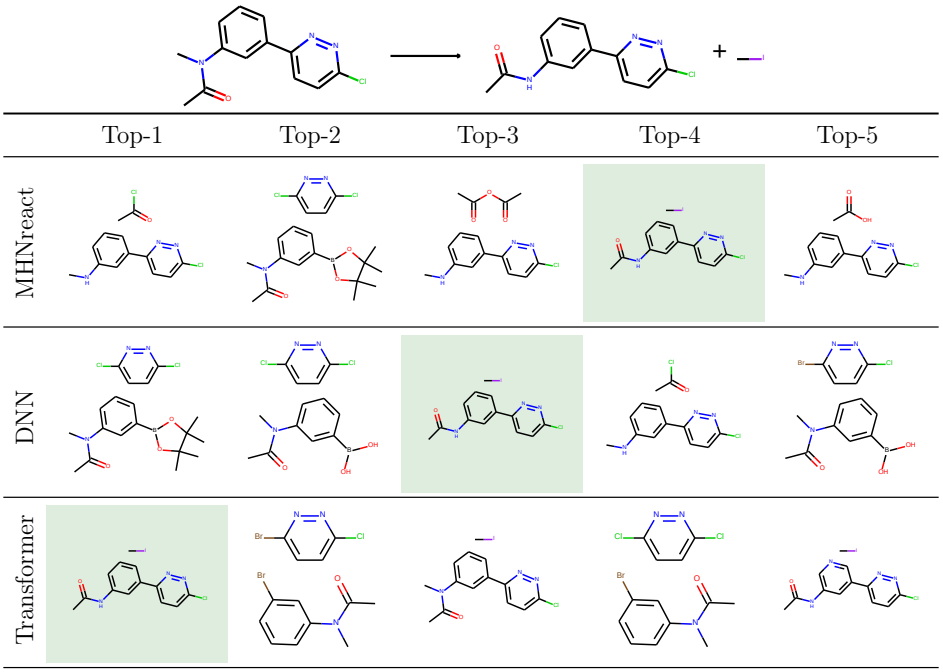
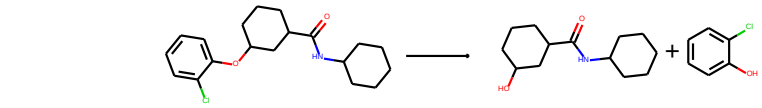


Figure S6: Sample #42 from the test-set.



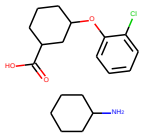
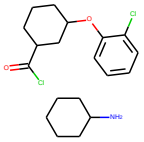
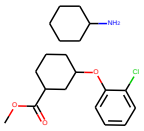
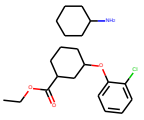
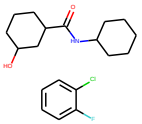
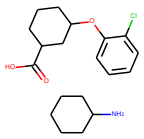
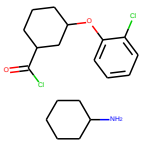
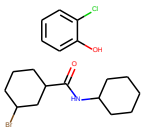
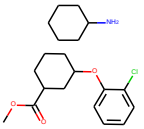
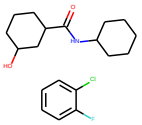
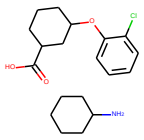
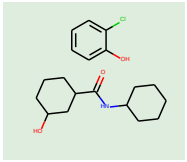
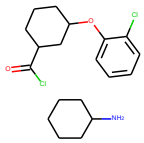
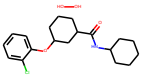
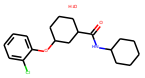
	Top-1	Top-2	Top-3	Top-4	Top-5
MHNreact					
DNN					
Transformer					

Figure S7: Sample #23 from the test-set.

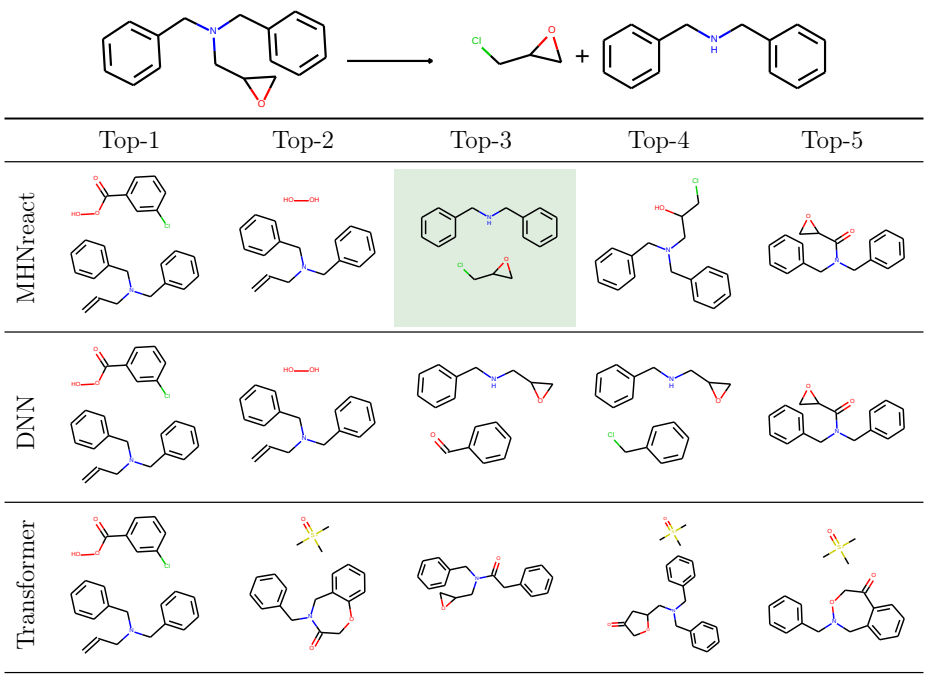


Figure S8: Sample #175 from the test-set.

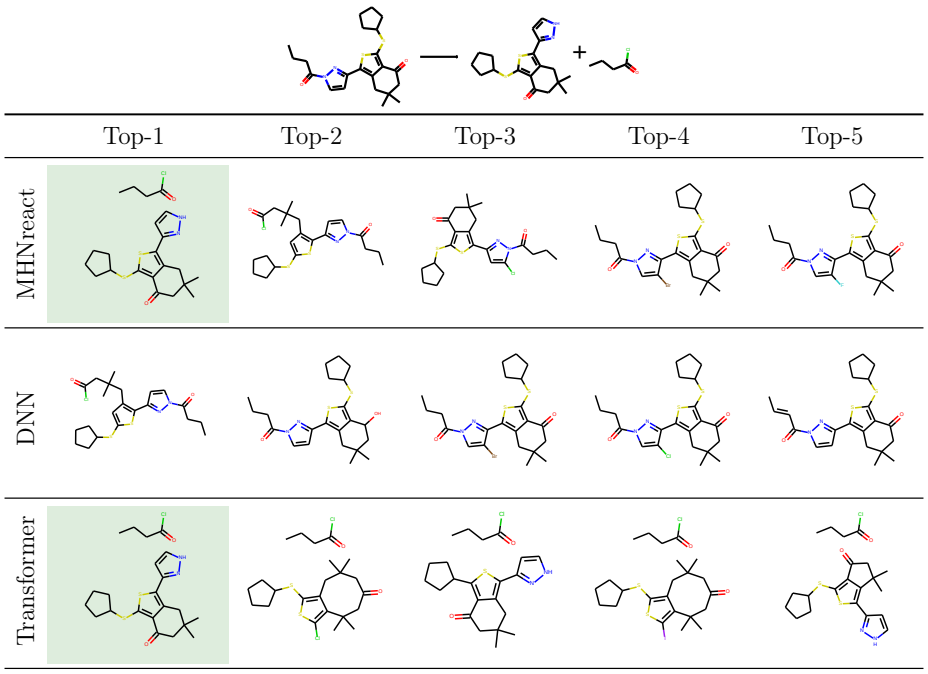


Figure S9: Sample #305 from the test-set. Note the copy errors in the Transformer, leading to ring expansion or contraction on the annelated ring

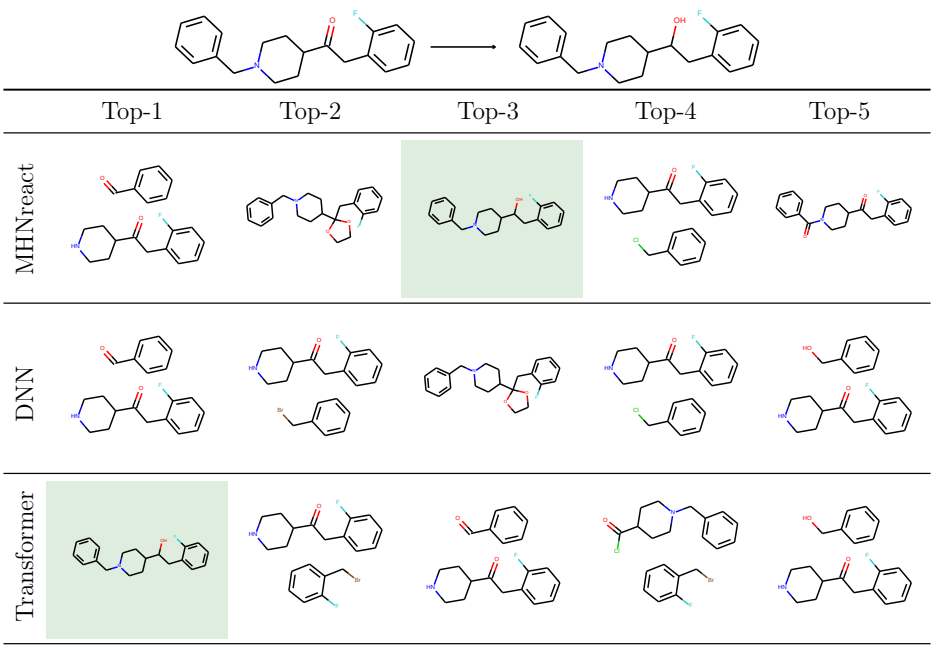


Figure S10: Sample #329 from the test-set.

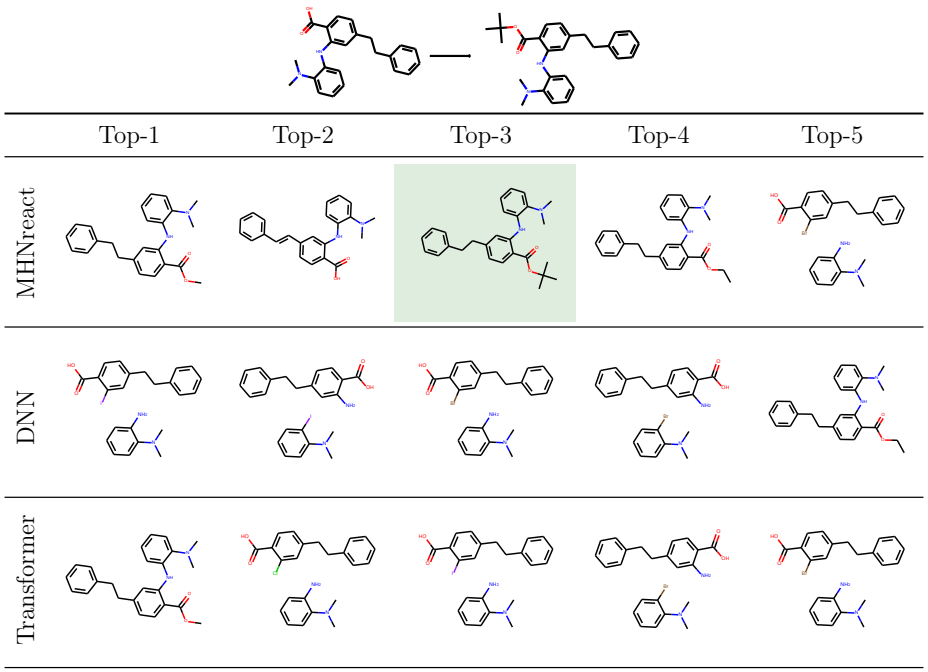


Figure S11: Sample #330 from the test-set.

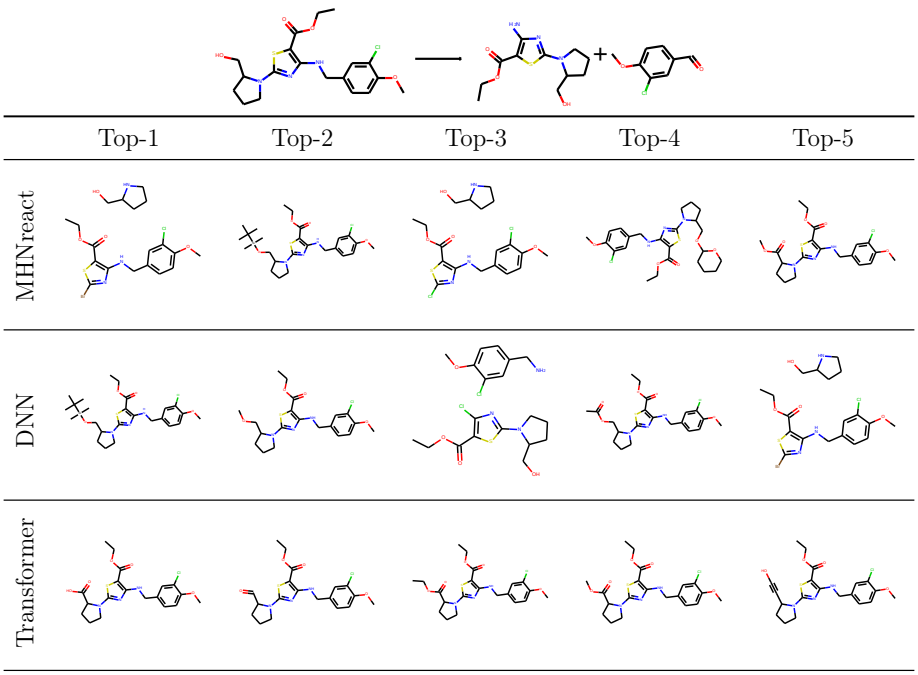


Figure S12: Sample #35 from the test-set. Ground-truth is not in one of the top-5 predictions.

S5 Hopfield Association Space

Figure S13 shows a t-Distributed Stochastic Neighbor Embedding (t-SNE) embedding of both the reaction fingerprints and the learned embeddings. Each point is colored according to its class as defined in[17].

For example, reactions belonging to the type `oxidations` can be distant in the fingerprint space (pink dots in the left plot in Figure S13), while in their learned representations are closer (pink dots in right figure). Note that our model did not have access to these reaction types. It can be seen that the chosen representation for reaction templates already captures information about the relationship, and the same reaction types are represented closer after embedding it using t-SNE.

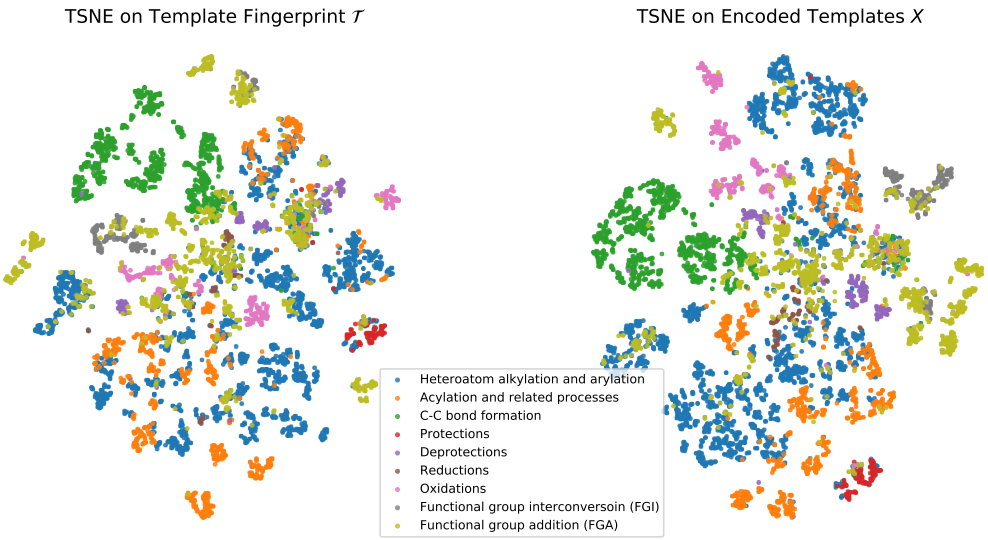


Figure S13: t-SNE downprojection of the reaction template fingerprints (left) and learned representations of reaction templates $\boldsymbol{X}$ (right). The colors represent reaction types of substructure-based expert systems as categorized by[17].

S6 Illustrative Example for Single-Step Retrosynthesis

We want to predict how to synthesize a given molecule as seen in Figure S14. First the molecule as well as multiple templates are encoded. Many templates could be applicable and would produce reactants, but we only want to predict relevant templates, those that are likely to correspondond to realistic reactions. MHNreact encodes the templates and associates them with the learned molecular representation and returns a ranking of the templates. Through encoding the templates, even templates with few training-data can be predicted. The templates are executed on the molecule in order of the provided ranking, in this case only the top-ranked template, which gives the reactants from which the desired molecule can likely be synthesized.

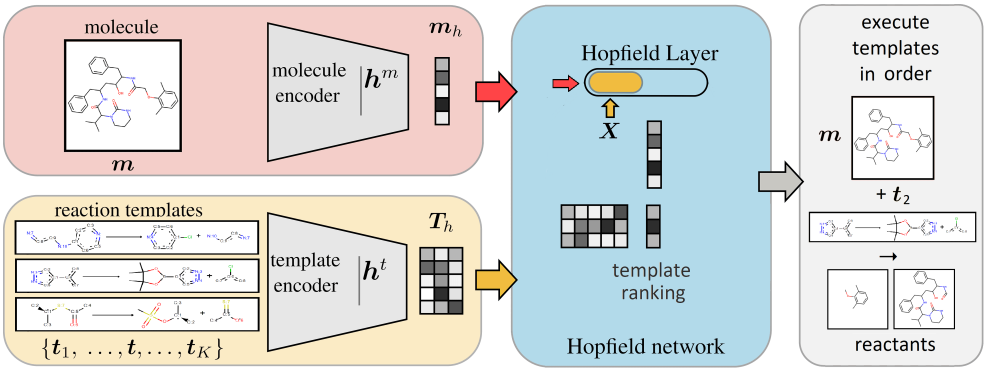


Figure S14: Illustration of the process for single-step retrosynthesis

S7 Objective and loss functions

Loss on retrieved patterns. We provide a more general view on the objective and the loss function from the perspective of Hopfield networks and retrieving patterns. The main idea is to retrieve patterns from label space, rather than from Hopfield space, because the loss functions operate in the label space. The last Hopfield layer of our architecture supplies both $\boldsymbol{p}$, the softmax vector of probabilities of drawing reaction templates, and $\boldsymbol{\xi}^{\text{new}}$, an average of reaction template representations $\boldsymbol{x}$. However, averages of reaction templates are no longer reaction templates, but we are interested in the *probability* ℓ_ξ of drawing an $\boldsymbol{x}_k$ that fits to $\boldsymbol{\xi}$.

The probability ℓ_ξ can still be computed via a slightly modified Hopfield network update, where instead of retrieving from a memory $\boldsymbol{X}$ of template representations in Hopfield space, we retrieve from the space of labels or scores. Such an update has been introduced previously and uses stored patterns that are augmented by labels[35] p.83ff.

The probability ℓ_ξ of drawing a $\boldsymbol{x}_k$ that fits to $\boldsymbol{\xi}$ can be computed by a modified Hopfield network update:

$$\boldsymbol{q}_\xi \;=\; \boldsymbol{L}\,\boldsymbol{p} \;=\; \boldsymbol{L}\,\mathrm{softmax}(\beta \boldsymbol{X}^T \boldsymbol{\xi}) \;, \quad \ell_\xi \;=\; \boldsymbol{1}^T\,\boldsymbol{q}_\xi \;, \tag{2}$$

where $\boldsymbol{L} \in \mathbb{R}^{K \times K}$ is the diagonal matrix of labels that are zero or one and $\boldsymbol{1} \in \mathbb{R}^K$ is the vector of ones. In general, $\boldsymbol{L}$ can be used to include unlabeled data points as stored patterns, where $\boldsymbol{L}$ is a Gram matrix times a diagonal label matrix to transfer label information from labeled data points to unlabeled data points. $L_{kk} = 1$ means that $\boldsymbol{x}_k$ fits to $\boldsymbol{\xi}$ and $L_{kk} = 0$ means that $\boldsymbol{x}_k$ does not fit to $\boldsymbol{\xi}$. In the context of the Hopfield networks, if more than one $\boldsymbol{x}_k$ fits to $\boldsymbol{\xi}$ then all $\boldsymbol{x}_k$ that fit to $\boldsymbol{\xi}$ constitute a metastable state. Instead of L_{kk} being equal to zero or one, L_{kk} can give a non-negative score for how well $\boldsymbol{x}_k$ does fit to $\boldsymbol{\xi}$. In this case ℓ_ξ is the *expected score*.

In this general view, our objective is to minimize $-\log(\ell_\xi)$. If L_{kk} is equal to zero or one, $\log(\ell_\xi)$ is the log-likelihood of drawing a fitting $\boldsymbol{x}_k$. If only one $\boldsymbol{x}_k$ fits (exactly one L_{kk} is one), then our objective is equivalent to the cross-entropy (CE) loss for multi-class classification. However, if more $\boldsymbol{x}_l$ are labeled as fitting, then our objective is different from CE, which might not be appropriate. If L_{kk} is a non-negative score for the molecule-template pair, our approach will maximize the expected score.

Cross-entropy loss. In a simple setting, in which each molecule only has a single correct reaction template in $\boldsymbol{T}$, a categorical cross-entropy loss is equivalent to the suggested loss. We encode the correct template by a one-hot vector $\boldsymbol{y} = (0, \ldots, 0, 1, 0, \ldots, 0)$, where 1 indicates the position of the correct template in the template set $\boldsymbol{t} = \boldsymbol{t}^k$. We then minimize the

cross-entropy loss function $\ell_{\text{CE}}(\boldsymbol{y}, \boldsymbol{p}) = \text{crossentropy}(\boldsymbol{y}, \boldsymbol{p})$ between ground truth $\boldsymbol{y}$ and the model's predictions $\boldsymbol{p}$ for a single pair of the training set and the overall loss is an average over all such pairs. The corresponding algorithm is given in Alg. 1.

Contrastive loss in Hopfield space. An alternative to the cross-entropy loss would be to use a contrastive loss on the retrieved pattern $\boldsymbol{\xi}^{\text{new}}$. This contrastive loss measures the cosine similarity of the retrieved pattern with the correct stored patterns with the InfoNCE function[13,36]:

$$\ell_{\text{p}}(\boldsymbol{\xi}^{\text{new}}, \boldsymbol{x}^{+}, \boldsymbol{X}^{-}) = \text{InfoNCE}(\boldsymbol{\xi}^{\text{new}}, \boldsymbol{x}^{+}, \boldsymbol{X}^{-}) \tag{3}$$

where $\text{sim}(.,.)$ is a pairwise similarity function, $\boldsymbol{x}^{+}$ is the representation of the correct reaction templates, and $\boldsymbol{X}^{-}$ is the set of representations of the incorrect reaction templates, that are contrasted against each other. This loss is equivalent to cross-entropy loss if a) $1/\tau = \beta$, b) the similarity function is the dot product, and c) $\boldsymbol{\xi}$ is used instead of $\boldsymbol{\xi}^{\text{new}}$. Our experiments show that this loss can lead to models with comparable performance to those trained with cross-entropy loss. The according algorithm with pattern loss as alternative loss is shown in Alg. 1. We envision that advances in estimating mutual information[37,38] and contrastive learning[39] could lead to improved zero- and few-shot capabilities of our model.

We provide a formulation of our method as simplified pseudo-code in a Python/Pytorch[12]-like language (see Algorithm 1). The pseudo-code provides a version with two stacked Hopfield layers and three possible formulations of the loss function.

Algorithm 1 MHN for reaction template prediction (simplified, e.g. skip-connections omitted).

```
#mol_encoder() — e.g. fully-connected or MPNN. Maps to dimension d_m.
#template_encoder1() Maps to dimension d_t1.
#template_encoder2() Maps to dimension d_t2.
#m_train, t_train — pair of product molecule and reaction template from training set
#T — set of K reaction templates including t_train
#d — dimension of Hopfield space

## FORWARD PASS
T1_h = template_encoder1(T)#[d_t1,K]
T2_h = template_encoder2(T)#[d_t2,K]
m_h = mol_encoder(m_train)#[d_m,1]
xinew1,_,_ = Hopfield(m_h,T1_h,dim=d)
xinew,p,X = Hopfield(xinew1,T2_h,dim=d)
p=pool(p,axis=1) #[K,1]

## LOSS
# cross-entropy loss, association loss
label = where(T==t_train)#[K,1]
loss = cross_entropy(p,label)
# alternative 1:  Hopfield loss
L = diag(where(T==t_train))#[K,K]
loss = -log(sum(L@p))
# alternative 2:  contrastive loss
label = where(T==t_train)#[K,1]
pos = X[label] #[d,1]
neg_label = where(T!=t_train)#[K,1]
neg = X[neg_label] #[d,K-1]
loss = -InfoNCE(xinew,pos,neg)
```

S8 List of Acronyms

CASP computer-assisted synthesis planning

CLIP Contrastive Language–Image Pre-training

ConVIRT Contrastive VIsual Representation Learning from Text

DNN deep neural network

ECFP extended connectivity fingerprint

FPF fingerprint filter

FP fingerprint

GLN graph logic network

InfoNCE Information Noise-Contrastive Estimation

MACCS molecular access system

MHFP mini-hash fingerprint

MHN modern Hopfield network

ML machine learning

NN neural network

SELU self-normalizing linear unit

SMARTS SMILES arbitrary target specification

SMILES simplified molecular-input line-entry system

USPTO United States patent and trademark office

t-SNE t-Distributed Stochastic Neighbor Embedding

References

(1) Baylon, J. L.; Cilfone, N. A.; Gulcher, J. R.; Chittenden, T. W. Enhancing Retrosynthetic Reaction Prediction with Deep Learning Using Multiscale Reaction Classification. *J. Chem. Inf. Model.* **2019**,

(2) Ishida, S.; Terayama, K.; Kojima, R.; Takasu, K.; Okuno, Y. Prediction and Interpretable Visualization of Retrosynthetic Reactions Using Graph Convolutional Networks. *J. Chem. Inf. Model.* **2019**, *59*, 5026–5033.

(3) Fortunato, M. E.; Coley, C. W.; Barnes, B. C.; Jensen, K. F. Data Augmentation and Pretraining for Template-Based Retrosynthetic Prediction in Computer-Aided Synthesis Planning. *J. Chem. Inf. Model.* **2020**, *60*, 3398–3407.

(4) Segler, M. H. S.; Waller, M. P. Neural-Symbolic Machine Learning for Retrosynthesis and Reaction Prediction. *Chemistry* **2017**, *23*, 5966–5971.

(5) Segler, M. H. S.; Preuss, M.; Waller, M. P. Planning Chemical Syntheses with Deep Neural Networks and Symbolic AI. *Nature* **2018**, *555*, 604–610.

(6) Bjerrum, E. J.; Thakkar, A.; Engkvist, O. Artificial Applicability Labels for Improving Policies in Retrosynthesis Prediction. *ChemRxiv* **2020**,

(7) Dai, H.; Li, C.; Coley, C.; Dai, B.; Song, L. Retrosynthesis Prediction with Conditional Graph Logic Network. *Adv. Neural Inf. Process. Syst.* **2019**, *32*, 8872–8882.

(8) Sun, R.; Dai, H.; Li, L.; Kearnes, S.; Dai, B. Energy-based View of Retrosynthesis. *arXiv (Machine Learning)* **2020**,

(9) Coley, C. W.; Green, W. H.; Jensen, K. F. RDChiral: An RDKit Wrapper for Handling Stereochemistry in Retrosynthetic Template Extraction and Application. *J. Chem. Inf. Model.* **2019**, *59*, 2529–2537.

(10) Morgan, H. L. The Generation of a Unique Machine Description for Chemical Structures-A Technique Developed at Chemical Abstracts Service. *J. Chem. Doc.* **1965**, *5*, 107–113.

(11) Landrum, G. RDKit: Open-Source Cheminformatics. **2006**, accessed on 2020-01-01.

(12) Paszke, A.; Gross, S.; Massa, F.; Lerer, A.; Bradbury, J.; Chanan, G.; Killeen, T.; Lin, Z.; Gimelshein, N.; Antiga, L.; Desmaison, A.; Köpf, A.; Yang, E.; DeVito, Z.; Raison, M. Tejani, A.; Chilamkurthy, S.; Steiner, B.; Fang, L.; Bai, J.; Chintala, S. Pytorch: An imperative style, high-performance deep learning library. *Adv. Neural Inf. Process. Syst.* **2019**, *32*, 8026–8037.

(13) Oord, A. v. d.; Li, Y.; Vinyals, O. Representation learning with contrastive predictive coding. *arXiv (Machine Learning)* **2018**,

(14) Ba, J. L.; Kiros, J. R.; Hinton, G. E. Layer normalization. *arXiv (Machine Learning)* **2016**,

(15) Loshchilov, I.; Hutter, F. Decoupled weight decay regularization. *arXiv (Machine Learning)* **2017**,

(16) Coley, C. W.; Rogers, L.; Green, W. H.; Jensen, K. F. Computer-Assisted Retrosynthesis Based on Molecular Similarity. *ACS Cent. Sci.* **2017**, *3*, 1237–1245.

(17) Schneider, N.; Lowe, D. M.; Sayle, R. A.; Landrum, G. A. Development of a Novel Fingerprint for Chemical Reactions and Its Application to Large-Scale Reaction Classification and Similarity. *J. Chem. Inf. Model.* **2015**, *55*, 39–53.

(18) Tetko, I. V.; Karpov, P.; Van Deursen, R.; Godin, G. State-of-the-Art Augmented NLP Transformer Models for Direct and Single-Step Retrosynthesis. *Nat. Commun.* **2020**, *11*, 5575.

(19) Probst, D.; Reymond, J.-L. A probabilistic molecular fingerprint for big data settings. *J. Cheminf.* **2018**, *10*, 1–12.

(20) Klambauer, G.; Unterthiner, T.; Mayr, A.; Hochreiter, S. Self-normalizing neural networks. *Adv. Neural Inf. Process. Syst.* **2017**, 972–981.

(21) Rogers, D.; Hahn, M. Extended-Connectivity Fingerprints. *J. Chem. Inf. Model.* **2010**, *50*, 742–754.

(22) Yan, C.; Ding, Q.; Zhao, P.; Zheng, S.; Yang, J.; Yu, Y.; Huang, J. RetroXpert: Decompose Retrosynthesis Prediction like a Chemist. *arXiv (Machine Learning)* **2020**,

(23) Somnath, V. R.; Bunne, C.; Coley, C. W.; Krause, A.; Barzilay, R. Learning graph models for template-free retrosynthesis. *arXiv (Machine Learning)* **2020**,

(24) Lee, H.; Ahn, S.; Seo, S.-W.; Song, Y. Y.; Hwang, S.-J.; Yang, E.; Shin, J. RetCL: A Selection-Based Approach for Retrosynthesis via Contrastive Learning. *arXiv (Machine Learning)* **2021**,

(25) Hasic, H.; Ishida, T. Single-Step Retrosynthesis Prediction Based on the Identification of Potential Disconnection Sites Using Molecular Substructure Fingerprints. *J. Chem. Inf. Model.* **2021**, *61*, 641–652.

(26) Guo, Z.; Wu, S.; Ohno, M.; Yoshida, R. A Bayesian Algorithm for Retrosynthesis. *J. Chem. Inf. Model.* **2020**, *60*, 4474–4486.

(27) Schwaller, P.; Laino, T.; Gaudin, T.; Bolgar, P.; Hunter, C. A.; Bekas, C.; Lee, A. A. Molecular Transformer: A Model for Uncertainty-Calibrated Chemical Reaction Prediction. *ACS Cent. Sci.* **2019**, *5*, 1572–1583.

(28) Ishiguro, K.; Ujihara, K.; Sawada, R.; Akita, H.; Kotera, M. Data Transfer Approaches to Improve Seq-to-Seq Retrosynthesis. *arXiv (Machine Learning)* **2020**,

(29) Ucak, U. V.; Kang, T.; Ko, J.; Lee, J. Substructure-based neural machine translation for retrosynthetic prediction. *J. Cheminf.* **2021**, *13*, 4.

(30) Liu, B.; Ramsundar, B.; Kawthekar, P.; Shi, J.; Gomes, J.; Luu Nguyen, Q.; Ho, S.; Sloane, J.; Wender, P.; Pande, V. Retrosynthetic Reaction Prediction Using Neural Sequence-to-Sequence Models. *ACS Cent. Sci.* **2017**, *3*, 1103–1113.

(31) Karpov, P.; Godin, G.; Tetko, I. V. A transformer model for retrosynthesis. *Int. Conf. on Artif. Neur. Netw.* **2019**, 817–830.

(32) Wang, X.; Li, Y.; Qiu, J.; Chen, G.; Liu, H.; Liao, B.; Hsieh, C.-Y.; Yao, X. RetroPrime: A Diverse, Plausible and Transformer-Based Method for Single-Step Retrosynthesis Predictions. *Chem. Eng. J.* **2021**, *420*, 129845.

(33) Sacha, M.; Błaz, M.; Byrski, P.; Dabrowski-Tumansski, P.; Chrominski, M.; Loska, R.; Włodarczyk-Pruszynski, P.; Jastrzebski, S. Molecule edit graph attention network: modeling chemical reactions as sequences of graph edits. *J. Chem. Inf. Model.* **2021**, *61*, 3273–3284.

(34) Lowe, D. M. Extraction of chemical structures and reactions from the literature. Ph.D. thesis, University of Cambridge, 2012.

(35) Ramsauer, H.; Schäfl, B.; Lehner, J.; Seidl, P.; Widrich, M.; Gruber, L.; Holzleitner, M.; Adler, T.; Kreil, D.; Kopp, M. K.; Klambauer, G.; Brandstetter, J.; Hochreiter, S. Hopfield Networks is All You Need. *Int. Conf. Learn. Rep.* **2021**,

(36) Chen, T.; Kornblith, S.; Norouzi, M.; Hinton, G. A simple framework for contrastive learning of visual representations. *Int. Conf. Mach. Learn.* **2020**, 1597–1607.

(37) Poole, B.; Ozair, S.; vanDenOord, A.; Alemi, A. A.; Tucker, G. On Variational Bounds of Mutual Information. *Proc. 36th Int. Conf. Mach. Learn.* **2019**, *97*, 5171–5180.

(38) Cheng, P.; Hao, W.; Dai, S.; Liu, J.; Gan, Z.; Carin, L. CLUB: A Contrastive Log-ratio Upper Bound of Mutual Information. *Proc. 37th Int. Conf. Mach. Learn.* **2020**, *119*, 1779–1788.

(39) Fürst, A.; Rumetshofer, E.; Tran, V.; Ramsauer, H.; Tang, F.; Lehner, J.; Kreil, D.; Kopp, M.; Klambauer, G.; Bitto-Nemling, A.; Hochreiter, S. CLOOB: Modern Hopfield Networks with InfoLOOB Outperform CLIP. *arXiv (Machine Learning)* **2021**,

3 Conclusion and Outlook

The work in this thesis has focused on advancing the application of generative models in drug discovery, concentrating on two main aspects: Firstly, we identified limitations in the evaluation of generative models for de novo molecular design, and proposed ways to make evaluation more informative and relevant to practical applications. Secondly we introduced a novel template-based model for retrosynthesis prediction that exceeds the performance of existing methods, performing particularly well on rare reaction templates.

In the first part of this thesis, we showed how established ways of evaluating distribution-learning models cannot differentiate complex models from trivial baseline generators, highlighting the need for more informative evaluation metrics. Furthermore, we showed how goal-directed generative models can overfit to machine learning based scoring functions and exhibit biases towards the training data. We proposed control scores as a diagnostic tool to identify overfitting and biases in goal-directed molecular generators. The control scores have been further investigated in more detail by others (Turk et al., 2022) and it is worth pointing out that that while they can detect overfitting, they cannot establish its absence. Similar findings have been reported by

The second part of this thesis introduced a diversity-based benchmark for goal-directed molecule generators aiming at measuring their performance in generating diverse high-scoring molecules. This benchmark addresses the shortcomings of previous benchmarks by adressing the issues of inadequate diversity measures, non-standardized compute budgets, and lack of model adaptation to the diverse optimization setting. We used this benchmark to evaluate a range of generative models comparing them in a meaningful way.

The last part of this thesis introduced a novel template-based model for retrosynthesis prediction based on Modern Hopfield Networks. This model leverages a multi-modal approach that combines reaction templates and target molecules. Our model reaches state-of-the-art performance while maintaining a significantly lower computational cost compared to existing methods. The model is able to generalize over reaction templates and performs particularly well on rare reaction templates.

While our work has advanced evaluation of generative models for de novo design, it is hard to translate current benchmark results into practical utility. Many of the scoring functions used in current studies are still rather simple one-dimensional functions that do not capture the the complexities of real-world drug discovery projects. We believe there is a need for more comprehensive benchmarks that better take the synthesizability, chemistry and novelty of the generated molecules into account. Furthermore, the model evaluation should move in a direction that better reflects the scale/resources in real-world drug discovery projects instead of being contrained to short algorithm runtimes.

Similarly, we think that there is a need for more holistic benchmarks of single-step retrosynthesis prediction models. While single-step performance is an important, we share the view of Maziarz et al. (2024) that there is a need for more standardized benchmarks that better reflect the performance aspects of single-step methods that are relevant in multi-step retrosynthesis planning. Furthermore, we believe it is important to acknowledege the limitations of a pure ML approach based on reaction databases as pointed out in (Strieth-Kalthoff et al., 2024). Those limitations are owed to the quality of the data and the fact that some information needed for robust models is missing from current reaction databases.

Overall we think that both fields would benefit from more collaboration between machine learning researchers and chemists, in order to better identify practical needs and to find computational solutions to these needs.

Bibliography

Bagal, V., Aggarwal, R., Vinod, P. K., and Priyakumar, U. D. (2022). "MolGPT: Molecular Generation Using a Transformer-Decoder Model". In: *J. Chem. Inf. Model.* 62.9, pp. 2064–2076. DOI: 10.1021/acs.jcim.1c00600.

Bengio, E., Jain, M., Korablyov, M., Precup, D., and Bengio, Y. (2021). *Flow Network Based Generative Models for Non-Iterative Diverse Candidate Generation*. DOI: 10.48550/arXiv.2106.04399. arXiv: 2106.04399 [cs]. Pre-published.

Bento, A. P., Gaulton, A., Hersey, A., Bellis, L. J., Chambers, J., Davies, M., Krüger, F. A., Light, Y., Mak, L., McGlinchey, S., Nowotka, M., Papadatos, G., Santos, R., and Overington, J. P. (2014). "The ChEMBL Bioactivity Database: An Update". In: *Nucleic Acids Res* 42.D1, pp. D1083–D1090. DOI: 10.1093/nar/gkt1031.

Brown, N., Fiscato, M., Segler, M. H., and Vaucher, A. C. (2019). "GuacaMol: Benchmarking Models for de Novo Molecular Design". In: *J. Chem. Inf. Model.* 59.3, pp. 1096–1108. DOI: 10.1021/acs.jcim.8b00839.

Chen, H., Engkvist, O., Wang, Y., Olivecrona, M., and Blaschke, T. (2018). "The Rise of Deep Learning in Drug Discovery". In: *Drug Discovery Today* 23.6, pp. 1241–1250. DOI: 10.1016/j.drudis.2018.01.039.

Cohen-Karlik, E., Rozenberg, E., and Freedman, D. (2024). "Overcoming Order in Autoregressive Graph Generation for Molecule Generation". In: *Transactions on Machine Learning Research*.

Coley, C. W., Green, W. H., and Jensen, K. F. (2018). "Machine Learning in Computer-Aided Synthesis Planning". In: *Acc. Chem. Res.* 51.5, pp. 1281–1289. DOI: 10.1021/acs.accounts.8b00087.

Corey, E. J. and Wipke, W. T. (1969). "Computer-Assisted Design of Complex Organic Syntheses". In: *Science* 166.3902, pp. 178–192. JSTOR: 1727162.

Corey, E. J. (1991). "The Logic of Chemical Synthesis: Multistep Synthesis of Complex Carbogenic Molecules (Nobel Lecture)". In: *Angewandte Chemie International Edition in English* 30.5, pp. 455–465. DOI: 10.1002/anie.199104553.

Dai, H., Li, C., Coley, C. W., Dai, B., and Song, L. (2020). "Retrosynthesis Prediction with Conditional Graph Logic Network". arXiv: 2001.01408 [cs, stat].

Dai, H., Tian, Y., Dai, B., Skiena, S., and Song, L. (2018). "Syntax-Directed Variational Autoencoder for Structured Data". arXiv: 1802.08786 [cs].

De Cao, N. and Kipf, T. (2018). "MolGAN: An Implicit Generative Model for Small Molecular Graphs". arXiv: 1805.11973 [cs, stat].

Degen, J., Wegscheid-Gerlach, C., Zaliani, A., and Rarey, M. (2008). "On the Art of Compiling and Using 'Drug-Like' Chemical Fragment Spaces". In: *ChemMedChem* 3.10, pp. 1503–1507. DOI: 10.1002/cmdc.200800178.

DiMasi, J. A., Grabowski, H. G., and Hansen, R. W. (2016). "Innovation in the Pharmaceutical Industry: New Estimates of R&D Costs". In: *Journal of Health Economics* 47, pp. 20–33. DOI: 10.1016/j.jhealeco.2016.01.012.

Du, Y., Jamasb, A. R., Guo, J., Fu, T., Harris, C., Wang, Y., Duan, C., Liò, P., Schwaller, P., and Blundell, T. L. (2024). "Machine Learning-Aided Generative Molecular Design". In: *Nat Mach Intell* 6.6, pp. 589–604. DOI: 10.1038/s42256-024-00843-5.

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

English: Caffeine 3D Structure (2010). URL: https://commons.wikimedia.org/wiki/File:Caffeine_3d_structure.png (visited on 06/09/2024).

Fortunato, M. E., Coley, C. W., Barnes, B. C., and Jensen, K. F. (2020). "Data Augmentation and Pretraining for Template-Based Retrosynthetic Prediction in Computer-Aided Synthesis Planning". In: *J. Chem. Inf. Model.* 60.7 (7), pp. 3398–3407. DOI: 10.1021/acs.jcim.0c00403.

Gao, W., Fu, T., Sun, J., and Coley, C. W. (2022). *Sample Efficiency Matters: A Benchmark for Practical Molecular Optimization*. DOI: 10.48550/arXiv.2206.12411. arXiv: 2206.12411 [cs, q-bio]. Pre-published.

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. (2018). "Automatic Chemical Design Using a Data-Driven Continuous Representation of Molecules". In: *ACS Cent. Sci.* 4.2, pp. 268–276. DOI: 10.1021/acscentsci.7b00572.

Goodfellow, I. J., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., and Bengio, Y. (2014). "Generative Adversarial Networks". arXiv: 1406.2661 [cs, stat].

Goodfellow, I. J., Shlens, J., and Szegedy, C. (2015). *Explaining and Harnessing Adversarial Examples*. DOI: 10.48550/arXiv.1412.6572. arXiv: 1412.6572 [cs, stat]. Pre-published.

Gorse, A.-D. (2006). "Diversity in Medicinal Chemistry Space". In: *Current Topics in Medicinal Chemistry* 6.1, pp. 3–18.

Guimaraes, G. L., Sanchez-Lengeling, B., Outeiral, C., Farias, P. L. C., and Aspuru-Guzik, A. (2017). "Objective-Reinforced Generative Adversarial Networks (ORGAN) for Sequence Generation Models". arXiv: 1705.10843 [cs, stat].

Guo, J. and Schwaller, P. (2023). *Augmented Memory: Capitalizing on Experience Replay to Accelerate De Novo Molecular Design*. arXiv: 2305.16160 [cs, q-bio]. URL: http://arxiv.org/abs/2305.16160 (visited on 09/11/2023). Pre-published.

Hartenfeller, M., Zettl, H., Walter, M., Rupp, M., Reisen, F., Proschak, E., Weggen, S., Stark, H., and Schneider, G. (2012). "DOGS: Reaction-Driven de Novo Design of Bioactive Compounds". In: *PLOS Computational Biology* 8.2, e1002380. DOI: 10.1371/journal.pcbi.1002380.

Heller, S. R., McNaught, A., Pletnev, I., Stein, S., and Tchekhovskoi, D. (2015). "InChI, the IUPAC International Chemical Identifier". In: *Journal of Cheminformatics* 7.1, p. 23. DOI: 10.1186/s13321-015-0068-4.

Hofmarcher, M., Mayr, A., Rumetshofer, E., Ruch, P., Renz, P., Schimunek, J., Seidl, P., Vall, A., Widrich, M., Hochreiter, S., and Klambauer, G. (2020). *Large-Scale Ligand-Based Virtual Screening for SARS-CoV-2 Inhibitors Using Deep Neural Networks*. DOI: 10.48550/arXiv.2004.00979. arXiv: 2004.00979 [cs, q-bio, stat]. Pre-published.

Irwin, J. J., Sterling, T., Mysinger, M. M., Bolstad, E. S., and Coleman, R. G. (2012). "ZINC: A Free Tool to Discover Chemistry for Biology". In: *J. Chem. Inf. Model.* 52.7, pp. 1757–1768. DOI: 10.1021/ci3001277.

Jaques, N., Gu, S., Bahdanau, D., Hernández-Lobato, J. M., Turner, R. E., and Eck, D. (2016). "Sequence Tutor: Conservative Fine-Tuning of Sequence Generation Models with KL-control". arXiv: 1611.02796 [cs].

Jensen, J. H. (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. DOI: 10.1039/C8SC05372C.

Jin, W., Barzilay, R., and Jaakkola, T. (2018). "Junction Tree Variational Autoencoder for Molecular Graph Generation". arXiv: 1802.04364 [cs, stat].

Kadurin, A., Nikolenko, S., Khrabrov, K., Aliper, A., and Zhavoronkov, A. (2017). "druGAN: An Advanced Generative Adversarial Autoencoder Model for de Novo Generation of New Molecules with Desired Molecular Properties in Silico". In: *Mol. Pharmaceutics* 14.9, pp. 3098–3104. DOI: 10.1021/acs.molpharmaceut.7b00346.

Karpov, P., Godin, G., and Tetko, I. V. (2019). "A Transformer Model for Retrosynthesis". In: *Artificial Neural Networks and Machine Learning – ICANN 2019: Workshop and Special Sessions*. Ed. by I. V. Tetko, V. Kůrková, P. Karpov, and F. Theis. Lecture Notes in Computer Science. Springer International Publishing, pp. 817–830.

Kim, S., Thiessen, P. A., Bolton, E. E., Chen, J., Fu, G., Gindulyte, A., Han, L., He, J., He, S., Shoemaker, B. A., Wang, J., Yu, B., Zhang, J., and Bryant, S. H. (2016). "PubChem Substance and Compound Databases". In: *Nucleic Acids Res* 44 (Database issue), pp. D1202–D1213. DOI: 10.1093/nar/gkv951. pmid: 26400175.

Kingma, D. P. and Welling, M. (2013). *Auto-Encoding Variational Bayes*. DOI: 10.48550/arXiv.1312.6114. arXiv: 1312.6114 [cs, stat]. Pre-published.

Klambauer, G., Hochreiter, S., and Rarey, M. (2019). "Machine Learning in Drug Discovery". In: *J. Chem. Inf. Model.* 59.3, pp. 945–946. DOI: 10.1021/acs.jcim.9b00136.

Krenn, M., Ai, Q., Barthel, S., Carson, N., Frei, A., Frey, N. C., Friederich, P., Gaudin, T., Gayle, A. A., Jablonka, K. M., Lameiro, R. F., Lemm, D., Lo, A., Moosavi, S. M., Nápoles-Duarte, J. M., Nigam, A., Pollice, R., Rajan, K., Schatzschneider, U., Schwaller, P., Skreta, M., Smit, B., Strieth-Kalthoff, F., Sun, C., Tom, G., Falk Von Rudorff, G., Wang, A., White, A. D., Young, A., Yu, R., and Aspuru-Guzik, A. (2022). "SELFIES and the Future of Molecular String Representations". In: *Patterns* 3.10, p. 100588. DOI: 10.1016/j.patter.2022.100588.

Krenn, M., Häse, F., Nigam, A., Friederich, P., and Aspuru-Guzik, A. (2020). "Self-Referencing Embedded Strings (SELFIES): A 100% Robust Molecular String Representation". arXiv: 1905.13741 [physics, physics:quant-ph, stat].

Kusner, M. J., Paige, B., and Hernández-Lobato, J. M. (2017). "Grammar Variational Autoencoder". arXiv: 1703.01925 [stat].

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

Li, Y., Vinyals, O., Dyer, C., Pascanu, R., and Battaglia, P. (2018). "Learning Deep Generative Models of Graphs". arXiv: 1803.03324 [cs, stat].

Liu, Q., Allamanis, M., Brockschmidt, M., and Gaunt, A. (2018). "Constrained Graph Variational Autoencoders for Molecule Design". In: *Advances in Neural Information Processing Systems 31*. Ed. by S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett. Curran Associates, Inc., pp. 7795–7804.

Madhawa, K., Ishiguro, K., Nakago, K., and Abe, M. (2019). "GraphNVP: An Invertible Flow Model for Generating Molecular Graphs". arXiv: 1905.11600 [cs, stat].

Makurvet, F. D. (2021). "Biologics vs. Small Molecules: Drug Costs and Patient Access". In: *Medicine in Drug Discovery* 9, p. 100075. DOI: 10.1016/j.medidd.2020.100075.

Martin, Y. C. (2001). "Diverse Viewpoints on Computational Aspects of Molecular Diversity". In: *J. Comb. Chem.* 3.3, pp. 231–250. DOI: 10.1021/cc000073e.

Mayr, A., Klambauer, G., Unterthiner, T., and Hochreiter, S. (2016). "DeepTox: Toxicity Prediction Using Deep Learning". In: *Frontiers in Environmental Science* 3.

Maziarz, K., Tripp, A., Liu, G., Stanley, M., Xie, S., Gaiński, P., Seidl, P., and Segler, M. (2024). "Re-Evaluating Retrosynthesis Algorithms with Syntheseus". In: *Faraday Discuss.*, 10.1039.D4FD00093E. DOI: 10.1039/D4FD00093E. arXiv: 2310.19796 [cs, q-bio].

Mazuz, E., Shtar, G., Shapira, B., and Rokach, L. (2023). "Molecule Generation Using Transformers and Policy Gradient Reinforcement Learning". In: *Sci Rep* 13.1, p. 8799. DOI: 10.1038/s41598-023-35648-w.

Méndez-Lucio, O., Baillif, B., Clevert, D.-A., Rouquié, D., and Wichard, J. (2018). "De Novo Generation of Hit-like Molecules from Gene Expression Signatures Using Artificial Intelligence". In: DOI: 10.26434/chemrxiv.7294388.v1.

Mullard, A. (2016). "Parsing Clinical Success Rates". In: *Nature Reviews Drug Discovery* 15.7, pp. 447–447. DOI: 10.1038/nrd.2016.136.

Nam, J. and Kim, J. (2016). "Linking the Neural Machine Translation and the Prediction of Organic Chemistry Reactions". arXiv: 1612.09529 [cs].

Nigam, A., Pollice, R., Krenn, M., Gomes, G. d. P., and Aspuru-Guzik, A. (2021). "Beyond Generative Models: Superfast Traversal, Optimization, Novelty, Exploration and Discovery (STONED) Algorithm for Molecules Using SELFIES". In: *Chem. Sci.* 12.20, pp. 7079–7090. DOI: 10.1039/D1SC00231G.

Noutahi, E., Gabellini, C., Craig, M., Lim, J. S. C., and Tossou, P. (2023). *Gotta Be SAFE: A New Framework for Molecular Design*. DOI: 10.48550/arXiv.2310.10773. arXiv: 2310.10773 [cs, q-bio]. Pre-published.

O'Boyle, N. and Dalke, A. (2018). "DeepSMILES: An Adaptation of SMILES for Use in Machine-Learning of Chemical Structures". In: DOI: 10.26434/chemrxiv.7097960.v1.

Olivecrona, M., Blaschke, T., Engkvist, O., and Chen, H. (2017). "Molecular De-Novo Design through Deep Reinforcement Learning". In: *Journal of Cheminformatics* 9.1, p. 48. DOI: 10.1186/s13321-017-0235-x.

Paul, S. M., Mytelka, D. S., Dunwiddie, C. T., Persinger, C. C., Munos, B. H., Lindborg, S. R., and Schacht, A. L. (2010). "How to Improve R&D Productivity: The Pharmaceutical Industry's Grand Challenge". In: *Nat Rev Drug Discov* 9.3, pp. 203–214. DOI: 10.1038/nrd3078.

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., Nikolenko, S., Aspuru-Guzik, A., and Zhavoronkov, A. (2020). "Molecular Sets (MOSES): A Benchmarking Platform for Molecular Generation Models". In: *Frontiers in Pharmacology* 11.

Preuer, K., Renz, P., Unterthiner, T., Hochreiter, S., and Klambauer, G. (2018). "Fréchet ChemNet Distance: A Metric for Generative Models for Molecules in Drug Discovery". In: *J. Chem. Inf. Model.* 58.9, pp. 1736–1741. DOI: 10.1021/acs.jcim.8b00234.

Ramsauer, H., Schäfl, B., Lehner, J., Seidl, P., Widrich, M., Gruber, L., Holzleitner, M., Pavlović, M., Sandve, G. K., Greiff, V., Kreil, D., Kopp, M., Klambauer, G., Brandstetter, J., and Hochreiter, S. (2020). "Hopfield Networks Is All You Need". arXiv: 2008.02217 [cs, stat].

Renz, P., Cutajar, K., Twomey, N., Cheung, G. K. C., and Xie, H. (2023). "Low-Count Time Series Anomaly Detection". In: *2023 IEEE 33rd International Workshop on Machine Learning for Signal Processing (MLSP)*. 2023 IEEE 33rd International Workshop on Machine Learning for Signal Processing (MLSP), pp. 1–6. DOI: 10.1109/MLSP55844.2023.10285979.

Renz, P., Hochreiter, S., and Klambauer, G. (2019a). "Uncertainty Estimation Methods to Support Decision-Making in Early Phases of Drug Discovery". In: *NeurIPS-2019 Workshop on Safety and Robustness in Decision Making*.

Renz, P., Luukkonen, S., and Klambauer, G. (2024). "Diverse Hits in De Novo Molecule Design: Diversity-Based Comparison of Goal-Directed Generators". In: *J. Chem. Inf. Model.* DOI: 10.1021/acs.jcim.4c00519.

Renz, P., Van Rompaey, D., Wegner, J. K., Hochreiter, S., and Klambauer, G. (2019b). "On Failure Modes in Molecule Generation and Optimization". In: *Drug Discovery Today: Technologies*. Artificial Intelligence 32–33, pp. 55–63. DOI: 10.1016/j.ddtec.2020.09.003.

Rezende, D. J. and Mohamed, S. (2016). "Variational Inference with Normalizing Flows". arXiv: 1505.05770 [cs, stat].

Ruddigkeit, L., van Deursen, R., Blum, L. C., and Reymond, J.-L. (2012). "Enumeration of 166 Billion Organic Small Molecules in the Chemical Universe Database GDB-17". In: *J. Chem. Inf. Model.* 52.11, pp. 2864–2875. DOI: 10.1021/ci300415d.

Sacha, M., Błaż, M., Byrski, P., Włodarczyk-Pruszyński, P., and Jastrzębski, S. (2020). "Molecule Edit Graph Attention Network: Modeling Chemical Reactions as Sequences of Graph Edits". arXiv: 2006.15426 [physics, stat].

Samanta, B., De, A., Jana, G., Chattaraj, P. K., Ganguly, N., and Gomez-Rodriguez, M. (2018). "NeVAE: A Deep Generative Model for Molecular Graphs". arXiv: 1802.05283 [physics, stat].

Sanchez-Lengeling, B. and Aspuru-Guzik, A. (2018). "Inverse Molecular Design Using Machine Learning: Generative Models for Matter Engineering". In: *Science* 361.6400, pp. 360–365. DOI: 10.1126/science.aat2663. pmid: 30049875.

Schneider, G. (2013). *De Novo Molecular Design*. John Wiley & Sons, Ltd. DOI: 10.1002/9783527677016.

Schneider, G. and Fechner, U. (2005). "Computer-Based de Novo Design of Drug-like Molecules". In: *Nat Rev Drug Discov* 4.8 (8), pp. 649–663. DOI: 10.1038/nrd1799.

Schwaller, P., Gaudin, T., Lányi, D., Bekas, C., and Laino, T. (2018). ""Found in Translation": Predicting Outcomes of Complex Organic Chemistry Reactions Using Neural Sequence-to-Sequence Models". In: *Chemical Science* 9.28 (28), pp. 6091–6098. DOI: 10.1039/C8SC02339E.

Schwaller, P., Laino, T., Gaudin, T., Bolgar, P., Hunter, C. A., Bekas, C., and Lee, A. A. (2019). "Molecular Transformer: A Model for Uncertainty-Calibrated Chemical Reaction Prediction". In: *ACS Cent. Sci.* 5.9, pp. 1572–1583. DOI: 10.1021/acscentsci.9b00576.

Segler, M. H. S., Kogej, T., Tyrchan, C., and Waller, M. P. (2018a). "Generating Focused Molecule Libraries for Drug Discovery with Recurrent Neural Networks". In: *ACS Cent. Sci.* 4.1, pp. 120–131. DOI: 10.1021/acscentsci.7b00512.

Segler, M. H. S., Preuss, M., and Waller, M. P. (2018b). "Planning Chemical Syntheses with Deep Neural Networks and Symbolic AI". In: *Nature* 555.7698 (7698), pp. 604–610. DOI: 10.1038/nature25978. arXiv: 1708.04202.

Segler, M. H. S. and Waller, M. P. (2017). "Neural-Symbolic Machine Learning for Retrosynthesis and Reaction Prediction". In: *Chemistry* 23.25 (25), pp. 5966–5971. DOI: 10.1002/chem.201605499.

Seidl, P., Renz, P., Dyubankova, N., Neves, P., Verhoeven, J., Wegner, J. K., Segler, M., Hochreiter, S., and Klambauer, G. (2022). "Improving Few- and Zero-Shot Reaction Template Prediction Using Modern Hopfield Networks". In: *J. Chem. Inf. Model.* 62.9, pp. 2111–2120. DOI: 10.1021/acs.jcim.1c01065.

Shi, C., Xu, M., Guo, H., Zhang, M., and Tang, J. (2020). "A Graph to Graphs Framework for Retrosynthesis Prediction". arXiv: 2003.12725.

Simonovsky, M. and Komodakis, N. (2018). "GraphVAE: Towards Generation of Small Graphs Using Variational Autoencoders". arXiv: 1802.03480 [cs].

Somnath, V. R., Bunne, C., Coley, C. W., Krause, A., and Barzilay, R. (2020). "Learning Graph Models for Template-Free Retrosynthesis". arXiv: 2006.07038 [cs, stat].

Southey, M. W. Y. and Brunavs, M. (2023). "Introduction to Small Molecule Drug Discovery and Preclinical Development". In: *Front. Drug Discov.* 3. DOI: 10.3389/fddsv.2023.1314077.

Stokes, J. M., Yang, K., Swanson, K., Jin, W., Cubillos-Ruiz, A., Donghia, N. M., MacNair, C. R., French, S., Carfrae, L. A., Bloom-Ackermann, Z., Tran, V. M., Chiappino-Pepe, A., Badran, A. H., Andrews, I. W., Chory, E. J., Church, G. M., Brown, E. D., Jaakkola, T. S., Barzilay, R., and Collins, J. J. (2020). "A Deep Learning Approach to Antibiotic Discovery". In: *Cell* 180.4, 688–702.e13. DOI: 10.1016/j.cell.2020.01.021.

Strieth-Kalthoff, F., Szymkuć, S., Molga, K., Aspuru-Guzik, A., Glorius, F., and Grzybowski, B. A. (2024). "Artificial Intelligence for Retrosynthetic Planning Needs Both Data and Expert Knowledge". In: *J. Am. Chem. Soc.* 146.16, pp. 11005–11017. DOI: 10.1021/jacs.4c00338.

Sun, R., Dai, H., Li, L., Kearnes, S., and Dai, B. (2020). "Energy-Based View of Retrosynthesis". arXiv: 2007.13437.

Szegedy, C., Zaremba, W., Sutskever, I., Bruna, J., Erhan, D., Goodfellow, I., and Fergus, R. (2014). "Intriguing Properties of Neural Networks". arXiv: 1312.6199 [cs].

Tang, H., Li, C., Kamei, S., Yamanishi, Y., and Morimoto, Y. (2024). *Molecular Generative Adversarial Network with Multi-Property Optimization*. DOI: 10.48550/arXiv.2404.00081. arXiv: 2404.00081 [cs, q-bio]. Pre-published.

Tetko, I. V., Karpov, P., Van Deursen, R., and Godin, G. (2020). "State-of-the-Art Augmented NLP Transformer Models for Direct and Single-Step Retrosynthesis". In: *Nature Communications* 11.1 (1), p. 5575. DOI: 10.1038/s41467-020-19266-y. arXiv: 2003.02804.

Thomas, M., O'Boyle, N. M., Bender, A., and De Graaf, C. (2022a). *Re-Evaluating Sample Efficiency in de Novo Molecule Generation*. arXiv: 2212.01385 [cs, q-bio]. URL: http://arxiv.org/abs/2212.01385 (visited on 09/04/2023). Pre-published.

Thomas, M., O'Boyle, N. M., Bender, A., and de Graaf, C. (2022b). "Augmented Hill-Climb Increases Reinforcement Learning Efficiency for Language-Based de Novo Molecule Generation". In: *Journal of Cheminformatics* 14.1, p. 68. DOI: 10.1186/s13321-022-00646-z.

Thomas, M., Smith, R. T., O'Boyle, N. M., de Graaf, C., and Bender, A. (2021). "Comparison of Structure- and Ligand-Based Scoring Functions for Deep Generative Models: A GPCR Case Study". In: *Journal of Cheminformatics* 13.1, p. 39. DOI: 10.1186/s13321-021-00516-0.

Turk, J.-A., Gendreau, P., Drizard, N., and Gaston-Mathé, Y. (2022). *A Molecular Assays Simulator to Unravel Predictors Hacking in Goal-Directed Molecular Generations*. preprint. Chemistry. DOI: 10.26434/chemrxiv-2022-dl347.

Vamathevan, J., Clark, D., Czodrowski, P., Dunham, I., Ferran, E., Lee, G., Li, B., Madabhushi, A., Shah, P., Spitzer, M., and Zhao, S. (2019). "Applications of Machine Learning in Drug Discovery and Development". In: *Nat Rev Drug Discov* 18.6, pp. 463–477. DOI: 10.1038/s41573-019-0024-5.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L., and Polosukhin, I. (2017). "Attention Is All You Need". arXiv: 1706.03762 [cs].

Waldman, M., Li, H., and Hassan, M. (2000). "Novel Algorithms for the Optimization of Molecular Diversity of Combinatorial libraries11Color Plates for This Article

Are on Pages 533–536." In: *Journal of Molecular Graphics and Modelling* 18.4, pp. 412–426. DOI: 10.1016/S1093-3263(00)00071-1.

Walters, W. P. (2019). "Virtual Chemical Libraries". In: *J. Med. Chem.* 62.3, pp. 1116–1124. DOI: 10.1021/acs.jmedchem.8b01048.

Weininger, D. (1988). "SMILES, a Chemical Language and Information System". In: *J. Chem. Inf. Comput. Sci.* 28.1, pp. 31–36. DOI: 10.1021/ci00057a005.

Wesolowski, S. S. and Brown, D. G. (2016). "The Strategies and Politics of Successful Design, Make, Test, and Analyze (DMTA) Cycles in Lead Generation". In: *Lead Generation*. John Wiley & Sons, Ltd, pp. 487–512. DOI: 10.1002/9783527677047.ch17.

Williams, R. J. (1992). "Simple Statistical Gradient-Following Algorithms for Connectionist Reinforcement Learning". In: *Mach Learn* 8.3-4, pp. 229–256. DOI: 10.1007/BF00992696.

Winter, R., Montanari, F., Steffen, A., Briem, H., Noé, F., and Clevert, D.-A. (2019). "Efficient Multi-Objective Molecular Optimization in a Continuous Latent Space". In: *Chem. Sci.* 10.34, pp. 8016–8024. DOI: 10.1039/C9SC01928F.

Xie, Y., Shi, C., Zhou, H., Yang, Y., Zhang, W., Yu, Y., and Li, L. (2021). "MARS: Markov Molecular Sampling for Multi-objective Drug Discovery". arXiv: 2103.10432 [cs, q-bio].

Xie, Y., Xu, Z., Ma, J., and Mei, Q. (2023). "How Much Space Has Been Explored? Measuring the Chemical Space Covered by Databases and Machine-Generated Molecules". In: ICLR.

Yan, C., Ding, Q., Zhao, P., Zheng, S., Yang, J., Yu, Y., and Huang, J. (2020). "RetroXpert: Decompose Retrosynthesis Prediction like a Chemist". arXiv: 2011.02893 [cs, q-bio].

Yang, X., Zhang, J., Yoshizoe, K., Terayama, K., and Tsuda, K. (2017). "ChemTS: An Efficient Python Library for de Novo Molecular Generation". In: *Science and Technology of Advanced Materials* 18.1, pp. 972–976. DOI: 10.1080/14686996.2017.1401424. pmid: 29435094.

Yoshikawa, N., Terayama, K., Honma, T., Oono, K., and Tsuda, K. (2018). "Population-Based de Novo Molecule Generation, Using Grammatical Evolution". arXiv: 1804.02134 [physics, q-bio].

You, J., Liu, B., Ying, R., Pande, V., and Leskovec, J. (2019). *Graph Convolutional Policy Network for Goal-Directed Molecular Graph Generation*. DOI: 10.48550/arXiv.1806.02473. arXiv: 1806.02473 [cs, stat]. Pre-published.