De novo drug design using deep generative models for biomedical applications

Dr. Y. Chen[1], Dr. H. Zhao[1], Dr. L. Xu[1], Dr. Q. Wang[1]*

[1] School of Public Health and Preventive Medicine, Zhejiang University School of Medicine, Hangzhou, China

Introduction

Machine learning (ML) and Artificial Intelligence (AI) have had a renaissance during the last few years and have become a hot topic not only in drug discovery but in the whole society. There are many reasons for the comeback: access to larger volume of data through automation, faster computers (i.e. GPUs) and methodological progress within deep learning. Drug discovery has also benefited from these trends and, as shown in this book, ML and AI are becoming much more prominent.[1] Besides impacting areas that has been using ML for many years such as QSAR modelling, completely new areas have opened up with deep learning (DL). One is synthesis prediction, where rule-based methods have been replaced by ML methods.[2,3] But most importantly, an area that has been transformed by DL is molecular *de novo* generation, which will be discussed in this chapter.

The goal with deep-learning-based *de novo* molecular design is to be able to sample the whole chemical space. Estimation on the size of the chemical space vary wildly, but the most common estimate is that it consists of 10^{60} molecules.[4] Irrespectively how large the chemical space is everyone agrees that it is too large to be explicitly enumerated.

Historically, molecular *de novo* design has been done in several ways.[5] Most commonly, when structure or ligand-based constraints are given, molecules can be generated in silico to fulfil them. This can be done using brute-force: fully enumerating a virtual library and scoring each of the compounds on how well they fulfil the constraints. The best scoring molecules are then prioritized for synthesis. These virtual libraries are mainly constructed from in-house or commercially available building blocks and reactions that are assumed to be robust. There have also been efforts to search libraries that are non-enumerable with various search techniques such as genetic algorithms. While many successes using these techniques have been reported in the literature, it is also possible to point out drawbacks.[6] The main difference with deep learning approaches that will be described in this chapter is the lack of prior knowledge of what a drug-like molecule should look like. This concept is neither present in combinatorial enumeration of libraries nor in genetic algorithm type of approaches.

DL-based molecular *de novo* generation has recently been reviewed extensively.[7] The prospective user needs to do several choices on how to generate molecules. A first one is to decide if the generation will be string or graph-based. Another is to decide which architecture to use. A main advantage of DL is that a huge array of architectures can be used. For example, recurrent neural networks (RNN), variational auto-encoders (VAE) or generative adversarial networks (GAN). In this chapter both string-based and graph-based methods are reviewed, and the different DL architectures discussed. With the explosion of articles describing DL-based molecular *de novo* generation in the last 2-3 years there has been an increased awareness that it is necessary to create benchmarks to measure the diversity and the coverage of the chemical space of generated molecules. That is why a large part of the chapter will focus

on discussing the latest developments in benchmarking. It is important that benchmarks both cover explorative aspects, which corresponds to identifying a new chemical series and exploitative aspects which corresponds to optimize a chemical series.

Sequence-based methods for *de novo* generation of small molecules

Sequence-based methods for *in silico* generation of molecules generally uses the Simple molecular line entry system (SMILES) format.[8] The format was originally proposed as a way to describe molecules as strings and uses a sequence of letters to specify elements combined with special characters that enables branching "(" and ")", ring closures ("1", "2", "3", …) and different bond orders ("-", "=" and "#"). Special properties such as charges, isotopes and explicit hydrogens are handled by adding modifiers ("+", "-", "H", etc.) to atoms enclosed in square brackets. The character case can be used to denote aromaticity of a molecule and the format can handle stereochemistry. Together these characters fully describe the molecular structure. **Error! Reference source not found.** A and B illustrates how a molecule is encoded as a SMILES string by choosing a path through the molecule and using branching and ring-closures for non-serial parts.

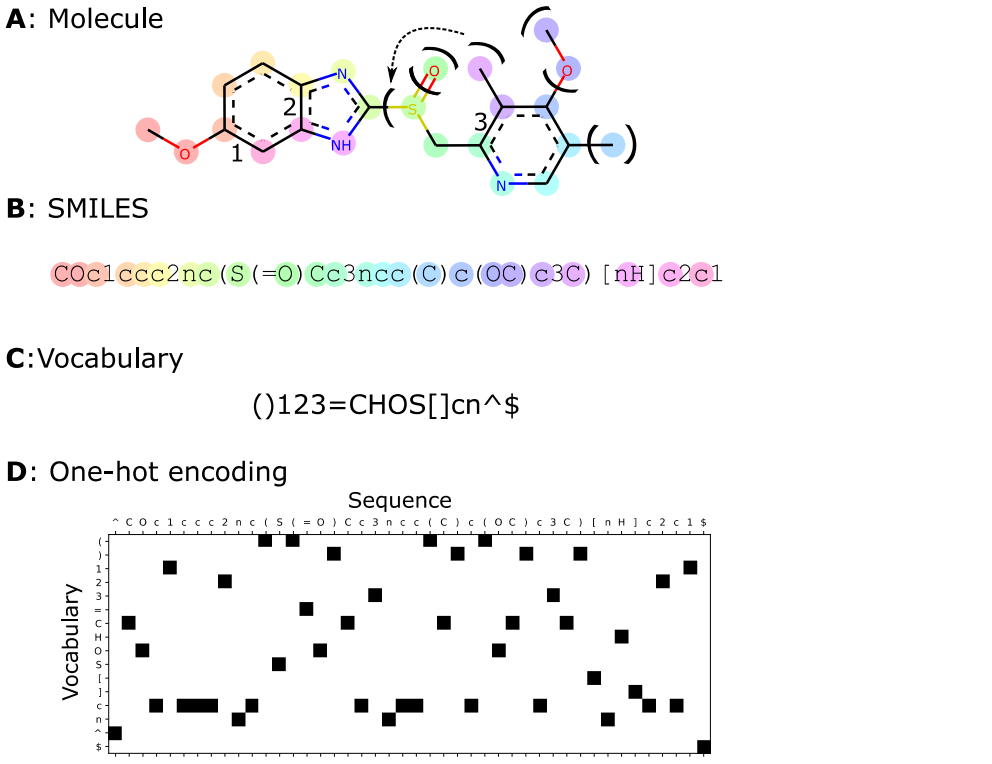

Figure 1: From molecule to one-hot encoded "piano-roll". **A:** The molecule is parsed to a SMILES string by taking a path through the molecule, inserting branches (brackets) and ring-closures (numbers) as needed. A long-range branch is indicated with a dashed arrow. **B:** SMILES representation of the molecule with the same color highlighting as the molecule above. **C:** The vocabulary is here shown simplified as the characters of the SMILES added ^ and $ as start- and end-tokens respectively. It should be derived from all the training data. **D:** In one-hot encoding a bit is set for each character corresponding to the index in the vocabulary.

A molecule can have several different SMILES strings, but one SMILES will always be parsed into the same molecule. The derivation of the SMILES will be dependent on how the path

through the molecular graph is taken. As this is a problem for identification based on the SMILES string, algorithms were quickly developed that canonicalize how it should be derived and ensure a one to one relationship between SMILES and molecule.[9] However, most cheminformatics toolkits do not use the same canonicalization algorithm. This one-to-many relationship between molecules and SMILES strings has been exploited as data augmentation for SMILES-based neural network modelling.[10–14]

De novo generation is not a new field and various methods to generate novel chemical matter *in silico* have been proposed such as fragment combinations in virtual library enumerations and genetic algorithm-based methods.[5] However, the realization that the SMILES syntax could be utilized together with recurrent neural network to generate novel SMILES strings after training the networks token by token have led to a renewed interest in the field.[15–18] The flexibility of the neural network architectures alongside the simplicity of the SMILES syntax have led to many different architectures and solutions.

Embeddings and tokenization

In all the SMILES-based generative architectures, the SMILES string are represented as one-hot encoded vectors, in which each character or token is represented by setting a bit in a vector. The first step is to determine which characters (or tokens) are used as a vocabulary. It is also possible to deconstruct the SMILES into tokens with multiple characters. This enables each atom-type to get its own representation. As an example, the chlorine atom can be represented with the code for "C" followed by the code for "l" or be merged into the code for "Cl", better representing the actual chemistry. The encoded SMILES are usually prepended a "<BEGIN>" token and the end of the sequence appended a "<END>" token. These characters can be anything, given that they are not part of the regular SMILES syntax. Traversing through the SMILES from one end to the other, the one-hot encoded token vectors are concatenated into a two-dimensional array having the tokens along one side and the position in the original SMILES along the other in a piano roll-like format. An example of the encoded format and how it relates to the SMILES string is shown in **Error! Reference source not found.** part D. To enable training on SMILES of different length in the same batch multiple "<END>" tokens can be appended or use SMILES of the same length in each mini-batch.

Recurrent Neural Networks

Recurrent neural networks (RNNs) are a type of neural network that feed their output back to themselves via a recurrent connection spanning a timestep. The influence of previous computations makes possible that an output at a given timestep could be different even though the input is completely the same as in another timestep. This makes these networks ideally suited to work with sequences of arbitrary length and to handle the case where the same input is dependent on the context and should lead to different output. In the context of a SMILES string, an oxygen atom encoded as an "O" could mean different things depending on the surrounding characters, such a double bond "=" or implicit single bond. In the first case, it is much more likely to be terminated with a branch closure ")" or maybe the end token than in the second.

The RNNs performance when handling long range relations are significantly improved with the use of long short-term memory cells (LSTMs)[19] or gated recurrent units (GRUs).[20] These special neural network mini-architectures contain gates that control the flow of information, and

allow the network to store computations for more steps through their hidden state. Thus, the modified RNNs can better fit long-range correlations in the sequences analyzed, which is important when handling SMILES ring closures and closing brackets. As an example, in the SMILES string in **Error! Reference source not found.**, the ring closure pair "1" and "1" spans nearly the whole SMILES and setting a premature "<END>" token during sampling before the last "1" would lead to an invalid SMILES strings. Likewise, the model must keep accounting of opening and closing brackets. Both LSTM and GRU units have been used with success for *de novo* generation,[15–18,21,22] and most deep learning frameworks have the LSTM and GRU cell layers implemented.[23,24] There is a limit to the sequence length that can be handled with LSTM and GRU cells, and it has been shown that temporal convolutional networks (TCNs) can handle certain types of long-range tasks better LSTM.[25] Fortunately, the maximum SMILES length of most drug-like small molecules is well within the length correctly handled by GRU or LSTM. Molecular validity of the generated SMILES is generally higher than 90%, yet a modified SMILES grammar, DeepSMILES,[26] has been proposed to improve these rates. This grammar changes how ring numbering and branching works and claims that it can perform better in deep learning tasks, but a more careful assessment is needed.

RNNs are trained in a sequential fashion, so that the network at each timestep is fed a token and must predict the next one from a probability distribution with the same size as the available characters or tokens (**Error! Reference source not found.**). The result of the training is that the neural network fits a posterior probability distribution of the next character that is sampled. After a start token there are certain tokens or characters that are more common, such as C, N and O, and some that are never seen (e.g. ")", "]" or bonds "=", "#", "–"). These probabilities change depending on the previous sequence and current input. A good way to illustrate this is through heatmaps of the output probabilities as shown in **Error! Reference source not found.**, where it is possible to follow how the model varies the probabilities of the next token or character depending on the sequence of inputs so far.

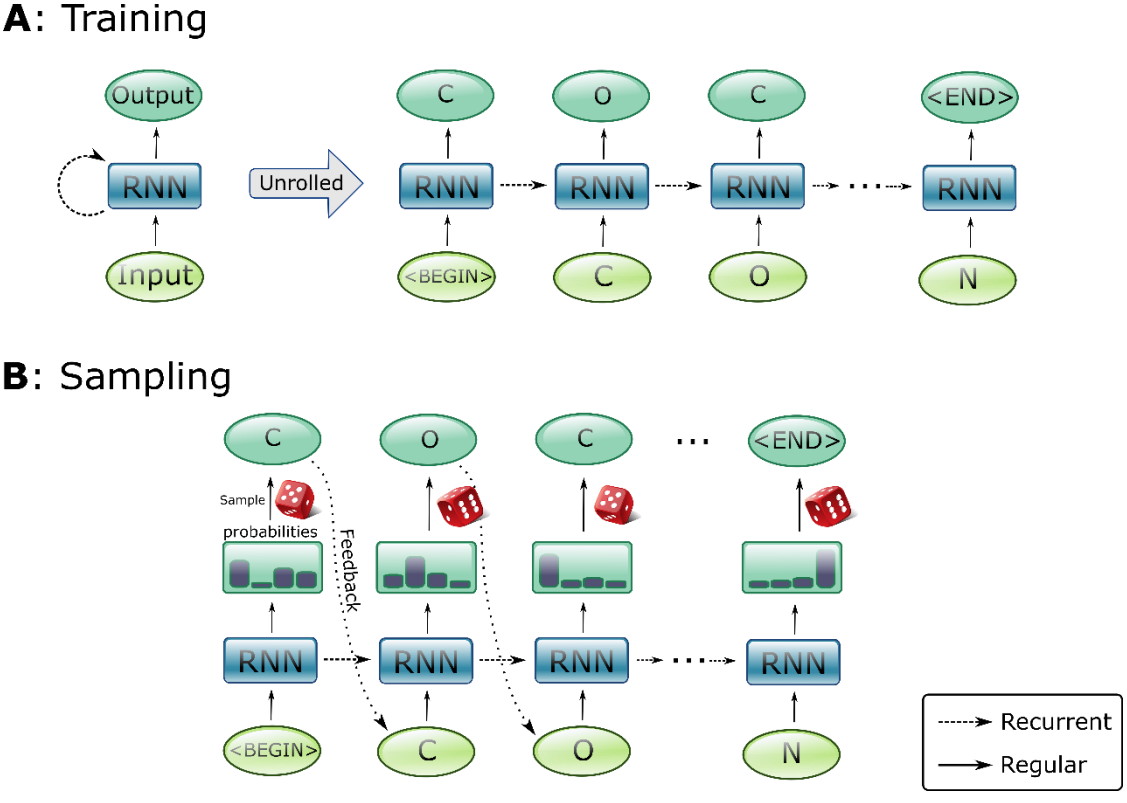

Figure 2: Training and sampling of recurrent neural networks. **A**: **Training phase**. The recurrent neural network is shown simplified on the left and unrolled on the right. The network is trained to predict the next token or character of existing molecules. The target output is simply the input SMILES with an offset of one step (without the "<BEGIN>" token). The RNN is depicted as a single cell but contains multiple cells and multiple connections to all gates. The recurrent connections (dashed) allows information to flow from previous inputs and computations to the next prediction. **B**: **Sampling phase**. It is done one character or token at a time. The learned output probabilities are sampled randomly with multinomial sampling. The choice is recorded and fed back to the network as the next input. Each time the sampling is performed a different SMILES is produced due to the stochasticity of the multinomial sampling.

Sampling SMILES from RNNs

After fitting to a training set, these stepwise output probabilities provide a way to sample new molecules. It is possible to get the next character/token in the sequence by using multinomial sampling where the chance of sampling a token is related to the probability distribution (**Error! Reference source not found.**). After the start token, the chance is biggest for sampling a C (**Error! Reference source not found.**, leftmost column). The sampled character is then fed back as input to the network, giving a probability output for the next character that is then sampled and the result fed back as new input. This process is repeated until the end token is sampled. Due to the stochasticity of the multinomial sampling, the SMILES string produced will not be the same, but will follow the rules for sequence construction extracted from the training set. This works surprisingly well and gives valid SMILES easily. The molecules sampled are similar to, but not the same as in the training set. For metrics and ways to measure the performance of the sampling, see section below on benchmarks and metrics.

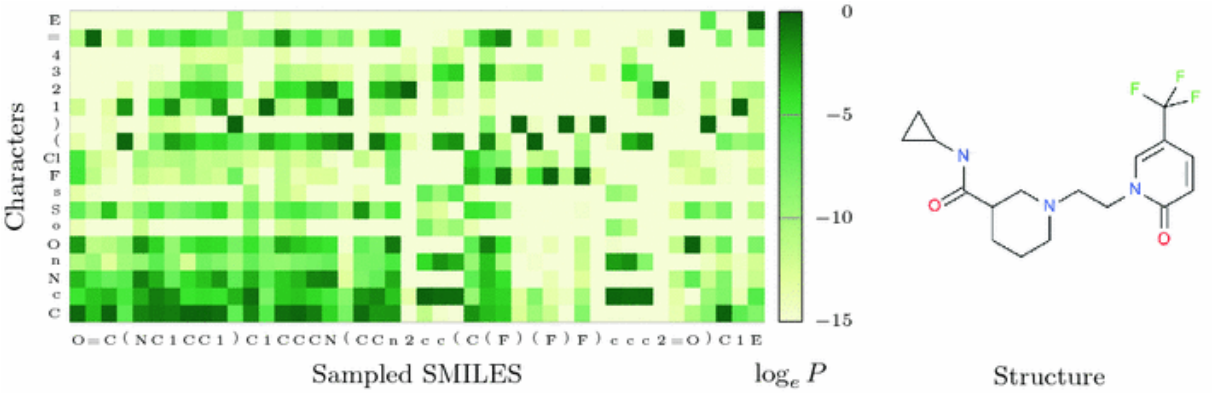

Figure 3: Sampling a molecule one character at a time.[18] The heatmap shows the output probabilities given the sequence up to the point of prediction. The multinomial sampling does not always sample the highest probability, and this will give different molecules for each sampling run, as an example, the token "O" is sampled for the first character although "C" has higher probability (darker green). Areas with less possibilities are found in between areas of larger possible variation. Decoding the CF_3 group seem to be very limited, possibly reflection the fact that CF_3 groups are much more prevalent in training database (ChEMBL) than CF and CF_2 groups.

Temperature scaling can be used to alter the raw, non-scaled probability values (logits). With $temperature > 1$, the lower probabilities are increased and allows the networks to sample more uncommon combinations of tokens with a greater chance. This gives more diversity to the sampled SMILES. However, higher temperatures also leads to an increased rate of invalid SMILES strings that cannot be parsed as molecules.[15] Formally, given the vocabulary V and a temperature γ:

$$\forall v \in V; logit'_v = \frac{logit_v}{\gamma}$$

$$p_v = softmax(logit'_v) = \frac{e^{logit'_v}}{\sum_{w \in V} e^{logit'_w}}$$

Properties and Synthesizability

In general, the distribution of properties follows the prior training set. If fragment-based molecules are used for training, fragment-based molecules will be generated and similar for drug-like molecules.[15] Distributions of measured descriptors also follow the training sets closely. A similar effect happens with synthesizability. The generated molecules are of limited use if the ligand candidates suggested only exists as virtual molecules and cannot be made in the laboratory in a stable form. To solve this problem, surrogate scores for synthetic accessibility (e.g. SAscore[27]) can be employed to asses this aspect and sometimes retrosynthetic evaluation using in-silico tools.[15] The distribution of the SAscores follows the training sets, although the retrosynthetic analysis showed issues for the compounds with the worst SAscore.[15] The synthesis of generated compounds has also been evaluated experimentally,[28,29] albeit with a significant amount of work-up and filtering after the *de novo* generation.

Advanced neural architectures

Gaining better control over the molecular generation has led to a variety of proposed architectures and protocols for controlling and tuning a recurrent neural network. A simple solution is to seed the RNN with a SMILES fragment and let the RNN continue the fragment by sampling.[21,28] This is useful if there is a known core structure crucial for binding, but is limited as it builds up the molecule from only one point of attachment because of the serial nature of SMILES strings. The RNN output "module" described above is either changed post training (**Error! Reference source not found.** top) or the RNN is steered by setting the initial states of the RNN in encoder style networks (**Error! Reference source not found.** bottom). Combinations of the approaches and various modifications to the same basic architectures have led to a large amount of proposed solutions. The lack of agreed standards and metrics make it difficult to compare the approaches based on the publications themselves.

Retuning the generator by transfer learning

The simplest approach to finely-tune the RNN is transfer learning (**Error! Reference source not found.**: 1, A), which uses a large prior dataset of general molecules (e.g. the whole ChEMBL database[30]) to train the RNN the SMILES syntax and chemical rules. After training on the prior, the networks are refocused by training on a smaller, focused dataset. This has shown that an RNN generates increased amounts of actives similar to the focused dataset.[17,21] Learning the general SMILES syntax on a large dataset first gives much better results than training only on the small dataset itself. The approach was experimentally validated with synthesis of five selected molecules after transfer learning on and fragment growing,[28] in which four molecules were found to be active in the target assay. The prior does not necessarily need to be from the same distribution as the set used for transfer learning, considering the results obtained from training a prior from a database of small drug-like fragments (FDB-17[31]) with drug-like molecules.[32]

Reinforcement Learning

In reinforcement learning an agent takes actions that lead to a reward in an iterative process aimed to maximize the rewards. In the *de novo* generation, The RNN is the agent and the generation of molecules the action. The reward is then calculated from a designed score and the output from the RNN is modified through several iterations to optimize this score. The reinforcement learning re-training of the network enables the definition of reward functions that focus on molecular properties, predicted ADME properties, existence of certain motifs and predicted activity in QSAR models, etc. This makes possible getting the RNN to produce molecules with very specific and desirable properties. Reinforcement learning has been used with success to refocus the character-based RNN to generate molecules that better fulfill the criteria of the reward function.[18,33]

Generative adversarial networks

A Generative Adversarial Network (GAN) has two components, a generator and a discriminator, that compete against each other during training. The generator tries to generate new SMILES that are indistinguishable by the discriminator from a pool of SMILES from known compounds. With each round of training the generator becomes better at generating SMILES that follow the properties of the dataset, while at the same time the discriminator learns to discriminate between real or generated SMILES. To get the generator to sample molecules with specific properties, the GAN training is coupled to an auxiliary task where the generator produces SMILES with the desirable properties and the generator is jointly optimized with reinforcement learning during the GAN training. Examples of such implementations are ORGAN[34] and ORGANIC.[35] The former was tested with both molecular generation as well as musical scores, whereas the latter was targeted directly at inverse design of molecules. The latter had trouble optimizing towards the discrete values from Lipinski's Rule of Five[36] heuristic score, but showed some success in optimizing the QED[37] score. GAN + RL-based training was also used in RANC[38] and ATNC[39] where the central RNN was substituted by a differential neural computer (DNC)[40] without and with an adversarial threshold block. The differential neural computer is a more advanced recurrent architecture, where the units have an associated memory block that can be written, accessed and deleted in contrast to the single hidden state of regular GRUs and LSTMs. The authors demonstrated that the DNC-based architectures can handle longer SMILES and have more diversity in the output than the ORGANIC implementation.

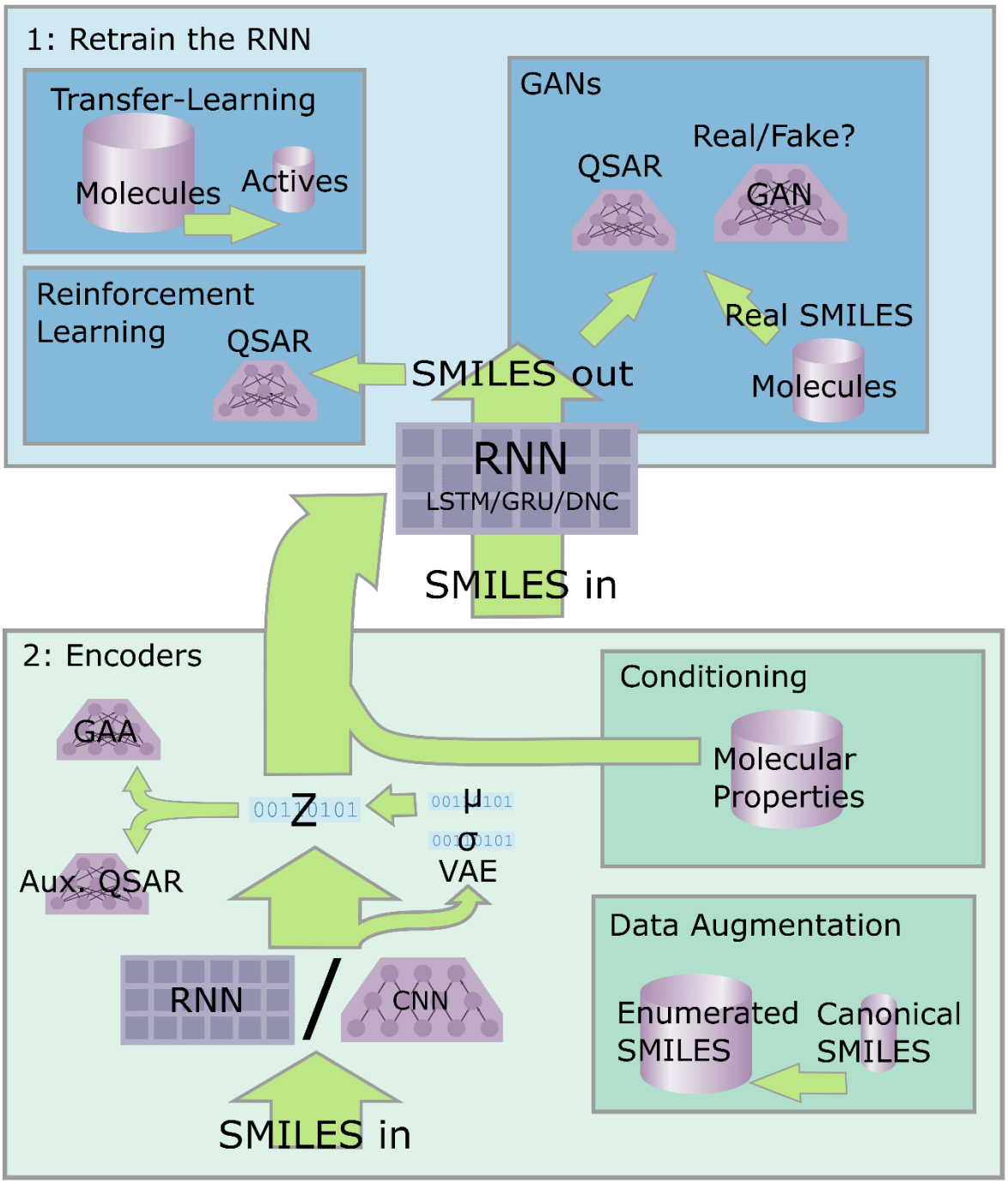

Figure 4: Approaches to adjust and tune the molecular generation. **1:** Manipulate the RNN by retraining on smaller, specific datasets (transfer learning) or directly tune it to output desired molecules (reinforcement learning). **2:** Train the RNN to output molecules indistinguishable from molecules with known desirable properties using the RNN in a GAN. **3:** Manipulate the initial state or generation process itself. Encoder type networks encode the SMILES into a latent code (z) and uses this to start the molecular generation. Z can be sampled from known vectors (μ and σ) in variational encoders (VAE) compared to a wanted distribution (GAA) or coupled to QSAR tasks to disentangle the latent space. **4:** Manipulation of the training data itself with data augmentation or different representations also influences the properties of the RNN.

Encoder-Decoder architectures

An Encoder-Decoder network tries to obtain control of the generative process directly by feeding information to the generator module either by setting the initial states or at each

timestep. One of these architectures, the autoencoder, consists of an encoder – a code layer – and a decoder that are trained together to give the same output as fed into the network. During training the autoencoder will have to produce a representation in the code layer that makes it possible to reconstruct the input with the decoder part. For molecular generation the decoder is usually similar to the character-based RNNs described above where the initial states are set using information from the code layer before sampling begins. The task of the encoder is to convert the SMILES string into a latent vector representation that the RNN can decode to the same SMILES. After training on large molecular databases, the code layer can be used to navigate the "molecular space" of the training set with optimization algorithms, such as Bayesian optimization.[16,22,41,42] With this optimization process, the molecular generation can be directed towards given properties such as cLogP, QED and TPSA. Moreover, the latent space representation of molecules has also been used as descriptors in QSAR models[11,12,43] and represent a sort of automatically engineered fingerprint.

In regular autoencoders, the code layer is restricted in its fitting capacity, which means that one-to-one copies are not possible. This can be achieved through compression to lower number of dimensions (bottlenecks). Alternatively, the code layer can be regularized with L2, dropouts, noise layers, etc.

In variational autoencoders the fitting capacity is lowered by inserting a sampling from learned means and variances. The mean and variance of the code is set by the decoder and used to sample the latent representation before passing the information to the decoder. Further restrictions are imposed on the code-layer. Specifically, the distribution on average should look similar to a uniform Gaussian. To achieve that, the distributions are compared using the Kullback-Leibler divergence (KLD). This is important as the encoder could otherwise set the variance of the latent representation to be very low and thus transfer it unaltered to the decoder. This adds a certain uncertainty to the latent information and prevents the decoder from overfitting. The assumption is that meaningful (and normal distributed) variances are extracted from the data. Additionally, when the latent space dimensionality is too large, there is a risk that the network transfers the information with a selected number of bits with a low variance, but still fulfill the KLD by having bits not used by the decoder output a gaussian.

To further smoothen and disentangle the latent code and introduce *smoothness* and relevance in the code space, the encoder network is often co-trained to predict calculated properties of the molecules from the latent code. In this approach the information from the code layer is not just passed to the decoder, but also to an additional neural network that tries to build a QSAR model of computed properties of the molecules in the training set. The assumption is that the code space itself will be changed to be relevant to both the QSAR task and to the decoding of molecules. This architecture has been used to sample and optimize molecules in the ChemVAE[16], GrammarVAE[22] and SD-VAE.[42] The two later add extra processing on both input and output to pre-process and correct the syntax of the SMILES before training and during sampling.

Adversarial auto encoders (AAEs) also enforce a prior distribution to the latent vector. They do this by coupling the code layer to a discriminator (similar to the GANs) that tries to distinguish the resulting code layers from a defined prior distribution. The random prior distribution is usually uniform or normal. Knowing the distribution that the latent layer resembles enables an easier random sampling of novel points but can include artificial biases in it. After comparing several options, an AAE trained to have a uniform distributed latent

space and an external support vector machine (SVM) model to optimize the molecular generation using an AAE of compounds active against DRD2.[41]

Instead of optimizing the latent space that is fed to the decoder, the calculated or predicted molecular properties can be fed to the autoencoder during training to further condition the decoder. This approach has been used both for an AAE, the entangled conditional AAE architecture (ECAAE)[29] and for a conditional VAE.[44] After training a network with the provided conditions together with the compressed code layer, it can be used to direct the generator to produce compounds with specific properties.

A third interesting option to manipulate the properties of the latent space is to use format translation or heteroencoders. Instead of using autoencoders, where the same SMILES is encoded and decoded to itself, the encoders are trained translate between different formats (e.g. INCHI-to-SMILES, enumerated SMILES to SMILES)[12] or from one example of a non-canonical SMILES to another (enumerated to enumerated SMILES).[11] The fundamental assumption is that by presenting variations of the same underlying latent information (the molecule), the code layer is forced to represent the molecular information in the latent space rather than one representation or another. This training approach was shown to lead to code layers much more relevant in QSAR and QSPR tasks.[11,12]

Using enumerated SMILES in the input shows that the sequence-based encoders handle information in the many-to-one cases. Different non-canonical SMILES of the same molecule encode to similar latent spaces. Moreover, heteroencoders can handle one-to-many cases where the same input information needs to match several outputs. Feeding the decoder from a heteroencoder the same latent space vector several times with subsequent sampling, makes the decoder produce different SMILES strings where the majority was non-canonical versions from the same molecule.[11] On the other hand, using the enumeration approach in the output also introduces an uncertainty and fuzziness in the decoding that can be beneficial or disturbing depending on the intended use-case. This one-to-many capability was also used in a case where reduced graph fingerprints were decoded back to probable molecules using LSTM networks. The reduced graphs identified as useful scaffolds, could be translated back to collections of SMILES for further filtering and processing.[45]

Remapping the latent space with dimensionality reduction has also been demonstrated as an effective way to steer the molecular generation into areas of interest. After mapping active molecules to the reduced space, generative topographic mapping in this case, the coordinates of the active areas are identified. The coordinates of the mapping can afterwards be sampled back to probable latent space vectors and then to molecules using the decoder.[46]

A problem seen with encoder architectures is that points in the latent space may not lead to any molecule. Specifically, the decoder will not be able to decode to a SMILES string that can be parsed into a molecule. The latent space points derived from real molecules in the test-set can have very high SMILES reconstruction accuracy and validity, but suggested vectors during interpolation or optimization can lead to production of a high percentage of invalid SMILES. There is the possibility that trying to impose a continuous representation on fundamentally discrete molecules is not feasible. Adding a penalty term to the optimization if the points suggested do not give valid SMILES string can help mitigating the problem.

Graph-based *de novo* structure generation

The first approaches to deep learning based *de novo* molecule design focused on structure generation using SMILES strings. However, when generating molecules using SMILES intermediate smiles strings cannot be converted to valid molecules , which make it difficult to estimate the intermediate reward for reinforcement learning-based structure generation.[18] To address this issue, methods of generating molecules based on molecular graphs have been proposed and, using this type of methods molecules can be directly generated step-by-step as molecular graphs. One of the first proposed approaches was the GGT-NN (Gated Graph Transformation Neural Network), which modifies a graph-based on a sequence of input sentences.[47] This idea was further refined by considering molecule structure generation as a sequence of graph transformations (Figure 5).[48] Specifically, a generative model was used to predict if an atom should be added or a bond should be formed among existing atoms.

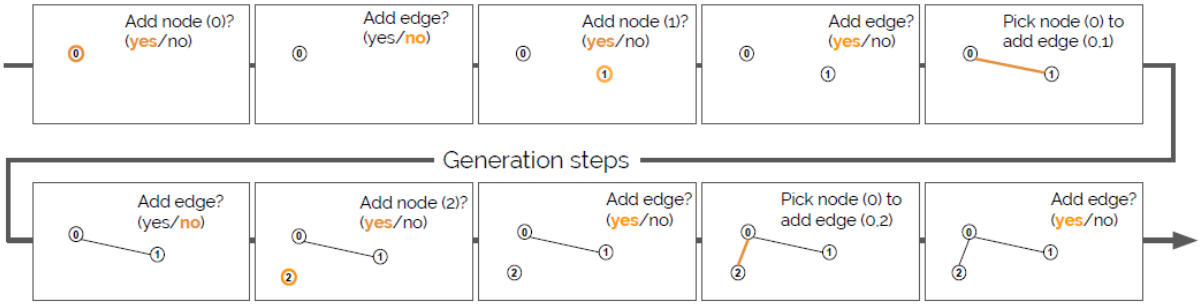

Figure 5: The demonstration of the action sequences in the graph-based structure generation.[48]

The algorithm is described as such: For a graph $G = (V, E)$, for each node an embedding vector is generated. These embedding vectors are computed initially from node features and edge features, and then propagated on the graph to aggregate information from the local neighborhood. A GRU NN was used to generate the node embeddings. The graph generative model defines a policy distribution over the sequence of graph generating decisions (Figure 5) and the action for each step is taken based on the NN predicted probability distribution over all possible actions. Several types of action are considered in the algorithm, which includes adding new atoms, forming new bond among the new atom and existing atoms and the structure termination. The probability distribution of these types of actions are predicted by individual NNs with the given molecular graph as input and these NNs are trained together to learn the generative model for molecular generation.

Another graph-based structure creation method based on the previous architecture is MolMP.[49] Three types of action are defined for structure generation (Figure 6): (1) Append, adding a specific new atom to the graph and forming certain bond types; (2) Connect, forming intra-graph bond between a pair of existing atoms in the graph; (3) Stop, the generation process stopped. The molecule generation process can be treated as a Markov decision process (MDP) or have a recurrence in the generative model. For the MDP approach, a model is trained to predict the policy of making one action at each step by stacking together several graph convolution NNs and feed forward NNs. Introducing recurrence in the network is done by adding a RNN that kept the information of the previous steps. Results showed that using a RNN improved the physicochemical properties of generated structures.

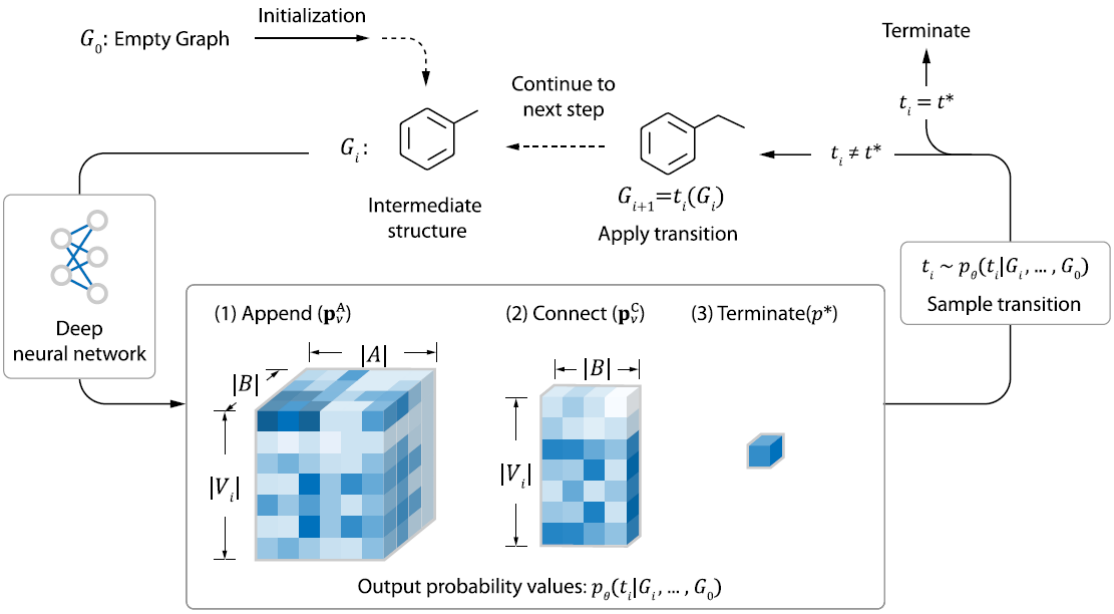

Figure 6: The representation of the structure generation process proposed by Li et al.[49] The intermediate graph is used as the input to the graph convolution neural networks to predict the probability distribution (i.e. policy prediction) among all possible actions. Once the probability distribution is generated, next action is taken by sampling the policy.

Graph convolutions[50] can also be used successfully to generate molecules.[51] A graph convolutional policy network is firstly used to calculate the node embedding vectors of the intermediate graph. Then, four different link prediction-based actions are decided using a generative model: (1) Selection of first atom; (2) Selection of second atom; (3) Bond connect between two selected atoms; (4) Structure termination. After a structure generative model is trained, the adversarial-based reinforcement learning is performed to generate structures satisfying certain objectives. Results show that GCPN can achieve 61% improvement on chemical property optimization over state-of-the-art baselines while resembling known molecules and achieve 184% improvement on the constrained property optimization tasks.

The SMILES-based variational autoencoder methods in general suffer from low validity of generated structures due to the inability of learning smooth molecular embedding. A graph-based autoencoder method, Junction tree neural network (JTNN), was proposed to address this issue.[52] Specifically, the pharmacophore definition used in feature tree method[53] can be employed to first convert molecule structures to junction trees. In these, each node represents a cluster of atoms and there are no cycles. As it can be seen in Figure 7, two different message passing neural networks (MPNN)[54] are used to encode the molecular graph (graph embedding vector Z_G) and its respective junction trees (tree embedding vector Z_t) separately. The tree embedding vector is then decoded to recover the junction tree and, in the end, using together graph embedding Z_G and the decoded junction tree, the molecule graph is decoded to reproduce the original input structure. Then, the tree decoder and graph decoder are trained to achieve maximal likelihood of tree structure given Z_t and predicting correct subgraphs G_i of the ground true graph G at each tree node. In the experiment, their method is able to generate 100% valid molecule structures which significantly improves the results of SMILES-based VAE methods. This is largely due to the usage of predefined chemical substructures in the junction tree which helps focusing the neural network in learning the connectivity among chemical substructures instead of the substructures themselves.

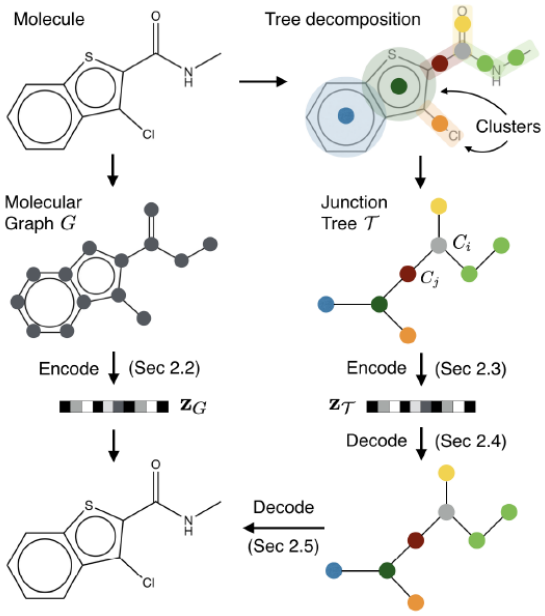

Figure 7: The scheme of the junction tree autoencoder method.

Generative Adversarial Networks (GANs)[55] have been used in graph-based molecular generation. One of the first approaches to appear was MolGAN[56], which behaves in a similar way as the sequence-based models. In detail, the generator is a feed-forward neural network that takes a multi-dimensional vector sampled from a standard normal distribution as input and has two outputs: an atom feature matrix and a 3D adjacency matrix representing the connectivity among the atoms across the different atom types. A softmax function is applied on the output values to form the probability distribution for the atom type and connectivity at each node of the graph. A categorical sampling is then carried out to obtain a graph. The generated graphs are then sent to the discriminator to see if the discriminator can distinguish them from the real graph data. Besides the normal GAN loss, a reinforcement learning related reward is fed back to the generator to optimize the generated structures toward pre-defined molecular properties. This method does not rely on making a sequence of action to generate the graph and performs faster than other approaches, but it can only generate graphs with fixed molecular size.

Graph-based generative models are a new frontier in *de novo* molecular design. Comparing them with SMILES-based methods, they enable of injecting chemistry directly into the structure generation process. On the other hand, they generally perform slower than the SMILES based methods. This is because molecules need to be broken down into many subgraphs, thus increasing the size of training samples. We believe that graph-based *de novo* design method is still at its infancy, yet it would be interesting to have some research comparing the structures generated by SMILES and graph methods in the future.

Benchmarking generative molecular *de novo* design models

One of the main challenges when working with generative models is trying to assess whether a model has been trained correctly for the task it has been created for. A benchmark, comprised of many complementary metrics, helps assessing the properties of the chemical space sampled and the suitability of the generated molecules given the challenges being addressed. Moreover, it can be used to optimize the training of a model or to compare

different generative architectures. Unfortunately, the metrics used in benchmarks for generative models of different content types (i.e. image, audio, video) generally cannot be directly applied to molecule generation for two reasons. Firstly, each content type has completely different ways of assessing its quality and secondly, the content generated by any given model affects its domain in a unique way. For example, evaluating correctness in continuous formats such as images or audio is difficult: an image can be mostly correct, but this is not the case of molecules: either have correct valency in all atoms or not. In addition, a careful choice of metrics must be made to prevent ambiguity: if we compare a metric that evaluates the correctness of molecules generated with another that checks its diversity (i.e. how different sampled molecules are from those in the training set), both metrics are independent between each other, because non-diverse sets can perfectly have high percent of valid molecules, such as the case of a model generating mostly valid molecules from the training set. Furthermore, all metrics can be labelled depending on the type of model that they are benchmarking. Models range between *explorative* and *exploitative*. Explorative models focus on generating as much chemical space as possible within the limitations of the training set a typical example is scaffold hopping, where a new chemical series needs to be identified. Exploitative models focus on obtaining novel molecules with specific structural or physicochemical properties, which correspond to optimization of a chemical series. Therefore, any benchmark must include a careful selection of metrics that is able to holistically characterize the generative model and account for all possible shortcomings.

Benchmarking explorative models

The chemical space explored by a generative model, or domain, is comprised of a set of molecules, each of which having a probability of being sampled. It has been shown that generative models whose generated molecule probability distribution tends to be uniform (i.e. all molecules have the same probability of being sampled) maximizes the sampleable chemical space.[57] This is important for explorative models since their main objective is to maximize the chemical space generated. That is why metrics that measure the breadth and the diversity of the chemical space generated are of great importance. Until late 2018, publications describing new explorative generative model architectures used a diverse set of metrics to characterize the models. They can be summarized in the following list:

1. **Validity** (fraction):[11,16–18,21,33,38,39,41,48–52,56,58–60] Fraction of sampled molecules that are valid according to a given chemistry software library. Note that some toolkits (i.e. RDKit) fix and alter some otherwise invalid SMILES.
2. **Uniqueness** (fraction):[18,33,38,39,49,56,59,60] Fraction of sampled molecules that are valid and different to each other. This uniqueness can be at a SMILES level (for models that train with SMILES) or at a molecular level, given the multiple SMILES representations for each molecule.
3. **Diversity** (fraction):[29,33,38,41,48,49,56,59–61] Fraction of valid sampled molecules that are not in the training set. This metric can compare to the exact molecules in the training set (i.e. the canonical SMILES) or it can otherwise use some similarity threshold.
4. **Drug-likeness** (QED) (scalar):[16,34,35,38,49] Fraction of molecules that resemble drug-like molecules. It is obtained by aggregating different descriptors (e.g. logP, HBD, HBA, MW, etc.) in a single score. We would advise caution when using this metric, as the applicability domain may be limited. Moreover, the metric contains no stability assessment, so unstable molecules with good QED scores may be generated.

5. **Synthetic availability score** (SAS) (scalar):[15,16,33,34,58,62] Score that predicts the potential synthesizability of molecules. It is obtained by enumerating the fragments of a molecule and checking if these fragments are common in PubChem molecules. After that a complexity score is added to penalize difficult features such as macrocycles or bridge atoms. This score has the same problems as QED, given that its applicability domain may be limited.
6. **Loss** (scalar):[15,21,61] The training process of a generative model is based on minimizing the loss function. The lower the loss function is, the more the sampled molecules will resemble those in the training set. This can evolve into overfitting problems.
7. **Descriptors** (scalar):[15,16,18,21,34,38,39,49,56,58,62] Descriptors such as clogP, molecular weight, number of rotatable bonds, etc. are calculated and the distribution is compared to the training set distribution either by mean ± standard deviation or using more sophisticated methods like the KLD.
8. **Plot representations**:[11,16,17,21] Some descriptors (or fingerprints) are calculated from sampled molecules and from molecules from the training set. Optionally, a dimensionality reduction method such as Principal Component Analysis (PCA) or t-Distributed Stochastic Neighbor Embedding (t-SNE) is used and the first pair of dimensions are plotted. This gives a visual representation of how the generated chemical space differs from the training set.

There is not a general rule on which metrics are used in each publication and it is problematic, as some of the published architectures do not give a holistic overview on the domain of the model. As it can be seen in Table 1, publications that describe new explorative model architectures generally use validity, descriptors, diversity and uniqueness. Except for descriptors, these metrics are generally the best ones to assess the quality of any explorative model, as they can measure the size of the domain. The other metrics are more specific to each architecture and they must be used carefully because they can add substantial bias to the benchmarking. This must be specially noted for QED and SAS scores, given that their applicability domain is not known, and they may output invalid scores for some molecules.

	Total	Val.	Uniq.	Div.	QED	SAS	Loss	Desc.	Plots
SMILES-based RNN	6	4	2	1	0	3	2	4	2
VAEs	8	7	2	4	1	2	0	2	2
Graph-based	3	3	1	2	1	0	0	1	0
GANs	6	3	3	3	3	1	1	4	0
Total	23	17	8	10	5	5	3	11	4

Table 1: From a selected group of 23 publications of molecular generative models grouped by the architecture, the number of publications that use each explorative metric in benchmarks. Note that no metric is used throughout all published architectures.

Recently, a novel metric appeared that includes information from many of the basic metrics in a standardized way. The Fréchet ChemNet Distance (FCD)[63] requires two molecule sets: a set of real-world molecules and a sample from the model. These sets are discretized by forward passing all the molecules in these sets up to the penultimate activation layer of ChemNet,[64] a bioactivity prediction model. Then, assuming that the distribution obtained for each set is a multidimensional gaussian, the Fréchet Distance is calculated between them. If the trained generative model can create molecules with similar chemical properties than the

real-world molecules, the distance will be low. The researchers also showed that this distance is sensitive to changes on many of the widely known metrics, such as descriptors, diversity and drug-likeness. Nonetheless, there is no way of assessing which are the differences between both sets or what is the magnitude of that difference. Additionally, because the ChemNet was trained with a set of known drug-like molecules the domain of applicability of the FCD may be restricted to the known chemical space. This could imply that the metric is not suited for benchmarking models that explore novel regions of the chemical space.

Benchmarking exploitative models

Exploitative models focus on exploring a specific region of the chemical space. This is done by either including in the training process a score formed by a set of descriptors (reinforcement learning, loss function, etc.) or by using focused training sets (transfer learning, etc.). To benchmark these models some of the metrics used in explorative models, such as validity, drug-likeness, descriptors, etc., can be used. Although more specific metrics that better fit the optimization problem are generally preferred. A list with some of the most common used in literature follows:

1) **Similarity** (scalar):[17,18,21,41,49,51,52,58] Fingerprint (e.g. ECFP4, MACCS, etc.) similarity of all generated molecules to a known molecule or to each other. This metric can be used to assess whether the generated focused chemical space is dense or sparse. Depending on the optimization problem this metric is important, as it is able to show the amount of novelty.
2) **Discovery** (scalar):[16,18,33,41,58,60] Measuring the time (training epochs, distance in latent space, number of iteration in Bayesian Optimization, etc.) that takes a model to be able to sample a molecule not present in the training set. This metric is especially used to benchmark VAEs and models that use reinforcement learning.
3) **Plot representations**:[17,21,33,49] Using the same method as in the explorative method but this time plotting the sampled molecules with real molecules of interest. For example, plotting known inhibitors of a target with the molecules generated.
4) **Descriptors** (scalar):[16–18,33–35,49,51,52,56,58] Calculate descriptors on the sampled molecules. This metric is mainly used when a model is optimized to obtain molecules with given structural or physicochemical properties.
5) **Enrichment over Random** (EOR) (scalar):[17,49] This metric allows to quantify the improvement of a focused model compared to a non-focused one. It is calculated as the ratio of molecules in a training set T on the focused set sampled molecules compared to the molecules found on a non-focused set. It is exclusively used when comparing explorative with exploitative models, such as those trained with transfer or reinforcement learning.
6) **Target prediction models** (TPM) (fraction):[17,18,29,33,49] Using TPMs or similar to predict whether generated molecules are active on a given set of targets. This metric should be used with care, as the TPM should be trained with molecules outside of the training set, to minimize bias.

As it can be seen in Table 2, the most common metric used are descriptor-based statistics. This makes sense because most of the optimization problems in published literature are about obtaining molecules with optimized properties. The second most common metric is similarity, which assesses if the generated chemical space is diverse enough. It is surprising that no

published GAN model uses similarity metrics, and it may be to the specificities of the architecture.

	Total	Sim.	Disc.	Plots	Desc.	EOR	TPM
SMILES-based RNN	6	3	2	3	3	1	3
VAEs	8	3	4	0	3	0	1
Graph-based	3	2	0	1	2	1	1
GANs	6	0	0	0	3	0	0
Total	23	8	6	4	11	2	5

Table 2: From a selected group of 23 publications of molecular generative models grouped by the architecture, the number of publications that use each exploitative metric in the benchmarking section. Note, that as in the previous table, no metric is used throughout.

Benchmarking models during training

One of the main uses of benchmarking is to optimize the training process of a given model architecture. Models with different hyperparameter configurations are trained and after each training epoch some metrics can be calculated and used to assess the quality of the models. Any of the metrics of the previous sections can be used to assess the quality of the half-trained models, although the metrics should not be computationally expensive to calculate. Most published models do not go into details on which metrics were used when training. It is important to notice that in generative models we aim to maximize the likelihood of sampling molecules from the training set, but without controlling the diversity it may yield to models that output mostly molecules similar to the training set.

A new benchmarking method has been published[57] that trains models that sample as much as possible the GDB-13 database and describes an approach to training RNN/Graph/GAN-based models which can be applied to any other chemical space. Two sets of molecules are obtained beforehand: a training set and a validation set, which include molecules from GDB-13 but not part of the training set. After each epoch, the negative log-likelihood (NLL) distributions of the molecules is calculated from a sample, the training set and the validation set. If the generated molecules from the model have the same probability of being generated than those from the training set and the validation set, means that the domain of the model is bigger. To assess that, the three Jensen-Shannon Divergences (JSD) between each pair of distributions are obtained and plotted for all epochs. Figure 1a is an example of a model trained with the 1 million molecules obtained randomly from GDB-13. See that the three distances are minimal around epoch 70-100, meaning that generating a molecule from the training set has the same probability as generating a molecule outside of it. Afterwards, the likelihood of training molecules in the training set is higher than from outside, meaning that the model is overfitting. See that in Figure 1b the fraction of valid molecules is impervious to this change, as it is always increasing. This happens because when the model is overfit, it becomes more conservative and creates more molecules that are similar to those in the training set.

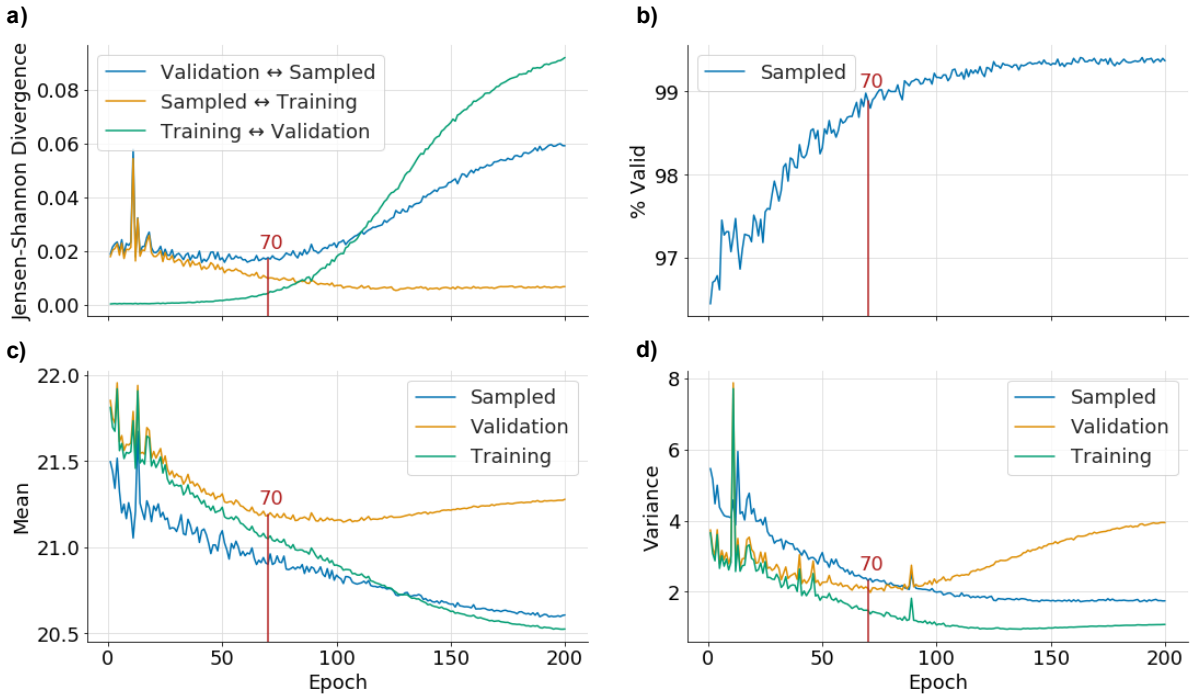

Figure 8: Metrics used to evaluate the training process of an RNN+SMILES molecular generative model trained with 1 million molecules from GDB-13.[57] The red line at epoch 70 represents the chosen epoch in the end. **a)** JSD plot between the three NLL distributions for each of the 200 epochs. **b)** Percentage of valid molecules in each epoch. Notice that the plot already starts at around 96,5%. **c-d)** Mean (c) and variance (d) of the three distributions. Note that spikes around epochs 1-20 are statistical fluctuations common in the beginning of the training process of a RNN, when the learning rate is high.

The same happens when the variance of the NLL distributions of the three sets is plotted. A low variance of the validation set NLL distribution implies that the generated molecule probability distribution is more uniform throughout the entire chemical space generated by the model. This means that every time the model is sampled, there will be more diversity. In Figure 1c the variance of the validation set NLL distribution is also minimal around epochs 60-90. Also note that in Figure 1d, the loss function (NLL average) is also plotted and see that, even though the model is overfitting, the sampled set loss function continues to decrease. With this approach three different training stages were assessed for each model: a learning phase, followed by an optimally trained phase to then move to overfitting. The only downside of this method is that it is sensitive to noise, since it uses small samples to generate the plots each iteration.

Comparing model architectures

In the recent years a wealth of molecular generative model architectures has been published. Unfortunately, the lack of a consistent and standardized set of metrics has made the process of comparing the improvements between them unreliable. One of the main issues is that there are neither datasets nor case problems to use when performing benchmarks. Moreover, some metrics have different meaning, or even are not applicable, depending on the generative model architecture benchmarked. Three manuscripts have been published[57,65,66] that offer a set of standardized methods that try to alleviate some of the current limitations.

A first approach, more akin to how image predictive models are benchmarked,[67] was suggested.[57] The objective of the benchmark is to train a model with a small subset of GDB-13

and use it to generate as much of it as possible sampling the model a specific number of times. The manuscript shows that a uniform and complete model, which samples uniformly molecules from and only from GDB-13, is the best possible. From this an upper bound can be obtained for any sample size: for example, sampling 2 billion molecules from the uniform model would yield on average 87.12% of the complete database. Different generative models can be trained with any combination of hyperparameters and the unique percent of molecules generated from GDB-13 obtained. This gives a measure of the learning capability of a molecular generative model.

The second one, GuacaMol,[65] focuses on describing a range of metrics and tests that evaluate different characteristics of generative models. Some of the metrics have been also described in this section, and include validity, uniqueness, diversity, similarity, FCD, discovery, QED and descriptors. It also includes other interesting metrics, such as generating all possible isomers from a given molecular formula and a multi-objective optimization problem. A software implementation was also released alongside the manuscript.

The last manuscript, MOSES,[66] was made public and describes a more constrained benchmark that measures fragment, scaffold and nearest neighbor similarity, diversity, FCD and descriptors. Another important difference with GuacaMol is that it recommends using a specific subset of molecules to train from obtained from the ZINC Clean Leads database.[68] Also, a software implementation is available.

Conclusions

DL-based molecular *de novo* generation has emerged during the last three years as one of the most interesting and fast-moving fields in cheminformatics. In this chapter different molecular representations (strings and graphs) and different architectures such as RNNs, VAEs and GANs have been described. The openness in terms of using open source (e.g. RDKit[69], TensorFlow[23] and PyTorch[24]), public databases (e.g. ChEMBL[30]) and readily preprint publication on preprint servers such as *arXiv* (https://arxiv.org/) and *ChemRxiv* (https://chemrxiv.org/) have contributed to the remarkable progress that has been seen in the field during the last 2-3 years. However, this *Cambrian explosion* of methods has made it necessary to start focusing on benchmarks to assess how well different methods work. The benchmark criteria used so far are extensively reviewed in this chapter. Several different architectures can generate drug-like molecules and so far, no architecture has been shown to be superior to others in all metrics. It might very well be that different architectures will be superior under different circumstances. It can also be expected that new architectures might be used in the future.

Although there has been considerable progress in generating molecules covering the whole chemical space, much less has been in scoring the ideas. While experimental validation of DL-based molecular *de novo* generation has been published,[28] it is still too early to fully assess the potential impact on drug design. However, we think that the full potential of DL-based molecular *de novo* design will most likely be in conjunction with synthesis prediction and automation.[70] The potential speed-up of the *design-make-test* cycle through applying AI together with automation has the potential to enhance the drug design process. Creating much faster high-quality tool compounds to validate targets *in vivo* will surely have a transformative impact on the drug discovery process.

Acknowledgement

The project leading to this article has received funding from the European Union's Horizon 2020 research and innovation program under the Marie Skłodowska-Curie grant agreement No 676434, "Big Data in Chemistry" ("BIGCHEM", http://bigchem.eu). The article reflects only the authors' view and neither the European Commission nor the Research Executive Agency are responsible for any use that may be made of the information it contains.

References

(1) Chen, H.; Engkvist, O.; Wang, Y.; Olivecrona, M.; Blaschke, T. The Rise of Deep Learning in Drug Discovery. *Drug Discov. Today* **2018**, *23* (6), 1241–1250. https://doi.org/10.1016/J.DRUDIS.2018.01.039.

(2) Engkvist, O.; Norrby, P.-O.; Selmi, N.; Lam, Y.; Peng, Z.; Sherer, E. C.; Amberg, W.; Erhard, T.; Smyth, L. A. Computational Prediction of Chemical Reactions: Current Status and Outlook. *Drug Discov. Today* **2018**, *23* (6), 1203–1218. https://doi.org/10.1016/J.DRUDIS.2018.02.014.

(3) Segler, M. H. S.; Preuss, M.; Waller, M. P. Planning Chemical Syntheses with Deep Neural Networks and Symbolic AI. *Nature* **2018**, *555* (7698), 604–610. https://doi.org/10.1038/nature25978.

(4) Bohacek, R. S.; McMartin, C.; Guida, W. C. The Art and Practice of Structure-Based Drug Design: A Molecular Modeling Perspective. *Med. Res. Rev.* **1996**, *16* (1), 3–50. https://doi.org/10.1002/(SICI)1098-1128(199601)16:1<3::AID-MED1>3.0.CO;2-6.

(5) Schneider, G.; Fechner, U. Computer-Based de Novo Design of Drug-like Molecules. *Nat. Rev. Drug Discov.* **2005**, *4* (8), 649–663. https://doi.org/10.1038/nrd1799.

(6) Schneider, G.; Lee, M. L.; Stahl, M.; Schneider, P. De Novo Design of Molecular Architectures by Evolutionary Assembly of Drug-Derived Building Blocks. *J. Comput. Aided. Mol. Des.* **2000**, *14* (5), 487–494. https://doi.org/10.1023/A:1008184403558.

(7) Xu, Y.; Lin, K.; Wang, S.; Wang, L.; Cai, C.; Song, C.; Lai, L.; Pei, J. Deep Learning for Molecular Generation. *Future Med. Chem.* **2019**, fmc-2018-0358. https://doi.org/10.4155/fmc-2018-0358.

(8) Weininger, D. SMILES, a Chemical Language and Information System: 1: Introduction to Methodology and Encoding Rules. *J. Chem. Inf. Comput. Sci.* **1988**, *28* (1), 31–36. https://doi.org/10.1021/ci00057a005.

(9) Weininger, D.; Weininger, A.; Weininger, J. L. SMILES. 2. Algorithm for Generation of Unique SMILES Notation. *J. Chem. Inf. Comput. Sci.* **1989**, *29* (2), 97–101. https://doi.org/10.1021/ci00062a008.

(10) Bjerrum, E. J. SMILES Enumeration as Data Augmentation for Neural Network Modeling of Molecules. *ArXiv* **2017**.

(11) Bjerrum, E. J.; Sattarov, B. Improving Chemical Autoencoder Latent Space and Molecular De Novo Generation Diversity with Heteroencoders. *Biomolecules* **2018**, *8* (4), 1–17. https://doi.org/10.3390/biom8040131.

(12) Winter, R.; Montanari, F.; Noé, F.; Clevert, D.-A. Learning Continuous and Data-Driven Molecular Descriptors by Translating Equivalent Chemical Representations. *Chem. Sci.* **2018**, *10* (6), 1692–1701. https://doi.org/10.1039/c8sc04175j.

(13) 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* (28), 6091–6098. https://doi.org/10.1039/c8sc02339e.

(14) Kimber, T. B.; Engelke, S.; Tetko, I. V.; Bruno, E.; Godin, G. Synergy Effect between Convolutional Neural Networks and the Multiplicity of SMILES for Improvement of Molecular Prediction. **2018**. https://doi.org/arXiv:1812.04439v1.

(15) Bjerrum, E. J.; Threlfall, R. Molecular Generation with Recurrent Neural Networks (RNNs). **2017**.

(16) 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.; Aspuru-Guzik, A. Automatic Chemical Design Using a Data-Driven Continuous Representation of Molecules. *ACS Cent. Sci.* **2018**, *4* (2), 268–276. https://doi.org/10.1021/acscentsci.7b00572.

(17) 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* (1), 120–131. https://doi.org/10.1021/acscentsci.7b00512.

(18) Olivecrona, M.; Blaschke, T.; Engkvist, O.; Chen, H. Molecular De-Novo Design through Deep Reinforcement Learning. *J. Cheminform.* **2017**, *9* (1), 48. https://doi.org/10.1186/s13321-017-0235-x.

(19) Hochreiter, S.; Schmidhuber, J. Long Short-Term Memory. *Neural Comput.* **1997**, *9* (8), 1735–1780. https://doi.org/10.1162/neco.1997.9.8.1735.

(20) Cho, K.; van Merrienboer, B.; Gulcehre, C.; Bahdanau, D.; Bougares, F.; Schwenk, H.; Bengio, Y. Learning Phrase Representations Using RNN Encoder-Decoder for Statistical Machine Translation. **2014**. https://doi.org/10.3115/v1/D14-1179.

(21) Gupta, A.; Müller, A. T.; Huisman, B. J. H.; Fuchs, J. A.; Schneider, P.; Schneider, G. Generative Recurrent Networks for De Novo Drug Design. *Mol. Inform.* **2018**, *37* (1), 1700111. https://doi.org/10.1002/minf.201700111.

(22) Kusner, M. J.; Paige, B.; Hernández-Lobato, J. M. Grammar Variational Autoencoder. **2017**. https://doi.org/10.1134/S1063785008050192.

(23) Abadi, M.; Agarwal, A.; Barham, P.; Brevdo, E.; Chen, Z.; Citro, C.; Corrado, G. S.; Davis, A.; Dean, J.; Devin, M.; et al. *TensorFlow: Large-Scale Machine Learning on Heterogeneous Distributed Systems*; 2016. https://doi.org/10.1109/TIP.2003.819861.

(24) Paszke, A.; Chanan, G.; Lin, Z.; Gross, S.; Yang, E.; Antiga, L.; Devito, Z. Automatic Differentiation in PyTorch. *Adv. Neural Inf. Process. Syst. 30* **2017**, No. Nips, 1–4.

(25) Bai, S.; Kolter, J. Z.; Koltun, V. An Empirical Evaluation of Generic Convolutional and Recurrent Networks for Sequence Modeling. **2018**. https://doi.org/10.1016/S0925-

5273(03)00047-1.

(26) O'Boyle, N.; Dalke, A. DeepSMILES: An Adaptation of SMILES for Use in Machine-Learning of Chemical Structures. *chemRxiv: 10.26434* **2018**. https://doi.org/10.26434/chemrxiv.7097960.v1.

(27) 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. https://doi.org/10.1186/1758-2946-1-8.

(28) Merk, D.; Friedrich, L.; Grisoni, F.; Schneider, G. De Novo Design of Bioactive Small Molecules by Artificial Intelligence. *Mol. Inform.* **2018**, *37* (1). https://doi.org/10.1002/minf.201700153.

(29) Polykovskiy, D.; Zhebrak, A.; Vetrov, D.; Ivanenkov, Y.; Aladinskiy, V.; Mamoshina, P.; Bozdaganyan, M.; Aliper, A.; Zhavoronkov, A.; Kadurin, A. Entangled Conditional Adversarial Autoencoder for de Novo Drug Discovery. *Mol. Pharm.* **2018**, *15* (10), 4398–4405. https://doi.org/10.1021/acs.molpharmaceut.8b00839.

(30) Gaulton, A.; Hersey, A.; Nowotka, M.; Bento, A. P.; Chambers, J.; Mendez, D.; Mutowo, P.; Atkinson, F.; Bellis, L. J.; Cibrián-Uhalte, E.; et al. The ChEMBL Database in 2017. *Nucleic Acids Res.* **2017**, *45* (D1), D945–D954. https://doi.org/10.1093/nar/gkw1074.

(31) Visini, R.; Awale, M.; Reymond, J. L. Fragment Database FDB-17. *J. Chem. Inf. Model.* **2017**, *57* (4), 700–709. https://doi.org/10.1021/acs.jcim.7b00020.

(32) Awale, M.; Sirockin, F.; Stiefl, N.; Reymond, J. Drug Analogs from Fragment Based Long Short-Term Memory Generative Neural Networks. **2018**, *17* (1), 0–19. https://doi.org/10.26434/chemrxiv.7277354.v1.

(33) Popova, M.; Isayev, O.; Tropsha, A. Deep Reinforcement Learning for de Novo Drug Design. *Sci. Adv.* **2018**, *4* (7), eaap7885. https://doi.org/10.1126/sciadv.aap7885.

(34) Guimaraes, G. L.; Sanchez-Lengeling, B.; Outeiral, C.; Farias, P. L. C.; Aspuru-Guzik, A. Objective-Reinforced Generative Adversarial Networks (ORGAN) for Sequence Generation Models. **2017**. https://doi.org/arXiv:1705.10843v3.

(35) Sanchez-Lengeling, B.; Outeiral, C.; Guimaraes, G. L.; Aspuru-Guzik, A. Optimizing Distributions over Molecular Space. An Objective-Reinforced Generative Adversarial Network for Inverse-Design Chemistry (ORGANIC). *ChemRxiv* **2017**, 1–18. https://doi.org/10.26434/chemrxiv.5309668.v3.

(36) Lipinski, C. A.; Lombardo, F.; Dominy, B. W.; Feeney, P. J. Experimental and Computational Approaches to Estimate Solubility and Permeability in Drug Discovery and Development Settings. *Adv. Drug Deliv. Rev.* **2012**, *64* (SUPPL.), 4–17. https://doi.org/10.1016/j.addr.2012.09.019.

(37) Besnard, J.; Muresan, S.; Hopkins, A. L.; Paolini, G. V.; Bickerton, G. R. Quantifying the Chemical Beauty of Drugs. *Nat. Chem.* **2012**, *4* (2), 90–98. https://doi.org/10.1038/nchem.1243.

(38) Putin, E.; Asadulaev, A.; Ivanenkov, Y.; Aladinskiy, V.; Sanchez-Lengeling, B.; Aspuru-Guzik, A.; Zhavoronkov, A. Reinforced Adversarial Neural Computer for de Novo

Molecular Design. *J. Chem. Inf. Model.* **2018**, *58* (6), 1194–1204. https://doi.org/10.1021/acs.jcim.7b00690.

(39) Putin, E.; Asadulaev, A.; Vanhaelen, Q.; Ivanenkov, Y.; Aladinskaya, A. V.; Aliper, A.; Zhavoronkov, A. Adversarial Threshold Neural Computer for Molecular de Novo Design. *Mol. Pharm.* **2018**, *15* (10), 4386–4397. https://doi.org/10.1021/acs.molpharmaceut.7b01137.

(40) King, H.; Zwols, Y.; Blunsom, P.; Hermann, K. M.; Reynolds, M.; Colmenarejo, S. G.; Agapiou, J.; Kavukcuoglu, K.; Grabska-Barwińska, A.; Grefenstette, E.; et al. Hybrid Computing Using a Neural Network with Dynamic External Memory. *Nature* **2016**, *538* (7626), 471–476. https://doi.org/10.1038/nature20101.

(41) Blaschke, T.; Olivecrona, M.; Engkvist, O.; Bajorath, J.; Chen, H. Application of Generative Autoencoder in De Novo Molecular Design. *Mol. Inform.* **2018**, *37* (1). https://doi.org/10.1002/minf.201700123.

(42) Dai, H.; Tian, Y.; Dai, B.; Skiena, S.; Song, L. Syntax-Directed Variational Autoencoder for Structured Data. **2018**.

(43) Xu, Z.; Wang, S.; Zhu, F.; Huang, J. Seq2seq Fingerprint: An Unsupervised Deep Molecular Embedding for Drug Discovery. *Proc. 8th ACM Int. Conf. Bioinformatics, Comput. Biol. Heal. Informatics - ACM-BCB '17* **2017**. https://doi.org/10.1145/3107411.3107424.

(44) Lim, J.; Ryu, S.; Kim, J. W.; Kim, W. Y. Molecular Generative Model Based on Conditional Variational Autoencoder for de Novo Molecular Design. *J. Cheminform.* **2018**, *10* (1). https://doi.org/10.1186/s13321-018-0286-7.

(45) Pogány, P.; Arad, N.; Genway, S.; Pickett, S. D. De Novo Molecule Design by Translating from Reduced Graphs to SMILES. *J. Chem. Inf. Model.* **2018**, acs.jcim.8b00626. https://doi.org/10.1021/acs.jcim.8b00626.

(46) Sattarov, B.; Baskin, I. I.; Horvath, D.; Marcou, G. G.; Bjerrum, E. J.; Varnek, A. De Novo Molecular Design by Combining Deep Autoencoder Recurrent Neural Networks with Generative Topographic Mapping. *J. Chem. Inf. Model.* **2019**, *0* (0), acs.jcim.8b00751. https://doi.org/10.1021/acs.jcim.8b00751.

(47) Johnson, D. D. Learning Graphical State Transitions. *Int. Conf. Learn. Represent.* **2017**, 1–19. https://doi.org/10.1007/s10865-013-9551-4.

(48) Li, Y.; Vinyals, O.; Dyer, C.; Pascanu, R.; Battaglia, P. Learning Deep Generative Models of Graphs. *Iclr* **2018**, 1–16. https://doi.org/10.1146/annurev-statistics-010814-020120.

(49) Li, Y.; Zhang, L.; Liu, Z. Multi-Objective de Novo Drug Design with Conditional Graph Generative Model. *J. Cheminform.* **2018**, *10* (1), 1–24. https://doi.org/10.1186/s13321-018-0287-6.

(50) Kipf, T. N.; Welling, M. Semi-Supervised Classification with Graph Convolutional Networks. **2016**, 1–14. https://doi.org/10.1051/0004-6361/201527329.

(51) You, J.; Liu, B.; Ying, R.; Pande, V.; Leskovec, J. Graph Convolutional Policy Network for Goal-Directed Molecular Graph Generation. **2018**, 1–11.

https://doi.org/arXiv:1806.02473v1.

(52) Jin, W.; Barzilay, R.; Jaakkola, T. Junction Tree Variational Autoencoder for Molecular Graph Generation. **2018**.

(53) Rarey, M.; Dixon, J. S. Feature Trees: A New Molecular Similarity Measure Based on Tree Matching. *J. Comput. Aided. Mol. Des.* **1998**, *12* (5), 471–490. https://doi.org/10.1023/A:1008068904628.

(54) Yakoub, K.; Jung, S.; Sattler, C.; Damerow, H.; Weber, J.; Kretzschmann, A.; Cankaya, A. S.; Piel, M.; Rösch, F.; Haugaard, A. S.; et al. Structure-Function Evaluation of Imidazopyridine Derivatives Selective for δ-Subunit-Containing γ-Aminobutyric Acid Type A (GABAA) Receptors. *J. Med. Chem.* **2018**, *61* (5), 1951–1968. https://doi.org/10.1021/acs.jmedchem.7b01484.

(55) Goodfellow, I. J.; Pouget-Abadie, J.; Mirza, M.; Xu, B.; Warde-Farley, D.; Ozair, S.; Courville, A.; Bengio, Y. Generative Adversarial Networks. **2014**, 1–9. https://doi.org/10.1001/jamainternmed.2016.8245.

(56) De Cao, N.; Kipf, T. MolGAN: An Implicit Generative Model for Small Molecular Graphs. **2018**. https://doi.org/arXiv:1805.11973v1.

(57) Arús-Pous, J.; Blaschke, T.; Ulander, S.; Reymond, J.-L.; Chen, H.; Engkvist, O. Exploring the GDB-13 Chemical Space Using Deep Generative Models. *J. Cheminform.* **2019**, *11* (1), 20. https://doi.org/10.1186/s13321-019-0341-z.

(58) Liu, Q.; Allamanis, M.; Brockschmidt, M.; Gaunt, A. L. Constrained Graph Variational Autoencoders for Molecule Design. **2018**, 1–12. https://doi.org/10.1093/bja/aer158.

(59) Simonovsky, M.; Komodakis, N. GraphVAE: Towards Generation of Small Graphs Using Variational Autoencoders. *Lect. Notes Comput. Sci. (including Subser. Lect. Notes Artif. Intell. Lect. Notes Bioinformatics)* **2018**, *11139 LNCS*, 412–422. https://doi.org/10.1007/978-3-030-01418-6_41.

(60) Samanta, B.; De, A.; Jana, G.; Chattaraj, P. K.; Ganguly, N.; Gomez-Rodriguez, M. NeVAE: A Deep Generative Model for Molecular Graphs. **2018**.

(61) 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), 3098–3104. https://doi.org/10.1021/acs.molpharmaceut.7b00346.

(62) Yang, X.; Zhang, J.; Yoshizoe, K.; Terayama, K.; Tsuda, K. ChemTS: An Efficient Python Library for de Novo Molecular Generation. *Sci. Technol. Adv. Mater.* **2017**, *18* (1), 972–976. https://doi.org/10.1080/14686996.2017.1401424.

(63) 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), 1736–1741. https://doi.org/10.1021/acs.jcim.8b00234.

(64) Unterthiner, T.; Mayr, A.; Klambauer, G.; Hochreiter, S.; Steijaert, M.; Wegner, J. K.; Ceulemans, H.; Clevert, D.-A. Large-Scale Comparison of Machine Learning Methods for Drug Target Prediction on ChEMBL. *Chem. Sci.* **2018**, *9* (24), 5441–5451.

https://doi.org/10.1039/c8sc00148k.

(65) Brown, N.; Fiscato, M.; Segler, M. H. S.; Vaucher, A. C. *GuacaMol: Benchmarking Models for De Novo Molecular Design*; 2018. https://doi.org/arXiv:1811.09621v1.

(66) Polykovskiy, D.; Zhebrak, A.; Sanchez-Lengeling, B.; Golovanov, S.; Tatanov, O.; Belyaev, S.; Kurbanov, R.; Artamonov, A.; Aladinskiy, V.; Veselov, M.; et al. Molecular Sets (MOSES): A Benchmarking Platform for Molecular Generation Models. **2018**. https://doi.org/arXiv:1811.12823v1.

(67) Kussul, E.; Baidyk, T. Improved Method of Handwritten Digit Recognition Tested on MNIST Database. *Image Vis. Comput.* **2004**, *22* (12 SPEC. ISS.), 971–981. https://doi.org/10.1016/j.imavis.2004.03.008.

(68) Sterling, T.; Irwin, J. J. ZINC 15 - Ligand Discovery for Everyone. *J. Chem. Inf. Model.* **2015**, *55* (11), 2324–2337. https://doi.org/10.1021/acs.jcim.5b00559.

(69) Landrum, G. RDKit: Open-Source Cheminformatics. 2014, p http://www.rdkit.org/.

(70) Schneider, G. Automating Drug Discovery. *Nat. Rev. Drug Discov.* **2018**, *17* (2), 97–113. https://doi.org/10.1038/nrd.2017.232.