# Deep learning for molecular generation and optimization - a review of the state of the art

Daniel C. Elton,[1, a)] Zois Boukouvalas,[1] Mark D. Fuge,[1] and Peter W. Chung[1]
*Department of Mechanical Engineering, University of Maryland, College Park*

(Dated: 12 March 2019)

In the space of only a few years, deep generative modeling has revolutionized how we think of artificial creativity, yielding autonomous systems which produce original images, music, and text. Inspired by these successes, researchers are now applying deep generative modeling techniques to the generation and optimization of molecules- in our review we found 45 papers on the subject published in the past two years. These works point to a future where such systems will be used to generate lead molecules, greatly reducing resources spent downstream synthesizing and characterizing bad leads in the lab. In this review we survey the increasingly complex landscape of models and representation schemes that have been proposed. The four classes of techniques we describe are recursive neural networks, autoencoders, generative adversarial networks, and reinforcement learning. After first discussing some of the mathematical fundamentals of each technique, we draw high level connections and comparisons with other techniques and expose the pros and cons of each. Several important high level themes emerge as a result of this work, including the shift away from the SMILES string representation of molecules towards more sophisticated representations such as graph grammars and 3D representations, the importance of reward function design, the need for better standards for benchmarking and testing, and the benefits of adversarial training and reinforcement learning over maximum likelihood based training.



The average cost to bring a new drug to market is now well over one billion USD,[1] with an average time from discovery to market of 13 years.[2] Outside of pharmaceuticals the average time from discovery to commercial production can be even longer, for instance for energetic molecules it is 25 years.[3] A critical first step in molecular discovery is generating a pool of candidates for computational study or synthesis and characterization. This is a daunting task because the space of possible molecules is enormous—the number of potential drug-like compounds has been estimated to be between $10^{23}$ and $10^{60}$,[4] while the number of all compounds that have been synthesized is on the order of $10^8$. Heuristics, such as Lipinski's "rule of five" for pharmaceuticals[5] can help narrow the space of possibilities, but the task remains daunting. High throughput screening (HTS)[6] and high throughput virtual screening (HTVS)[7] techniques have made larger parts of chemical space accessible to computational and experimental study. Machine learning has been shown to be capable of yielding rapid and accurate property predictions for many properties of interest and is being integrated into screening pipelines, since it is orders of magnitude faster than traditional computational chemistry methods.[8] Techniques for the interpretation and "inversion" of a machine learning model can illuminate structure-property relations that have been learned by the model which can in turn be used to guide the design of new lead molecules.[9,10] However even with these new techniques bad leads still waste limited supercomputer and laboratory resources, so minimizing the number of bad leads generated at the start of the pipeline remains a key priority. The focus of this review is on the use of deep learning techniques for the targeted generation of molecules and guided exploration of chemical space. We note that machine learning (and more broadly artificial intelligence) is having an impact on accelerating other parts of the chemical discovery pipeline as well, via machine learning accelerated ab-initio simulation,[8] machine learning based reaction prediction,[11,12] deep learning based synthesis planning,[13] and the development of high-throughput "self-driving" robotic laboratories.[14,15]

Deep neural networks, which are often defined as networks with more than three layers, have been around for many decades but until recently were difficult to train and fell behind other techniques for classification and regression. By most accounts, the deep learning revolution in machine learning began in 2012, when deep neural network based models began to win several different competitions for the first time. First came a demonstration by Cireşan et al. of how deep neural networks could achieve near-human performance on the task of handwritten digit classification.[16] Next came groundbreaking work by Krizhevsky et al. which showed how deep convolutional networks achieved superior performance on the 2010 ImageNet image classification challenge.[17] Finally, around the same time in 2012, a multitask neural network developed by Dahl et al. won the "Merck Molecular Activity Challenge" to predict the molecular activities of molecules at 15 different sites in the body, beating out more traditional machine learning approaches such as boosted decision trees.[18] One of the key technical advances published that year and used by both Krizhevsky et al. and Dahl et al. was a novel regularization trick called "dropout".[19] Another important technical advance was the efficient implementation of neural network training on graphics processing units (GPUs). By 2015 better

[a)]Electronic mail: daniel.elton@nih.gov

hardware, deeper networks, and a variety of further technical advances had reduced error rates on the ImageNet challenge by a factor of 3 compared to the Krizhevsky's 2012 result.[20]

In addition to the tasks of classification and regression, deep neural networks began to be used for generation of images, audio, and text, giving birth to the field of "deep generative modeling". Two key technical advances in deep generative modeling were the variational autoencoder (Kimga et al., 2013[21]) and generative adversarial networks (Goodfellow et al. 2014[22]). The first work demonstrating deep generative modeling of molecules was the "molecular autoencoder" work of Gómez-Bombarelli et al. which appeared on the arXiv in October 2016 and was published in *ACS Central Science* in 2018.[23] Since then, there has been an explosion of advancements in deep generative modeling of molecules using several different deep learning architectures and many variations thereof, as shown in table II. In addition to new architectures, new representation schemes, many of which are graph based, have been introduced as alternatives to the SMILES representation used by Gómez-Bombarelli et al. The growing complexity of the landscape of architectures and representations and the lack of agreement upon standards for benchmarking and comparing different approaches has prompted us to write this review.

While much of the work so far has focused on deep generative modeling for drug molecules,[24] there are many other application domains which are benefiting from the application of deep learning to lead generation and screening, such as organic light emitting diodes,[25] organic solar cells,[26] energetic materials,[10,27] electrochromic devices,[28] polymers,[29] polypeptides,[30–32] and metal organic frameworks.[33,34]

In this review we discuss the pros and cons underlying different choices of model architecture and representation. We have also included some of the key equations underlying each major architecture type to better illuminate how generative models work "under the hood" and how they are related to each other mathematically. One major emphasis in this work is to highlight the possible advantages of adversarial training and reinforcement learning over maximum likelihood based training. Another emphasis is the movement away from the popular Simplified Molecular-Input Line-Entry System (SMILES) string representation towards representations which are "closer to the chemical structure", such as graph grammar based methods. We also touch on techniques for molecular optimization using generative models, an important area of research which has seen many concomitant developments. Here, a key contribution of this review is a discussion of reward function design, which is crucial for the practical application of reinforcement learning based optimization. Reward functions must be carefully engineered if one wishes to generate a set of leads which is chemically stable, diverse, novel, has good properties, and is synthesizable. Finally, we discuss how to quantitatively evaluate different approaches for molecular generation and optimization. By highlighting important quantitative comparisons we draw attention to the approaches which are most promising for future development.

There are reasons to be skeptical about whether today's deep generative models can outperform traditional computational approaches to lead generation and optimization. Traditional approaches are fundamentally combinatorial in nature and involve mixing scaffolds, functional groups, and fragments known to be relevant to the problem at hand (for a review, see Pirard et al.[35]). A naive combinatorial approach to molecular generation leads to most molecules being unstable or impossible to synthesize, so details about chemical bonding generally must be incorporated. One approach is to have an algorithm perform virtual chemical reactions, either from a list of known reactions, or using ab initio methods for reaction prediction.[36] Another popular approach is to use genetic algorithms with custom transformation rules which are known to maintain chemical stability.[37] One of the latest genetic algorithm based approaches ("Grammatical Evolution") can match the performance of the deep learning approaches for molecular optimization under some metrics.[38] Deep generative modeling of molecules has made rapid progress in just a few years and there are reasons to expect this progress to continue, not just with better hardware and data, but due to new architectures and approaches. For instance, generative adversarial networks and deep reinforcement learning (which may be combined or used separately) have both seen technical advancements recently.

## CONTENTS

| | method | unique? | invertible? |
|---|---|---|---|
| 3D | raw voxels | ✗ | ✓ |
| | smoothed voxels | ✗ | ✓ |
| | tensor field networks | ✗ | ✗ |
| 2D graph | SMILES | ✗ | ✓ |
| | canonical SMILES | ✓ | ✓ |
| | InChI | ✓ | ✓ |
| | MACCS keys | ✓ | ✗ |
| | tensors | ✗ | ✓ |
| | Chemception images | ✓ | ✓ |
| | fingerprinting | ✓ | ✗ |

TABLE I. Different representation schemes.

## I. MOLECULAR REPRESENTATION

A key choice before attempting either generative modeling or predictive modeling of molecules is the choice of representation. Creating an appropriate representation from a molecular structure is called featurization. Two important properties that are desirable (but not required) for representations are uniqueness and invertibility. Uniqueness means that each molecular structure is associated with a single representation. Invertibility means that each representation is associated with a single molecule (a one-to-one mapping). Most representations used for molecular generation are invertible, but many are non-unique. There are many reasons for non-uniqueness, including the representation not being invariant to the underlying physical symmetries of rotation, translation, and permutation of atomic indexes.

Another factor one should consider when choosing a representation is the whether it is a character sequence or tensor. Some methods only work with sequences, while others only work with tensor. Sequences may be converted into tensors using one-hot encoding. Another choice is whether to use a representation based on the 3D coordinates of the molecule or a representation based on the 2D connectivity graph. Molecules are fundamentally three dimensional quantum mechanical objects, typically visualized as consisting of nuclei with well-defined positions surrounded by many electrons which are described by a complex-valued wavefunction. Fundamentally, all properties of a molecule can be predicted using quantum mechanics given only the relative coordinates of the nuclei and the type and ionization state of each atom. However, for many applications, working with 3D representations is cumbersome and unnecessary. In this section, we review both 3D and 2D representation schemes that have been developed recently.

### A. Representations of 3D geometry

Trying to implement machine learning directly with nuclear coordinates introduces a number of issues. The main issue is that coordinates are not invariant to molecular translation, rotation, and permutation of atomic indexing. While machine learning directly on coordinates is possible, it is much better to remove invariances to create a more compact representation (by removing degrees of freedom) and develop a scheme to obtain a unique representation for each molecule. One approach that uses 3D coordinates uses a 3D grid of voxels and specifies the nuclear charge contained within each voxel, thus creating a consistent representation. Nuclear charge (i.e. atom type) is typically specified by a one-hot vector of dimension equal to the number of atom types in the dataset. This scheme leads to a very high dimensional sparse representation, since the vast majority of voxels will not contain a nuclear charge. While sparse representations are considered desirable in some contexts, here sparsity leads to very large training datasets being required. This issue can be mitigated via spatial smoothing (blurring) by placing spherical Gaussians or a set of decaying concentric waves around each atomic nuclei.[39] Alternatively, the van der Waals radius may be used.[40] Amidi et al. use this type of approach for predictive modeling with 3D convolutional neural networks (CNNs),[41] while Kuzminykh et al. and Skalic et al. use this approach with a CNN-based autoencoder for generative modeling.[39,40]

Besides high dimensionality and sparsity, another issue with 3D voxelized representations is they do not capture invariances to translation, rotation, and reflection, which are hard for present-day deep learning based architectures to learn. Capturing such invariances is important for property prediction, since properties are invariant to such transformations. It is also important for creating compact representations of molecules for generative modeling. One way to deal with such issues is to always align the molecular structure along a principal axis as determined by principle component analysis to ensure a unique orientation.[39,41] Approaches which generate feature vectors from 3D coordinates that are invariant to translation and rotation are wavelet transform invariants,[42] solid harmonic wavelet scattering transforms,[43] and tensor field networks.[44] All of these methods incur a loss of information about 3D structure and are not easy to invert, so their utility for generative modeling may be limited. Despite this issue, tensor field networks have been suggested to have utility for generative modeling since it was shown they can accurately predict the loca-

tion of missing atoms in molecules where one atom was removed.[44] We expect future work on 3D may proceed in the direction of developing invertible representations that are based on the internal (relative) coordinates of the molecule.

### B. Representations of molecular graphs

#### 1. SMILES and string-based representations

Most generative modeling so far has not been done with coordinates but instead has worked with molecular graphs. A molecule can be considered as an undirected graph $\mathcal{G}$ with a set of edges $\mathcal{E}$ and set of vertices $\mathcal{V}$. The obvious disadvantage of such graphs is that information about bond lengths and 3D conformation is lost. For some properties one may wish to predict, the specific details of a molecule's 3D conformations may be important. For instance, when packing in a crystal or binding to a receptor, molecules will find the most energetically favorable conformation, and details of geometry often have a big effect. Despite this, graph representations have been remarkably successful for a variety of generative modeling and property prediction tasks. If a 3D structure is desired from a graph representation, molecular graphs can be embedded in 3D using distance geometry methods (for instance as implemented in the *OpenBabel* toolkit[45,46]). After coordinate embedding, the most energetically favorable conformation of the molecule can be obtained by doing energy minimization with classical forcefields or quantum mechanical simulation.

There are several ways to represent graphs for machine learning. The most popular way is the SMILES string representation.[47] SMILES strings are a non-unique representation which encode the molecular graph into a sequence of ASCII characters using a depth-first graph traversal. SMILES are typically first converted into a one-hot based representation. Generative models then produce a categorical distribution for each element, often with a softmax function, which is sampled. Since standard multinomial sampling procedures are non-differentiable, sampling can be avoided during training or a Gumbel-softmax can be used.[48,49]

Many deep generative modeling techniques have been developed specifically for sequence generation, most notably Recurrent Neural Networks (RNNs), and the transference of such techniques to SMILES generation has proven to be a fruitful research direction. The non-uniqueness of SMILES arises from a fundamental ambiguity about which atom to start the SMILES string construction on, which means that every molecule with $N$ heavy (non-hydrogen) atoms has at least $N$ SMILES strings representing it. There is additional non-uniqueness due to different conventions on whether to include charge information in resonance structures such as nitro groups and azides. The *MolVS*[50] or *RDKit*[51] cheminformatics packages can be used to standardize SMILES, putting them in a canonical form. However, Bjerrum et al. have pointed out that the latent representations obtained from canonical SMILES may be less useful because they become more related to specific grammar rules of canonical SMILES rather than the chemical structure of the underlying molecule.[52] This is considered an issue for interpretation and optimization since it is better if latent spaces encode underlying chemical properties and capture notions of chemical similarity rather than SMILES syntax rules. Bjerrum et al. have suggested SMILES enumeration (training on all SMILES representations of each molecule), rather than using canonical SMILES, as a better solution to the non-uniqueness issue.[52,53] An approach similar to SMILES enumeration is used in computer vision applications to obtain rotational invariance- image datasets are often "augmented" by including many rotated versions of each image. Another approach to obtain better latent representations explored by Bjerrum et al. is to input both enumerated SMILES and Chemception-like image arrays (discussed below) into a single "heteroencoder" framework.[52]

In addition to SMILES strings, Gómez-Bombarelli et al. have tried InChI strings[54] with their variational autoencoder, but found they led to inferior performance in terms of the decoding rate and the subjective appearance of the molecules generated. Interestingly, Winther et al. show more physically meaningful latent spaces can be obtained by training a variational autoencoder to translate between InChI to SMILES.[55] There is an intuitive explanation for this- the model must learn to extract the underlying chemical structures which are encoded in different ways by the two representations.

SMILES based methods often struggle to achieve a high percentage of valid SMILES. As a possible solution to this, Kusner et al. proposed decomposing SMILES into a sequence of rules from a context free grammar (CFG).[56] The rules of the context-free grammar impose constraints based on the grammar of SMILES strings.[57] Because the construction of SMILES remains probabilistic, the rate of valid SMILES generation remains below 100%, even when CFGs are employed and additional semantic constraints are added on top.[57] Despite the issues inherent with using SMILES, we expect it will continue to a popular representation since most datasets store molecular graphs using SMILES as the native format, and since architectures developed for sequence generation (i.e. for natural language or music) can be readily adopted. Looking longer term, we expect a shift towards methods which work directly with the graph and construct molecules according to elementary operations which maintain chemical valence rules.

Li et al. have developed a conditional graph generation procedure which obtains a very high rate of valid chemical graphs (91%) but lower negative log likelihood scores compared to a traditional SMILES based RNN model.[58] Another more recent work by Li et al. uses a deep neural network to decide on graph generation steps (append, connect, or terminate).[59] Efficient algorithms for graph

and tree enumeration have been previously developed in a more pure computer science context. Recent work has looked at how such techniques can be used for molecular graph generation,[60] and likely will have utility for deep generative models as well.

### 2. Image-based representations

Most small molecules are easily represented as 2D images (with some notable exceptions like cubane). Inspired by the success of Google's Inception-ResNet deep convolutional neural network (CNN) architecture for image recognition, Goh et al. developed "Chemception", a deep CNN which predicts molecular properties using custom-generated images of the molecular graph.[61] The Chemception framework takes a SMILES string in and produces an 80x80 greyscale image which is actually an array of integers, where empty space is '0', bonds are '2' and atoms are represented by their atomic number.[61] Bjerrum et al. extend this idea, producing "images" with five color channels which encode a variety of molecular features, some which have been compressed to few dimensions using PCA.[52]

### 3. Tensor representations

Another approach to storing the molecular graph is to store the vertex type (atom type), edge type (bond type), and connectivity information in multidimensional arrays (tensors). In the approach used by de Cao & Kipf,[49,62] each atom is a vertex $v_i \in \mathcal{V}$ which may be represented by a one-hot vector $\boldsymbol{x}_i \in \{0,1\}^{|\mathcal{A}|}$ which indicates the atom type, out of $|\mathcal{A}|$ possible atom types. Each bond is represented by an edge $(v_i, v_j)$ which is associated with a one-hot vector $\boldsymbol{y}_i \in \{0,1\}^Y$ representing the type of bond out of $Y$ possible bond types. The vertex and edge information can be stored in a vertex feature matrix $X = [\boldsymbol{x}_1, \ldots, \boldsymbol{x}_N]^T \in \mathbb{R}^{N \times |\mathcal{A}|}$ and an adjacency tensor $A \in \mathbb{R}^{N \times N \times Y}$ where $A_{ij} \in \mathbb{R}^Y$. Simonovsky et al.[63] use a similar approach- they taie a vertex feature matrix $X$ and concatenate the adjacency tensor $A$ with a traditional adjacency matrix where connections are indicated by a '1'. As with SMILES, adjacency matrices suffer from non-uniqueness- for a molecule with $N$ atoms, there are $N!$ equivalent adjacency matrices representing the same molecular graph, each corresponding to a different reordering of the atoms/nodes. This makes it challenging to compute objective functions, which require checking if two adjacency matrix representations correspond to the same underlying graph (the "graph equivalency" problem). Simonovsky et al. use an approximate graph matching algorithm to do this, but it is still computationally expensive.

### 4. Other graph-based representations

Another approach is to train an RNN or reinforcement learning agent to operate directly on the molecular graph, adding new atoms and bonds in each action step from a list of predefined possible actions. This approach is taken with the graph convolutional policy network[64] and in recent work using pure deep reinforcement learning to generate molecules.[65] Because these methods work directly on molecular graphs with rules which ensure that basic atom valence is satisfied, they generate 100% chemically valid molecules.

Finally, when limited to small datasets one may elect to do generative modeling with compact feature vectors based on fingerprinting methods or descriptors. There are many choices (Coulomb matrices, bag of bonds, sum over bonds, descriptor sets, graph convolutional fingerprints, etc.) which we have previously tested for regression,[27,66] but they are generally not invertible unless a very large database with a look-up table has been constructed. (In this context, by invertible we mean the complete molecular graph can be reconstructed without loss.) As an example of how it may be done, Kadurin et al. use 166 bit Molecular ACCess System (MACCS) keys[67] for molecular representation with adversarial autoencoders.[68,69] In MACCS keys, also called MACCS fingerprints, each bit is associated with a specific structural pattern or question about structure. To associate molecules to MACCS keys one must search for molecules with similar or identical MACCS keys in a large chemical database. Fortunately several large online chemical databases have application programming interfaces (APIs) which allow for MACCS-based queries, for instance *PubChem*, which contains 72 million compounds.

## II. DEEP LEARNING ARCHITECTURES

In this section we summarize the mathematical foundations of several popular deep learning architectures and expose some of their pros and cons. A basic familiarity with machine learning concepts is assumed.

### A. Recurrent neural networks (RNNs)

We discuss recurrent neural network sequence models first because they are fundamental to molecular generation- most VAE and GAN implementations include an RNN for sequence generation. In what follows, a sequence of length $T$ will be denoted as $S_{1:T} = (s_1, \cdots, s_T), s_t \in \mathcal{V}$, where $\mathcal{V}$ is the set of tokens, also called the vocabulary. For the purpose of this section we assume the sequences in question are SMILES strings, as they are by far the most widely used. As discussed previously in the context of SMILES the "tokens" are the different characters which are used to specify atom types,

| architecture | representation | $N_{\text{train}}$ | dataset(s) | citation(s) |
|---|---|---|---|---|
| RNN | SMILES | 1,611,889 | ZINC | Bjerrum, 2017[70] |
| RNN | SMILES | 541,555 | ChEMBL* | Gupta, 2017[71] |
| RNN | SMILES | 350,419 | DRD2 | Oliverona, 2017[72] |
| RNN | SMILES | 1,400,000 | ChEMBL | Segler, 2017[73] |
| RNN | SMILES | 250,000 | ZINC | Yang, 2017[74] |
| RNN | SMILES | 200,000 | ZINC | Cherti, 2017[75] |
| RNN | SMILES | 1,735,442 | ChEMBL | Neil, 2018[76] |
| RNN | SMILES | 1,500,000 | ChEMBL | Popova, 2018[77] |
| RNN | SMILES | 13,000 | PubChemQC | Sumita, 2018[78] |
| RNN | SMILES | 541,555 | ChEMBL* | Merk, 2018[79] |
| RNN | SMILES | 541,555 | ChEMBL* | Merk, 2018[80] |
| RNN | SMILES | 509,000 | ChEMBL | Ertl, 2018[81] |
| RNN | SMILES | 1,000,000 | GDB-13 | Arús-Pous, 2018[82] |
| RNN | SMILES | 163,000 | ZINC | Zheng, 2019[83] |
| RNN | RG+SMILES | 798,243 | ChEMBL | Pogány, 2018[84] |
| RNN | graph operations | 130,830 | ChEMBL | Li, 2018[58] |
| VAE | SMILES | 249,000 | ZINC/QM9 | Gómez-Bombarelli, 2016[23] |
| VAE | SMILES | 1,200,000 | ChEMBL | Blaschke, 2018[85] |
| VAE | SMILES | 500,000 | ZINC | Lim, 2018[86] |
| VAE | SMILES | 300,000 | ZINC | Kang, 2018[87] |
| VAE | SMILES | 190,000 | ZINC | Harel, 2018[88] |
| GVAE | CFG (SMILES) | 200,000 | ZINC | Kusner, 2017[89] |
| GVAE | CFG (custom) | 3,989 | PSC | Jørgensen, 2018[26,90] |
| SD-VAE | CFG (custom) | 250,000 | ZINC | Dai, 2018[57] |
| JT-VAE | graph operations | 25,000 | ZINC | Jin, 2018[91] |
| CVAE | graph | 250,000 | ZINC/CEPDB | Liu, 2018[92] |
| MHG-VAE | graph (MHG) | 220,011 | ZINC | Kajino, 2018[93] |
| VAE | graph (tensors) | 250,000 | ZINC/QM9 | Simonovsky, 2018[63] |
| VAE | graph | 10,000 | ZINC/QM9 | Samanta, 2018[94] |
| VAE | graph | 72,000,000 | ZINC+PubChem | Winter, 2018[55] |
| VAE | 3D wave transform | 4,8000,000 | ZINC | Kuzminkykh, 2018[39] |
| CVAE | 3D density | 192,813,983 | ZINC | Skalic, 2019[40] |
| GAN | SMILES | 5,000 | GBD-17 | Guimaraes, 2017[95] |
| GAN (ANC) | SMILES | 15,000 | ZINC/CHEMDIV | Putin, 2018[96] |
| GAN (ATNC) | SMILES | 15,000 | ZINC/CHEMDIV | Putin, 2018[97] |
| GAN | graph (tensors) | 133,885 | QM9 | De Cao, 2018[49,62] |
| GAN | MACCS (166bit) | 6,252 | MCF-7 | Kadurin, 2017[69] |
| sGAN | MACCS (166bit) | 20,000 | L1000 | Méndez-Lucio, 2017[98] |
| CycleGAN | graph operations | 250,000 | ZINC | Maziarka, 2019[99] |
| AAE | MACCS (166bit) | 6,252 | MCF-7 | Kadurin, 2017[68] |
| AAE | SMILES | 15,000 | HCEP | Sanchez-Lengeling, 2017[100] |
| BMI | SMILES | 16,674 | PubChem | Ikebata, 2017[101] |
| CAAE | SMILES | 1,800,000 | ZINC | Polykovskiy, 2018[102] |
| GCPN | graph | 250,000 | ZINC | You, 2018[64] |
| pure RL | graph | n/a | n/a | Zhou, 2018[65] |

TABLE II. For works that trained models separately on multiple datasets we report only the largest dataset used. Several of these datasets are described in table III, which lists the major publicly available datasets. Other datasets are "HCEP", the Harvard Clean Energy Project dataset of lead molecules for organic photovoltaic, "PSC", a dataset of monomer repeat units for polymer solar cells, "MCF-7", a database of anti-cancer molecules, and "L1000", a database of molecules and gene expression profiles.

Acronyms used are: AAE = adversarial autoencoder, ANC = adversarial neural computer, ATNC = adversarial threshold neural computer, BMI = Bayesian model inversion, CAAE = constrained AAE, CFG = context free grammar, CVAE = constrained VAE, ECC = edge-conditioned graph convolutions,[103]GAN = generative adversarial network, GCPN = graph convolutional policy network, GVAE = grammar VAE, JT-VAE = junction tree VAE, MHG = molecular hypergraph grammar, RG = reduced graph, RNN = recurrent neural network, sGAN = stacked GAN, SD-VAE = syntax-directed VAE, SSVAE = semi-supervised VAE, VAE = variational autoencoder.

* filtered to isolate likely bioactive compounds.

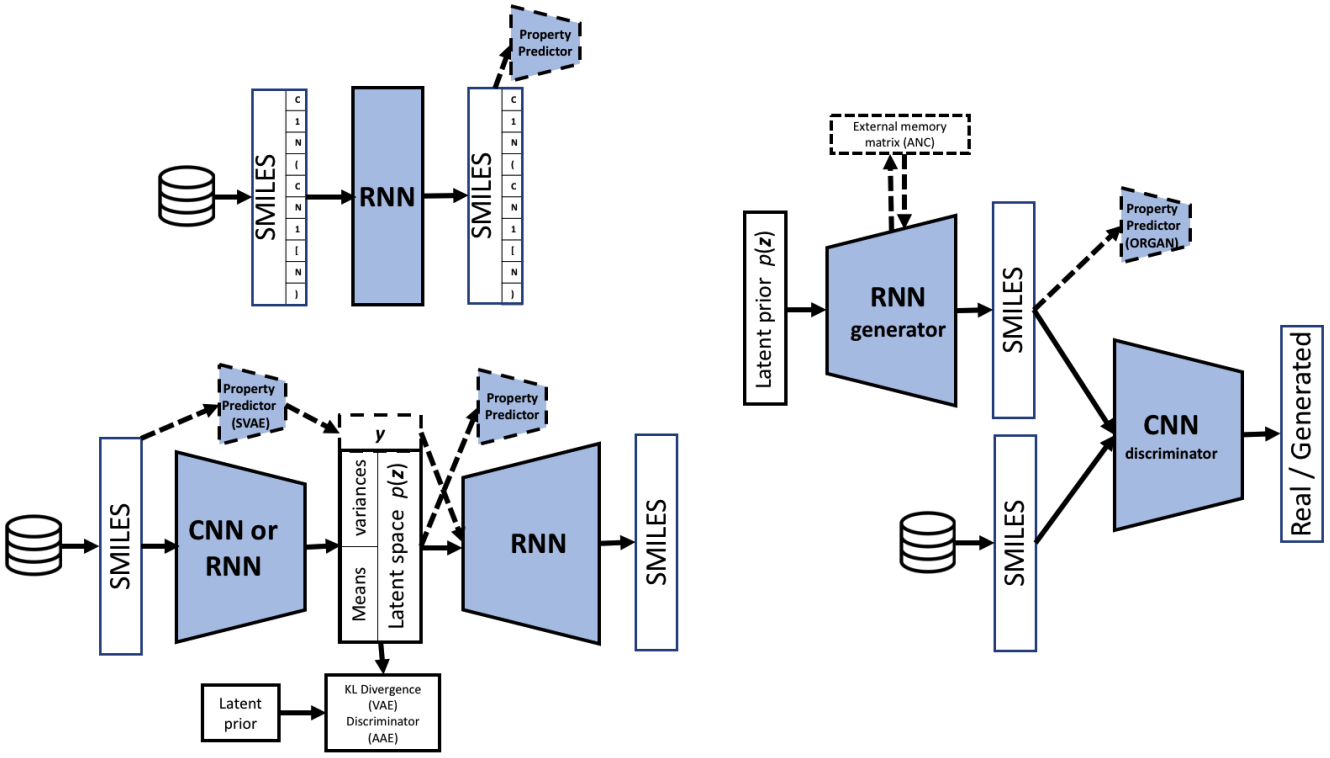


FIG. 1. Bird's eye views of three popular frameworks for generative modeling of SMILES strings- recurrent neural network trained with maximum likelihood (top left), the variational/adversarial autoencoder framework (bottom left) and the generative adversarial network framework (right). Possible variations are shown with dashed lines.

| dataset | description | $N$ | URL / citation |
|---|---|---|---|
| GDB-13 | Combinatorially generated library. | 977,468,314 | http://gdb.unibe.ch/downloads/[104] |
| ZINC15 | Commercially available compounds. | >750,000,000 | http://zinc15.docking.org[105] |
| GDB-17 | Combinatorially generated library. | 50,000,000 | http://gdb.unibe.ch/downloads/[106] |
| eMolecules | Commercially available compounds. | 18,000,000 | https://reaxys.emolecules.com/ |
| SureChEMBL | Compounds obtained from chemical patents. | 17,000,000 | https://www.surechembl.org/search/ |
| ChEMBL | A curated database of bioactive molecules. | 2,000,000 | https://www.ebi.ac.uk/chembl/ |
| SuperNatural | A curated database of natural products. | 2,000,000 | http://bioinformatics.charite.de/supernatural/ |
| QM9 | Stable small CHONHF organic molecules taken from GDB-17 with properties calculated from ab initio density functional theory. | 133,885 | http://quantum-machine.org/datasets/[107] |
| DrugBank | FDA drugs and other drugs available internationally. | 10,500 | https://www.drugbank.ca/ |
| HOPV15 | Harvard Organic Photovoltaic Dataset | 350* | https://figshare.com/articles /HOPV15_Dataset/1610063/4[108] |

TABLE III. Some publicly available datasets. *also contains numerous conformers for each molecule, for a total of 4,855 structures.

bond types, parentheses, and the start and stop points of rings. The first step in sequence modeling is typically one-hot encoding of the sequence's tokens, in which each token is represented as a unique $N$ dimensional vector where one element is 1 and the rest are 0 (where $N$ is the number of tokens in the vocabulary).

Recurrent neural networks (RNNs) are the most popular models for sequence modeling and generation. We will not go into detail of their architecture, since it is well described elsewhere.[109,110] An important detail to note however is that the type of RNN unit that is typically used for generating molecules is either the long short term memory (LSTM) unit,[111] or a newer more computationally efficient variant called the gated recurrent unit (GRU).[112] Both LSTMs and GRUs contain a memory cell which alleviates the exploding and vanishing gradient problems that can occur when training RNNs to predict long-term dependencies.[111,112]

Sequence models are often trained to predict just a single missing token in a sequence, as trying to predict

more than one token leads to a combinatorial explosion of possibilities. Any machine learning model trained to predict the next character in an input sequence can be run in "generative mode" or "autoregressive mode" by concatenating the predicted token to the input sequence and feeding the new sequence back into the model. However, this type of autoregressive generation scheme typically fails because the model was trained to predict on the data distribution and not its own generative distribution, and therefore each prediction contains at least a small error. As the network is run recursively, these errors rapidly compound, leading to rapid degradation in the quality of the generated sequences. This problem is known as "exposure bias".[113] The Data as Demonstrator (DaD) algorithm tries to solve the problem of exposure bias by running a model recursively during training and comparing the output to the training data during training.[114] DaD was extended to sequence generation with RNNs by Bengio et al., who called the method "scheduled sampling".[115] While research continues in this direction, issues have been raised about the lack of a firm mathematical foundation for such techniques, with some suggesting they do not properly approximate maximum likelihood.[116]

Better generative models can be obtained by training using maximum likelihood maximization on the sequence space rather than next-character prediction. In maximum likelihood training a model $\pi_\theta$ parametrized by $\theta$ is trained with the following differentiable loss:

$$L^{\text{MLE}} = -\sum_{s\in\mathcal{Z}}\sum_{t=2}^{T}\log \pi_\theta(s_t|S_{1:t-1}) \tag{1}$$

Here $\mathcal{Z}$ is the set of training sequences which are assumed to each be of length $T$. This expression is proportional to the negative cross entropy of the model distribution and the training data distribution (maximizing likelihood is equivalent to minimizing cross entropy). MLE training can be done with standard gradient descent techniques and backpropagation through time to compute the gradient of the loss. In practice though this type of training fails to generate valid SMILES, likely because of strict long term dependencies such as closing parentheses and rings. The "teacher forcing" training procedure[117] is an important ingredient which was found to be necessary to include in the molecular autoencoder VAE to capture such long term dependencies- otherwise the generation rate of valid SMILES was found to be near 0%.[118] In teacher forcing, instead of sampling from the model's character distribution to get the next character, the right character is given directly to the model during training.[110]

When running the model in generative mode, the typical method is to use a multinomial sampler to sample the full model distribution. This can yield samples that deviate greatly from the training data however, because MLE tends to focus on optimizing the peaks of the distribution and often improperly "fills in" regions between peaks. So called "thermal" rescaling can be used to sample further away from the peaks of the distribution by rescaling the probabilities as:

$$p_i^{\text{new}} = \frac{\exp\left(\frac{p_i}{T}\right)}{\sum_i \exp\left(\frac{p_i}{T}\right)} \tag{2}$$

where $T$ is a sampling temperature. Alternatively, if a softmax layer is used to generate the final output of a neural network, a temperature parameter can be built directly into it. Another alternative is the "freezing function":

$$p_i^{\text{new}} = \frac{p_i^{\frac{1}{T}}}{\sum_i p_i^{\frac{1}{T}}} \tag{3}$$

Sampling at low $T$ leads to the generation of molecules which are only slight variations on molecules seen in the training data. Generation at high $T$ leads to greater diversity but also higher probability of nonsensical results.

#### 1. Optimization with RNNs using reinforcement learning

Neil et al introduced a simple method for repeated MLE which biases generation towards molecules with good properties, which they call *HillClimb-MLE*.[76] Starting with a model that has been trained via MLE on the training data, they generate a large set of SMILES sequences. They then calculate a reward function $R(S)$ for each sequence $S$ generated and find the subset of $N'$ generated molecules with the highest rewards. This subset is used to retrain the model with MLE, and the process is repeated. Each time a new subset of $N'$ generated molecules is determined, it is concatenated on the previous set, so the amount of data being used for MLE grows with each iteration. As this process is repeated the model begins to generate molecules which return higher and higher values from $R(S)$.

A more common technique is to use *reinforcement learning* after MLE pretraining to fine tune the generator to produce molecules with high reward. The problem of sequence generation can be recast as a reinforcement learning problem with a discrete action space. At each timestep time $t$, the current state is the sequence generated so far is $(s_0, \cdots, s_t)$ and the action $a$ is next token to be chosen, $a = s_{t+1}$. The goal of reinforcement learning is to maximize the expected return $G_T$ for all possible start states $s_0$. The return function $G_t = \sum_{i=t}^{T} R_i$ simply sums the rewards over the length of time the agent is active, which is called an episode. Mathematically the optimization problem reinforcement learning tries to solve is expressed as:

$$\max_\theta J(\theta) = \mathbb{E}[G_T|s_0, \theta] \tag{4}$$

where $\theta$ are the parameters of the model. In our case, one episode corresponds to the generation of one molecule,

there is only one start state, (the 'GO' character) and $R_t = 0$ until the end-of-sequence ('EOS') token is generated or the max string length is reached. If $T$ denotes the max length of the SMILES string then only $R_T$ is nonzero and therefore $G_t = R_T$ for all $t$. The state transition is deterministic (i.e. $p^a_{s,s'} = 1$ for the next state $S_{1:t+1}$ if the current state is $S_{1:t}$ and the action $a = s_{t+1}$, while for other states $s''$, $p^a_{s,s''} = 0$). Because of these simplifications, eqn. 4 assumes a simple form:

$$J(\theta) = R_T \sum_{t=0}^{T} \pi_\theta(a_t|s_t) \tag{5}$$

Here the policy model $\pi_\theta(a|s)$ gives the probability for choosing the next action given the current state. In our case:

$$\pi_\theta(a_t|s_t) = \pi_\theta(s_{t+1}|S_{1:t}) \tag{6}$$

There are many reinforcement learning methods, but broadly speaking they can be broken into value learning and policy learning methods. Most work so far has used variants of the REINFORCE algorithm,[119] a type of policy learning method which falls into the class of policy gradient methods. It can be shown that for a run of length $T$ the gradient of $J(\theta)$ (eqn. 4) is:

$$\nabla J(\theta) = \mathbb{E}\left[G_t \frac{\nabla_\theta \pi_\theta(a_t|y_{1:t-1})}{\pi_\theta(a_t|y_{1:t-1})}\right] \tag{7}$$

Computing the exact expectation of $G_t$ for all possible action sequences is impossible, so instead the $G_t$ from a single run is used before each gradient update. This is sometimes referred to as a "Monte-Carlo" type approach. Fortunately, this process can be parallelized by calculating multiple gradient updates on different threads before applying them. Neil et al. recently tested several newer reinforcement learning algorithms- Advantage Actor-Critic (AAC) and Proximal Policy Optimization (PPO), where they report superior performance over REINFORCE (PPO > AAC > REINFORCE). Interestingly, they find *Hillclimb-MLE* is competitive with and occasionally superior to PPO.[76]

Olivecrona et al. argue that policy learning methods are a more natural choice for molecular optimization because they can start with a pre-trained generative model, while value-function learning based methods cannot.[72] Additionally, most policy learning methods have been proven to lead to an optimal policy and the resulting generative models are fast to sample.[72] In contrast, Zhou et al. argue that value function learning methods are superior in part because policy gradient methods suffer from issues with high variance in the gradient estimates during training.[65]

Empirically it has been found that using RL after MLE can cause the generated model to "drift" too far, causing important information about viable chemical structures learned during MLE to be lost. This can take the form of highly unstable structures being generated or invalid SMILES. One solution is to "augment" the reward function with the likelihood:[72,120]

$$R'(S) = [\sigma R(S) + \log P_{\text{prior}}(S) - \log P_{\text{current}}(S)]^2 \tag{8}$$

Other possibilities are explored by Olivecrona et al.[72] Fundamentally, whether "drift" during RL training becomes an issue depends on the details of the reward function—if the reward function is good, drift should be in a good direction. Recently Zhou et al. have questioned whether training a model with MLE is required at all for molecular optimization. In their RL based approach for molecular optimization, they do not use an RNN or any pre-trained generative model and instead use pure RL training.[65] The RL agent works directly on constructing molecular graphs, taking actions such as atom/bond addition and atom removal. The particular approach they use is deep-$Q$ learning, which incorporates several recent innovations that were developed at *DeepMind* and elsewhere.[121] Jaques et al. have also explored the application of deep $Q$-learning and $G$-Learning to molecular optimization.[120] Reinforcement learning is a rapidly developing field, and there remain many recent advancements such as new attention mechanisms which have not yet been tested in the domain of molecular optimization.

## B. Autoencoders

In 2006 Hinton and Salakhutdinov showed how advances in computing power allowed for the training of a deep autoencoder which was capable of beating other methods for document classification.[122] The particular type of neural network they used was a stack of restricted Boltzmann machines, an architecture which would later be called a "deep Boltzmann machine".[123] While deep Boltzmann machines are theoretically powerful, they are computationally expensive to train and impractical for many tasks. In 2013 Kingma et al. introduced the variational autoencoder (VAE),[21] which was used in 2016 by Bombarelli et al. to create the first machine learning based generative model for molecules.[23]

### 1. Variational autoencoders (VAEs)

VAEs are derived mathematically from the theory of variational inference and are only called autoencoders because the resulting architecture has the same high level structure as a classical autoencoder. VAEs are fundamentally a latent variable model $p(\boldsymbol{x}, \boldsymbol{z}) = p_\theta(\boldsymbol{x}|\boldsymbol{z})p(\boldsymbol{z})$ which consists of latent variables $\boldsymbol{z}$ drawn from a prespecified prior $p(\boldsymbol{z})$ and passed into a decoder $p_\theta(\boldsymbol{x}|\boldsymbol{z})$ parametrized by parameters $\theta$. To apply maximum likelihood learning to such a model we like to maximize the probability of each observed datapoint $p(\boldsymbol{x}) = \int p_\theta(\boldsymbol{x}|\boldsymbol{z})p(\boldsymbol{z})d\boldsymbol{z}$ for all datapoints in our training data. However for complicated models with many parameters $\theta$

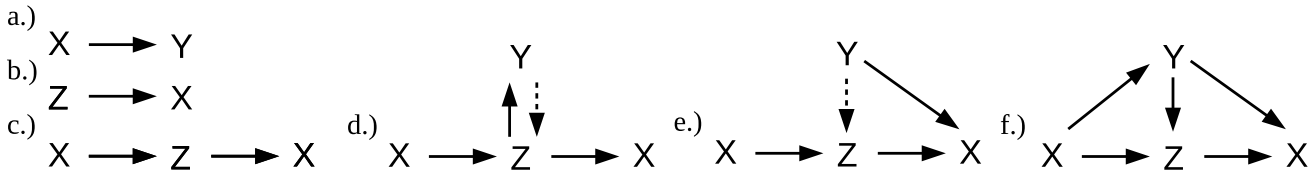


FIG. 2. Different deep learning approaches visualized as graphical models. Figure adapted from Kang et al.[87] Solid lines represent explicit conditional dependencies while dashed lines represent implicit conditional dependencies (arising from the relationship between $X$ and $Y$ inherent in the training data) for which disentanglement may be desired. a.) regression (property prediction) b.) direct generative model c.) autoencoder d.) supervised autoencoder, type 1[23] e.) supervised autoencoder, type 2[86,102] f.) supervised/semisupervised autoencoder, type 3[87]

(like neural networks) this integral is intractable to compute. The method of variational inference instead maximizes a lower bound on $\log p(\boldsymbol{x})$:

$$\log p(\boldsymbol{x}) \geq \mathbb{E}_{\boldsymbol{z}\sim q_\phi(\boldsymbol{z}|\boldsymbol{x})}\left[\log \frac{p_\theta(\boldsymbol{x}|\boldsymbol{z})p(\boldsymbol{z})}{q_\phi(\boldsymbol{z}|\boldsymbol{x})}\right] \tag{9}$$

where $q_\phi(\boldsymbol{z}|\boldsymbol{x})$ is an estimate of posterior distribution $p(\boldsymbol{z}|\boldsymbol{x}) = p_\theta(\boldsymbol{x}|\boldsymbol{z})p(\boldsymbol{z})/p(\boldsymbol{x})$. The right hand side of eqn. 9 is called the "negative variational free energy" or "evidence lower bound" (ELBO) and can be written as:

$$\mathcal{L}_{\theta,\phi}(\boldsymbol{x}) = \mathbb{E}_{\boldsymbol{z}\sim q_\phi(\boldsymbol{z}|\boldsymbol{x})}[\log p_\theta(\boldsymbol{x})] - D_{\mathrm{KL}}(q_\phi(\boldsymbol{z}|\boldsymbol{x}), p_\theta(\boldsymbol{z}|\boldsymbol{x})) \tag{10}$$

Here we encounter the Kullback-Leibler (KL) divergence:

$$D_{\mathrm{KL}}(q(\boldsymbol{z}), p(\boldsymbol{z})) \equiv \int q(\boldsymbol{z}) \log \frac{q(\boldsymbol{z})}{p(\boldsymbol{z})} d\boldsymbol{z} \tag{11}$$

After several manipulations, eqn. 10 can be written as

$$\begin{aligned}\mathcal{L}_{\theta,\phi}(\boldsymbol{x}) &= \mathbb{E}_{\boldsymbol{z}\sim q_\phi(\boldsymbol{z}|\boldsymbol{x})}[\log p(\boldsymbol{z},\boldsymbol{x})] + H[q_\phi(\boldsymbol{z}|\boldsymbol{x})] \\ &= \mathbb{E}_{\boldsymbol{z}\sim q_\phi(\boldsymbol{z}|\boldsymbol{x})}[\log p_\theta(\boldsymbol{x}|\boldsymbol{z})] - D_{\mathrm{KL}}[q_\phi(\boldsymbol{z}|\boldsymbol{x}), p(\boldsymbol{z})]\end{aligned} \tag{12}$$

where $H$ is the (differentiable) entropy. The loss function for the variational autoencoder for examples $\boldsymbol{x}$ in our training dataset $\mathcal{Z}$ is:

$$L_{\theta,\phi} = \sum_{\boldsymbol{x}\in\mathcal{Z}} -\mathcal{L}_{\theta,\phi}(\boldsymbol{x}) \tag{13}$$

In a VAE, during training first $q_\phi(\boldsymbol{z}|\boldsymbol{x})$ (the encoder) generates a $\boldsymbol{z}$. Then the decoder $p_\theta(\boldsymbol{x}|\boldsymbol{z})$ model attempts to recover $\boldsymbol{x}$. Training is done using backpropagation and the loss function (eqn. 13) which tries to maximize $\mathcal{L}(\boldsymbol{x},\theta,\phi)$. This corresponds to maximizing the chance of reconstruction $p_\theta(\boldsymbol{x}|\boldsymbol{z})$ (the first term) but also minimizing the KL-divergence between $q_\phi(\boldsymbol{z}|\boldsymbol{x})$ and the prior distribution $p(\boldsymbol{z})$. Typically the prior is chosen to be a set of independent unit normal distributions and the encoder is assumed to be a factorized multidimensional normal distribution:

$$\begin{aligned}p(\boldsymbol{z}) &= \mathcal{N}(\boldsymbol{z}, 0, \boldsymbol{I}) \\ q_\phi(\boldsymbol{z}|\boldsymbol{x}) &= \mathcal{N}(\boldsymbol{z}, \boldsymbol{\mu}(\boldsymbol{x}), \mathrm{diag}(\boldsymbol{\sigma}^2(\boldsymbol{x})))\end{aligned} \tag{14}$$

The encoder $q_\phi(\boldsymbol{z}|\boldsymbol{x})$ is typically a neural network with parameters $\phi$ used to find the mean and variance functions in $\mathcal{N}(\boldsymbol{z}, \boldsymbol{\mu}(\boldsymbol{x}), \mathrm{diag}(\boldsymbol{\sigma}^2(\boldsymbol{x})))$. The resulting "Gaussian VAE" has the advantage that the KL-divergence can be computed analytically. The parameters $\theta$ and $\phi$ in the decoder and encoder are all learned via backpropagation. There are several important innovations which have been developed to streamline backpropagation and training which are described in detail elsewhere.[21,110,124]

There are several reasons that variational autoencoders perform better than classical autoencoders. Since the latent distribution is probabilistic, this introduces noise which intuitively can be seen as a type of regularization that forces the VAE to learn more robust representations. Additionally, specifying that the latent space must be a Gaussian leads to a much smoother latent space which makes optimization much easier and also leads to fewer "holes" in the distribution corresponding to invalid or bad molecules. VAEs therefore are useful for interpolation between points corresponding to molecules in the training data.

In the molecular autoencoder of Gómez-Bombarelli et al. each SMILES $\boldsymbol{x}$ is converted to a one-hot representation and a convolutional neural network is used to find the parameters of the Gaussian distribution $q_\phi(\boldsymbol{z}|\boldsymbol{x})$.[23] The decoder in the molecular autoencoder is an RNN, but in contrast to pure RNN models, where high rates of valid SMILES generation have been reported (94-98 %),[70,73,76] the molecular autoencoder generates far fewer valid SMILES. The valid SMILES rate was found to vary greatly between $\approx 75\%$ for points near known molecules to only 4% for randomly selected latent points.[23] Kusner et al. report an average valid decoding rate of only 0.7% using a similar VAE architecture.[56] These low decoding rates are not a fatal issue however simply because a validity checker (such as found in RDKit) can easily be used to throw out invalid SMILES during generation. However, the low rate of validity suggests fundamental issues in quality of the learned latent representation. As mentioned previously, higher rates of SMILES validity have been achieved by representing SMILES in terms of rules from a context-free grammar (CFG).[56,57] Kusner et al. achieved somewhat higher rates of SMILES generation using a CFG (7.2%, as described in

the supplementary information of Kusner et al.[56]). Further work by Dai et al. added additional "semantic" constraints on top of a CFG yielding a higher rate of valid SMILES (43.5%).[57] Janz et al. recently proposed using Bayesian active learning as a method of forcing models to learn what makes a sequence valid, and incorporating this into RNNs in VAEs could lead to higher valid decoding rates.[125]

### 2. Adversarial autoencoders (AAEs)

Adversarial autoencoders are similar to variational autoencoders, but differ in the means by which they regularize the latent distribution by forcing it to conform to the prior $p(\boldsymbol{z})$.[126] Instead of minimizing KL-divergence metric to enforce the generator to output a latent distribution corresponding to a prespecified prior (usually a normal distribution), they use adversarial training with a discriminator $D$ whose job is to distinguish the generator's latent distribution from the prior. The discriminator outputs a probability $p \in (0, 1)$ which predicts the probability samples it sees are from the prior. The objective of the discriminator is *maximize* the following:

$$\mathcal{L}_{\text{adv}} = \mathbb{E}_{\boldsymbol{x}\sim p_d}[\log D((q_\Theta(\boldsymbol{z}|\boldsymbol{x}))] + \mathbb{E}_{\boldsymbol{x}\sim p_{\boldsymbol{z}}}[\log(1 - D(\boldsymbol{z}))] \tag{15}$$

The overall objective function for the AAE to *minimize* can be expressed as

$$L_{\theta,\phi} = \sum_{\boldsymbol{x}\in\mathcal{Z}} -\mathbb{E}_{\boldsymbol{x}\sim p_d}[\log p_\theta(\boldsymbol{x}|q_\phi(\boldsymbol{z}|\boldsymbol{x}))] - \mathcal{L}_{\text{adv}} \tag{16}$$

### 3. Supervised VAEs/AAEs for property prediction & optimization

In supervised VAEs, target properties $\boldsymbol{y}$ for each molecule are incorporated into the generator in addition to the SMILES strings or other molecular representation. Figure 2 shows several different ways this can be done, representing the generative models as graphical models. Everything we discuss in this section can also be applied to AAEs,[126] but we restrict our discussion to VAEs for simplicity.

In the work by Gómez-Bombarelli et al they attached a neural network (multilayer perceptron) to the latent layer and jointly trained the neural network to predict property values $y$ and the VAE to minimize reconstruction loss. The advantage of supervised VAEs is that the generator learns a good latent representation both for property prediction and reconstruction. With the property predictor trained, it becomes possible to do property optimization in the latent space, by either using Gaussian process optimization or gradient ascent. Interestingly, in supervised VAEs a particular direction in the latent space always became correlated with the property value $y$, while this was never observed in unsupervised VAEs. When one desires to do optimization, Gómez-Bombarelli et al. argue for using a Gaussian process model as the property predictor instead of a neural network, because it generates a smoother landscape.[23] The specific procedure they used was to first train a VAE using supervised training with a neural network property predictor and then train a Gaussian process model separately using the latent space representations of the training molecules as input. They then use the Gaussian process model for optimization, and they showed it was superior to a genetic optimization algorithm and random search in the latent space. Since that work, several other researchers have used Gaussian process regression to perform optimization in the latent space of a VAE.[94,101,127]

Two other types of supervised VAEs are shown in figure 2, which we call "type 2" and "type 3". Unlike the autoencoder frame work discussed in the previous section, these two types of autoencoder can be used for conditional generation. In "type 3" supervised VAEs the ELBO term in the objective function (eqn. 12) becomes:[87]

$$\mathbb{E}_{\boldsymbol{z}\sim q_\phi(\boldsymbol{z}|\boldsymbol{x},\boldsymbol{y})}[\log p_\theta(\boldsymbol{x}|\boldsymbol{y},\boldsymbol{z})] - D_{\text{KL}}[q_\phi(\boldsymbol{z}|\boldsymbol{x},\boldsymbol{y})||p(\boldsymbol{z},\boldsymbol{y})] \tag{17}$$

Kang et al. assume that the property values have a Gaussian distribution. Type 3 VAEs are particularly useful when $\boldsymbol{y}$ is known for only some of the training data (a semi-supervised setting). In semi-supervised VAEs, the generator is tasked with predict $\boldsymbol{y}$ and is trained on the molecules where $\boldsymbol{y}$ is known and makes a best guess prediction for the rest. In effect, when $\boldsymbol{y}$ is not known, it becomes just another latent variable and a different objective function is used (for details, see Kang et al.[87]).

In Type 2 VAEs, property data $\boldsymbol{y}$ is embedded directly into the latent space during training.[86,102] Supervised and semi-supervised VAEs can both be used for conditional sampling, and thus are sometimes called "conditional VAEs". In the traditional way of doing conditional sampling, $\boldsymbol{y}$ is specified and then one samples from the prior $p(\boldsymbol{z})$. Then one samples from the generator $p_\theta(\boldsymbol{x}|\boldsymbol{y},\boldsymbol{z})$. In the case of Type 1 and Type 2 VAEs, however, there is an issue pointed out by Polykovskiy et al. which they call "entanglement".[102] The issue is that when sampling we assumed that $p(\boldsymbol{z})$ is independent of $p(\boldsymbol{y})$. However, the two distributions are actually "entangled" by the implicit relationship between $x$ and $\boldsymbol{y}$ which is in the training data (this is show by a dashed line in figure 2). For consistency, one should be sampling from $p(\boldsymbol{z}|\boldsymbol{y})$. Polykovskiy et al. developed two "disentanglement" approaches to ameliorate this issue: learning $p(\boldsymbol{z}|\boldsymbol{y})$ and forcing all $p(\boldsymbol{z}|\boldsymbol{y})$ to match $p(\boldsymbol{z})$.[102]

When generating molecules with an RNN, we previously discussed sampling from the model's distribution by simply running the model and taking either the token with the maximum probability or using a multinomial sampler at each step of the sequence generation. When sampling from the generator of a conditional VAE, we wish to know what the model says is the likely molecule given $\boldsymbol{y}$ and $\boldsymbol{z}$, since we are interested in focusing on the

molecules the model predicts are most likely to be associated with a particular set of properties:

$$\hat{\boldsymbol{x}} = \arg\max_{\boldsymbol{x}} p_\theta(\boldsymbol{x}|\boldsymbol{y}, \boldsymbol{z}) \tag{18}$$

Taking the most likely token at each step (the "greedy" method) is only a rough approximation to $\hat{\boldsymbol{x}}$. Unfortunately, completing the optimization in eqn. 18 is a computationally intractable problem because the space of sequences grows exponentially with the length of the sequence. However, a variation on the greedy method called "beam search" can be used to get an approximation of $\hat{\boldsymbol{x}}$.[87,128] In brief, beam search keeps the top $K$ most likely (sub)sequences at each step of the generation.

### C. Generative adversarial networks (GANs)

The key idea underlying GANs is to introduce a discriminator network whose job is to distinguish whether the molecule it is looking at was generated by the generative model or came from the training data. In GAN training, the objective of the generative model becomes to try to fool the discriminator rather than maximizing likelihood. There are theoretical arguments and growing empirical evidence showing that GAN models can overcome some of the well known weaknesses of maximum likelihood based training. However, there are also many technical difficulties which plague GAN training and getting GANs to work well typically requires careful hyperparameter tuning and implementation of several non-obvious "tricks".[129] GANs are a rapidly evolving research area, and given space limitations we can only touch on several of the key developments here.

The original paper on GANs (Goodfellow et al 2014) introduced the following objective function:[22]

$$\begin{aligned}\min_G \max_D V(D,G) =& \mathbb{E}_{\boldsymbol{x}\in p_d(\boldsymbol{x})}[\log D(\boldsymbol{x})] \\ &+ \mathbb{E}_{\boldsymbol{z}\in p_{\boldsymbol{z}}(\boldsymbol{z})}[\log(1 - D(G(\boldsymbol{z}))]\end{aligned} \tag{19}$$

Here $p_d(\boldsymbol{x})$ is the data distribution. This form of the objective function has a nice interpretation as a two person minimax game. However, this objective function is rarely used in practice for a few reasons. Firstly, as noted in the original paper, this objective function does not provide a very strong gradient signal when training starts because then $\log(1 - D(G(\boldsymbol{z}))$ saturates (goes to negative infinity) and the numerical derivative becomes impossible to calculate. Still, understanding this objective function can help understand how generative modeling with GAN training can be superior to maximum likelihood based generative modeling. For a fixed $G$, the optimal discriminator is:

$$D^*_G(\boldsymbol{x}) = \frac{p_d(\boldsymbol{x})}{p_d(\boldsymbol{x}) + p_{\theta_G}(\boldsymbol{x})} \tag{20}$$

If we assume $D = D^*_G$, then the objective function $\mathcal{C}(G)$ for the generator can be expressed as:[22]

$$\mathcal{C}(G) = -\log(4) + 2D_{\mathrm{JS}}(p_d, p_{\theta_G}) \tag{21}$$

Where $D_{\mathrm{JS}}(p_d, p_{\theta_G})$ is the Jensen-Shannon divergence:

$$\begin{aligned}D_{\mathrm{JS}}(p_d, p_{\theta_G}) =& \frac{1}{2} D_{\mathrm{KL}}\left(p_d \middle|\middle| \frac{p_d + p_{\theta_G}}{2}\right) \\ &+ \frac{1}{2} D_{\mathrm{KL}}\left(p_{\theta_G} \middle|\middle| \frac{p_d + p_{\theta_G}}{2}\right)\end{aligned} \tag{22}$$

Here $D_{\mathrm{KL}}(p, q)$ is the Kullback-Leibler (KL) divergence. Maximizing the log-likelihood is equivalent to minimizing the forward KL divergence $D_{\mathrm{KL}}(p_d, p_{\theta_G})$.[124] To better understand what this means, we can rewrite the equation for KL divergence (eqn. 11) in a slightly different way:

$$D_{\mathrm{KL}}(p_d, p_{\theta_G}) = \int p_d(\boldsymbol{z})(\log p_d(\boldsymbol{z}) - \log p_{\theta_G}(\boldsymbol{z}))d\boldsymbol{z} \tag{23}$$

This shows us that KL divergence captures the difference between $p_d$ and $p_{\theta_G}$ weighted by $p_d$. Thus one of the weaknesses of maximum likelihood is that $p_{\theta_G}$ may have significant deviations from $p_d$ when $p_d \approx 0$. To summarize, the forward KL divergence ($D_{\mathrm{KL}}(p_d, p_{\theta_G})$) punishes models that underestimate the data distribution, while the reverse KL divergence ($D_{\mathrm{KL}}(p_{\theta_G}, p_d)$) punishes models that overestimate the data distribution. Therefore we see that eqn. 21, which contains both forward and backwards KL terms, takes a more "balanced" approach than maximum likelihood, which only optimizes forward KL divergence. Optimizing reverse KL divergence directly requires knowing an explicit distribution for $p_d$, which usually is not available. In effect, the discriminator component of the GAN works to learn $p_d$, and thus GANs provide a way of overcoming this issue.

As noted before, the GAN objective function given in eqn. 19, however, does not provide a good gradient signal when training first starts since typically $p_d$ and $p_{\theta_G}$ have little overlap to begin with. Empirically, this occurs because data distributions typically lie on a low dimensional manifold in a high dimensional space, and the location of this manifold is not known in advance. The Wasserstein GAN (WGAN) is widely accepted to provide a better metric for measuring the distance between $p_d$ and $p_{\theta_G}$ than the original GAN objective function and results in faster and more stable training.[130] The WGAN is based on the Wasserstein metric (also called the "Earth mover's distance") which can be informally understood by imagining the two probability distributions $p_d$ and $p_{\theta_G}$ to be piles of dirt, and the distance between them to be the number of buckets of dirt that need to be moved to transform one to the other, times the sum of the distances each bucket must be moved. Mathematically this is expressed as:

$$W(p, q) = \inf_{\gamma\in\Pi(p,q)} \mathbb{E}_{(x,y)\in\gamma}||x - y|| \tag{24}$$

$\Pi(x, y)$ can be understood to be the optimal "transport plan" explaining how much probability mass is moved from $x$ to $y$. Another feature of the WGAN is the introduction of a "Lipschitz constraint" which clamps the weights of the discriminator to lie in a fixed interval. The Lipschitz constraint has been found to result in a more reliable gradient signal for the generator and improve training stability. Many other types of GAN objective function have been developed which we do not have room to discuss here. For a review of the major GAN objective functions and architectures, see Kurach et al.[131]

Several papers have emerged so far applying GANs to molecular generation- Guimares et al. (ORGAN),[95] Sánchez-Lengling et al. (ORGANIC),[100] De Cao & Kipf (MolGAN),[49,62] and Putin et al. (RANC, ATNC).[96,97] The Objective-Reinforced GAN (ORGAN) of Guimares et al. uses the SMILES molecular representation and an RNN (LSTM) generator and a CNN discriminator.[95] The architecture of the ORGAN is taken from the SeqGAN of Yu et al.[132] and uses a WGAN. In ORGAN, the GAN objective function is modified by adding an additional "objective reinforcement" term to the generator RNN's reward function which biases the RNN to produce molecules with a certain objective property or set of objective properties. Typically the objective function returns a value $R(S) \in [0, 1]$. The reward for a SMILES string $S$ becomes a mixture of two terms:

$$R(S) = \lambda D(S) + (1 - \lambda))R(S) \tag{25}$$

where $\lambda \in [0, 1]$ is a tuneable hyperparameter which sets the mixing between rewards for fooling the discriminator and maximizing the objective function. The proof of concept of the ORGAN was demonstrated by optimizing three quick-to-evaluate metrics which can be computed with RDKit- druglikeliness, synthesizability, and solubility.

#### 1. The perfect discriminator problem and training instabilities

GAN optimization is a saddle point optimization problem, and such problems are known to be inherently unstable. If gradients for one part of the optimization dominate, optimizers can run or "spiral" away from the saddle point so that either the generator or the discriminator achieves a perfect score. The traditional approach to avoiding the perfect discriminator problem, taken by Guimares et al. and others, is to do additional MLE pretraining with the generator to strengthen it and then do $m$ gradient updates for the generator for every one gradient update for the discriminator. In this method, $m$ must be tuned to balance the discriminator and generator training. A different, more dynamic method for balancing the discriminator and generator was invented by Kardurin et al. in their work on the DruGAN AAE.[69] They introduce a hyperparameter $0.5 < p < 1$ which sets the desired "discriminator power". Then, after each training step, if the discriminator correctly labels samples from the generator with probability less than $p$, the discriminator is trained, otherwise the generator is trained. Clearly $p$ should be larger than 0.5 since the discriminator should do better than random chance in order to challenge the generator to improve. Empirically, they found $p = 3/5$ to be a good value.

Putin et al. show that the ORGANIC model[100] with its default hyperparameters suffers from a perfect discriminator problem during training, leading to a plateauing of the generator's loss.[96] To help solve these issues, Putin et al. implemented a differentiable neural computer (DNC) as the generator.[96] The DNC (Graves et al, 2016)[133] is an extension of the neural Turing machine (Graves et al 2014)[134] that contains a differentiable memory cell. A memory cell allows the generator to memorize key SMILES motifs, which results in a much "stronger" generator. They found that the discriminator never achieves a perfect score when training against a DNC. The strength of the DNC is also shown by the fact that it has a higher rate of valid SMILES generation vs. the ORGAN RNN generator (76% vs.
24%) and generates SMILES that are on average twice as long as the SMILES generated by ORGAN. In a subsequent work, Putin et al. also introduced the adversarial threshold neural computer, another architecture with a DNC generator.[97]

Another issue with GANs is mode collapse, where the generator only generates a narrow range of samples. In the context of molecules, an example might be a generator that only generates molecules with carbon rings and less than 20 atoms.

## III. METRICS AND REWARD FUNCTIONS

A key issue in deep generative modeling research is how to quantitatively compare the performance of different generative models. More generally a decline in rigor in the field of deep learning as a whole has been noted by Sculley, Rahimi and others.[135] While the recent growth in the number of researchers in the field has obvious benefits, the increased competition that can result from such rapid growth disincentivizes taking time for careful tuning and rigorous evaluation of new methods with previous ones. Published comparisons are often subtly biased towards demonstrating superior performance for technically novel methods vs. older more conventional methods. A study by Lucic et al. for instance found that in the field of generative adversarial networks better hyperparameter tuning and training lead to most recently proposed methods reaching very similar results.[129,136] Similarly, Melis et al. found that with proper hyperparameter tuning a conventional LSTM architecture could beat several more recently proposed methods for natural language modeling.[137] At the same time, there is a reproducibility crisis afflicting deep learning- codes published side-by-side with papers often give different results than

what was published,[131] and in the field of reinforcement learning it has been found that codes which purport to do the same thing will give different results.[136] The fields of deep learning and deep generative modeling are still young however, and much work is currently underway on developing new standards and techniques for rigorously comparing different methods. In this section we will discuss several of the recently proposed metrics for comparing generative models and the closely related topic of reward function design for molecular optimization.

### A. Metrics for scoring generative models

Theis et al. discuss three separate approaches- log-likelihood, estimating the divergence metric between the training data distribution $p(x)$ and the model's distribution $q(x)$, and human rating by visual inspection (also called the "visual Turing test") .[138,139] Interestingly, they show that these three methodologies measure different things, so good performance under one does not imply good performance under another.[139]

The "inception score" (IS) uses a third-party neural network which has been trained in a supervised manner to do classification.[138] In the original IS, Google's Inception network trained on ImageNet was used as the third-party network. IS computes the expected divergence between the distribution of classes predicted by the third-party neural network with the distribution of classes predicted for the dataset used to train the neural network. The main weakness of IS is that much information about the quality of images is lost by focusing only on classification labels. The Fréchet Inception Distance (FID) builds off the IS by comparing latent vectors obtained from a late-stage layer of a third-party classification network instead of the predictions.[140] Inspired by this, Preuer et al. created the Fréchet ChemNet Distance metric for evaluating models that generate molecules.[141] Unfortunately, there is a lack of agreement on how to calculate the FID- some report the score by comparing training data with generated data, while others report the score comparing a hold out test set with the generated data.[131] Comparing with test data gives a more useful metric which measures generalization ability, and is advocated as a standard by Kurach et al.[131]

In the world of machine learning for molecular property prediction, *MoleculeNet* provides a benchmark to compare the utility of different regression modeling techniques across a wide range of property prediction problems.[142] Inspired by *MoleculeNet*, Polykovskiy and collaborators have introduced the MOlecular SEtS (MOSES) package to make it easier to build and test generative models.[143] To compare the output of generative models, they provide functions to compute Fréchet ChemNet Distance, internal diversity, as well as several metrics which are of general importance for pharmaceuticals: molecular weight, logP, synthetic accessibility score, and the quantitative estimation of drug-likeness. In a similar vein, Benhenda et al. have released the *DiversityNet* benchmark, which was also (as the name suggests) inspired by *MoleculeNet*.[144] Finally, another Python software package called *GuacaMol* has also been released which contains 5 general purpose benchmarking methods and 20 "optimization specific" benchmarking methods for drug discovery.[145] One unique feature of *GuacaMol* is the ability to compute KL-divergences between the distributions from generated molecules and training molecules for a variety of physio chemical descriptors.

Recently in the context of generative modeling of images with GANs, Jiwoong Im et al. have shown significant pitfalls to using the Inception Distance metric.[146] As an alternative, they suggest using the same type of divergence metrics that are used during GAN training. This method has been explored recently to quantify generalization performance of GANS[147] and could be of use to the molecular modeling community.

### B. Reward function design

A good reward function is often important for molecular generation and essential for molecular optimization. The pioneering molecular autoencoder work resulted in molecules which were difficult to synthesize or contained highly labile (reactive or unstable) groups such as enamines, hemiaminals, and enol ethers which would rapidly break apart in the body and thus were not viable drugs.[148] Since then, the development of better reward functions has greatly helped to mitigate such issues, but low diversity and novelty remains an issue.[149–151] After reviewing the work that has been done so far on reward function design, we conclude that good reward functions should lead to generated molecules which meet the following desiderata:

1. **Diversity**- the set of molecules generated is diverse enough to be interesting.
2. **Novelty**- the set of molecules does not simply reproduce molecules in the training set.
3. **Stability**- the molecules are stable in the target environment and not highly reactive.
4. **Synthesizability**- the molecules can actually be synthesized.
5. **Non-triviality**- the molecules are not degenerate or trivial solutions to maximizing the reward function.
6. **Good properties**- the molecules have the properties desired for the application at hand.

#### 1. Diversity & novelty

A diversity metric is a key component of any reward function, especially when using a GAN, where it helps

counter the issue of mode collapse to a non-diverse subset. Given a molecular similarity metric between two molecules $T(x_1, x_2) \in [0, 1]$ the diversity of a generated set $\mathcal{G}$ can be defined quite simply as:

$$r_{\text{diversity}} = 1 - \frac{1}{|\mathcal{G}|} \sum_{(x_1,x_2)\in\mathcal{G}\times\mathcal{G}} D(x_1, x_2) \tag{26}$$

A popular metric is the Tanimoto similarity between two extended-connectivity fingerprint bit vectors.[143] Since the diversity of a single molecule does not make sense, diversity rewards are calculated on mini-batches during mini-batch stochastic gradient descent training. Eqn. 26 is called "internal diversity". An alternative which compares the diversity of the generated set with the diversity of the training data is the nearest neighbor similarity (SNN) metric:[143]

$$r_{\text{SSN}} = \frac{1}{|\mathcal{G}|} \sum_{x_G\in\mathcal{G}} \max_{x_D\in\mathcal{D}} D(x_G, x_G) \tag{27}$$

Of course, too much diversity can also be an issue. One option is to use the following negative reward which biases the generator towards matching the diversity of the training data:

$$R_{\text{diversity mismatch}} = -\left| r^{\text{generated}}_{\text{diversity}} - r^{\text{training}}_{\text{diversity}} \right| \tag{28}$$

Another diversity measure that has been employed is uniqueness, which aims to reduce the number of repeat molecules. The uniqueness reward $R_{\text{uniqueness}} \in (0, 1]$ is defined as:

$$R_{\text{uniqueness}} = \frac{|\text{set}(\mathcal{G})|}{|\mathcal{G}|} \tag{29}$$

Where $|\text{set}(\mathcal{G})|$ is the number of unique molecules in the generated batch $\mathcal{G}$ and $|\mathcal{G}|$ is the total number of molecules.

Novelty is just as important as diversity since a generator which just generates the training dataset over and over is of no practical utility. Guimares et al. define the "soft novelty" for a single molecule as:[95]

$$R_{\text{novelty}} = \begin{cases} 1 & \text{If } x \text{ is not in the training set} \\ 0.3 & \text{If } x \text{ is in the training set} \end{cases} \tag{30}$$

When measuring the novelty of molecules generated post-training to get an overall novelty measure for the model, it is important to do so on a hold-out test set $\mathcal{T}$. Then one can look at how many molecules in a set of generated molecules $\mathcal{G}$ appear in $\mathcal{T}$ and use a novelty reward such as:[73]

$$r_{\text{novel}} = 1 - \frac{|\mathcal{G} \cap \mathcal{T}|}{|\mathcal{T}|} \tag{31}$$

which gives the fraction of generated molecules not appearing in the test set. The diversity of the generated molecules and how they compare to the diversity of the training set can also be visualized by generating fingerprint vectors (which typically have dimensionalities of $d > 100$) and then projecting them into two dimensions using dimensionality reduction techniques. The resulting 2D point cloud can then be compared with the corresponding points from the training set and/or a hold out test set. There are many possible dimensionality reduction techniques to choose from —Yoshikawa et al.[150] use the ISOMAP method,[152] Merk et al.[79] use multidimensional scaling, and Selger et al.[73] use t-SNE projection.[153]

Interpolation between training molecules may be a useful way to generate molecules post-training which are novel, but not too novel as to be unstable or outside the intended property manifold. In the domain of image generation, GANs seem to excel at interpretation vs. VAEs, for reasons that are not yet fully understood. For instance with GANs trained on natural images, interpolation can be done between a $\boldsymbol{z}$ point corresponding to a frowning man to a point $\boldsymbol{z}'$ corresponding to a smiling woman, and all of the intervening points result in images which make sense.[154] Empirically most real world high dimensional datasets are found to lie on a low density manifold.[155] Ideally, the dimensionality of the latent space $p(\boldsymbol{z})$ used in a GAN, VAE, or AAE will correspond to the dimensionality of this manifold. If the dimensionality of $p(\boldsymbol{z})$ is higher than the intrinsic dimensionality of the data manifold, then interpolation can end up going "off manifold" into so-called "dead zones". For high dimensional latent spaces with a Gaussian prior, most points will lie on thin spherical shell. In such cases it has been found empirically that better results can be found by doing spherical linear interpolation or *slerp*.[156] The equation for *slerp* interpolation between two vectors $\boldsymbol{v}_1$ and $\boldsymbol{v}_2$ is

$$slerp(\boldsymbol{v}_1, \boldsymbol{v}_2, t) = \frac{\sin((1-t)\theta)}{\sin(\theta)}\boldsymbol{q}_1 + \frac{\sin(t\theta)}{\sin(\theta)}\boldsymbol{q}_2 \tag{32}$$

where $\theta = \arccos(\boldsymbol{v}_1 \cdot \boldsymbol{v}_2)$ and $0 \leq t \leq 1$ is the interpolation parameter.

Another option for generating molecules close to training molecules but not too close is to have a reward for being similar to the training data but not too similar. Olivecrona et al. use a reward function $R_s(S) \in [-1, 1]$ of the form:[72]

$$R_s(S) = 1 - 2\frac{\min(\text{Sim}(S, T), k)}{k} \tag{33}$$

here $S$ is the input SMILES and $T$ is the target SMILES, and $\text{Sim}(S, T) \in [0, 1]$ is similarity scoring function which computes fingerprint-based similarity between the two molecules. $k$ is a tuneable cutoff parameter which sets the maximum similarity accepted. This type of reward can be particularly useful for generating focused libraries of molecules very similar to a single target molecule or a small set of "actives" which are known to bind to a receptor. Note that most generative models can be tweaked to generate molecules close to a given molecule. For

instance, with RNNs, one can do "fragment growing", which allows molecular designers to explore molecules which share a predefined scaffold.[71] Similarly, with reinforcement learning one can simply start the agent with a particular model and let it add or remove bonds. Finally, with a VAE one can find the latent representation for a given molecule and then inject a small amount of Gaussian noise to generate "nearby" molecules.[88]

### 2. Stability and synthesizability

Enforcement of synthesizability has thus far mainly been done using the synthetic accessibility (SA) score developed by Ertl & Schuffenhauer,[157] although other synthesizability scoring functions exist.[158,159] The model underlying the SA score was designed and fit specifically to match scores from synthetic chemists on a set of drug molecules, and therefore may be of limited applicability to other types of molecules. When using SA score as a reward in their molecular autoencoder, Gómez-Bombarelli et al. found that it still produced a large number of molecules with unrealistically large carbon rings. Therefore, they added an additional "RingPenalty" term to penalize rings with more than six carbons. In the ORGANIC GAN code, Sánchez-Lengling et al. added several penalty terms to the original SA score function, and also developed an additional reward for chemical symmetry, based on the observation that symmetric molecules are typically easier to synthesize.[100]

For drug molecules, the use of scoring functions developed to estimate how "drug-like" or "natural" a compound is can also help improve the synthesizability, stability, and usefulness of the generated molecules.[95] Examples of such functions include Lipinski's Rule of Five score,[5] the natural product-likeness score,[160] the quantitative estimate of drug-likeness,[161] and the Muegge metrics.[162,163] Another option of particular utility to drug discovery is to apply medicinal chemistry filters either during training or post-training to tag unstable, toxic, or unsynthesizable molecules. For drug molecules, catalogs of problematic functional groups to avoid have been developed in order to limit the probability of unwanted side-effects.[164] For energetic molecules and other niche domains an analogous set of functional groups to avoid has yet to be developed.

Finally, many software packages exist for checking molecules which may be integrated into training or as a post-training filter. For example Popova et al. use the ChemAxon structure checker software to do a validity check on the generated molecules.[77] Bjerrum et al. use the Wiley ChemPlanner software post-training to find synthesis routes for 25-55% of the molecules generated by their RNN.[70] Finally, Sumita et al. check for previously discovered synthetic routes for their generated molecules using a SciFinder literature search.[78]

### 3. Rewards for good properties

Because they are called often during training, reward functions should be quick to compute, and therefore fast property estimators are called for. Examples of property estimation functions which are fast to evaluate are the estimated octanol-water partition coefficient (LogP), and the quantitative measure of drug-likeness (QED),[161] both of which can be found in the open source package RDKit.[51]

Since physics based prediction is often very computationally intensive, a popular approach is to train a property predicting machine learning model ahead of time. There are some predictive models available "off the shelf" as well, for instance a collection of predictive models for ADME (absorption, distribution, metabolism, and excretion) called "SwissADME" was recently published.[165] It has been shown that traditional physics-based simulations can be used —Sumita et al. optimize absorption wavelength by converting SMILES strings into 3D structures using RDKit and then calculating their UV-VIS absorption spectra on-the-fly with time-dependent density functional theory (TD-DFT). Instead of the obvious reward function $-\alpha|\lambda^* - \lambda|$, where $\lambda^*$ is the target wavelength, they used the following:[78]

$$R = \begin{cases} \frac{-\alpha|\lambda^*-\lambda|}{1+\alpha|\lambda^*-\lambda|} & \text{If DFT calculation successful} \\ -1 & \text{If DFT calculation fails} \end{cases} \tag{34}$$

From the molecules generated by their RNN, Sumita et al. selected six molecules for synthesis and found that 5/6 exhibited the desired absorption profiles.[78]

A reward function which has been used by several different researchers to generate drug molecules is:

$$J(S) = \text{logP}(S) - \text{SA}(S) - \text{RingPenalty}(S) \tag{35}$$

Yang et al. add an additional penalty for generating invalid SMILES which could be used more broadly:[74]

$$R(S) = \begin{cases} \frac{J(S)}{1+|J(S)|} & \text{for valid SMILES} \\ -1.0 & \text{for invalid SMILES} \end{cases} \tag{36}$$

In the context of training the ORGAN architecuture, Guimares et al. found that rotating the reward function metric from epoch to epoch had some advantages to using all metrics at once.[95] In other words, in one epoch the rewards may just be for diversity, while in the next they would just be for synthesizability, and so on. This idea could likely be explored further.

## IV. PROSPECTIVE AND FUTURE DIRECTIONS

In this review we have tried to summarize the current state of the art for generative modeling of molecules using deep learning. The current literature is composed of a rich array of representation strategies and model architectures. As in many areas of generative modeling

and deep learning, the present day work is largely empirical in nature. As our mathematical understanding of the landscape of generative models improves, so too will our ability to select the best approaches to a particular problem. There are many promising new techniques and architectures from deep generative modeling and optimization more broadly which are ripe to be transferred to the world of molecules. For example, for sequence modeling the Maximum Likelihood Augmented Discrete GAN (MaliGAN) has been shown to be superior to the SeqGAN on which ORGAN is based.[166] With RNNs, recently developed attention mechanisms and external memory cells offer a possible avenue to improve SMILES string generation.[167]

It is worth noting that the latest genetic algorithm based methods can still compete with today's deep learning based methods. Yoshikawa et al. developed a genetic algorithm which makes edits to SMILES and uses population-based evolutionary selection to generate molecules with high binding affinity as calculated via docking (rDock).[38] They compared their method with three other deep-learning based methods (CVAE[23], GVAE[56], and ChemTS[74]) for optimizing a particular reward function (eqn. 35). They found that with computer time fixed to eight hours, their method performed better or comparable to the deep learning methods.

In our discussion of GANs we highlighted an important way in which GANs are superior to maximum likelihood based methods- namely that they can avoid the "filling in" problem which occurs with maximum likelihood where the model's distribution ends up non-zero where the data distribution is zero. Another point is that the theorems on which the maximum likelihood methodology is based only hold true in the limit of infinite samples.[168] In general it appears that GANs can be trained with far fewer samples than maximum likelihood based methods- this can be seen by looking at the $N_{\text{train}}$ values in table II. In addition to their benefits, we also touched on several difficulties with GANs- small starting gradient, training instabilities, the perfect discriminator problem, and mode collapse. However, we also cited possible remedies for each of these issues and we expect more remedies to be developed in the future.

There are several major trends we have observed which present day practitioners and those entering the field should be cognizant of:

**New representation methods** SMILES based techniques are quickly being replaced with techniques that work directly with chemical graphs and three dimensional chemical structures. Directly working with chemical graphs, either by using chemistry-preserving graph operations or tensor representations avoids the problems associated with requiring deep generative models to learn SMILES syntax. At the same time, there is also growing interest in generative models which can generate 3D equilibrium structures, since in many applications the specific 3D positions of atoms can be important.

**Better reward functions** As mentioned earlier, reward function design is a critical component to molecular generation and optimization. We expect future work will use more sophisticated reward functions which combine multiple objectives into a single reward function. Using multiple reward functions and multi-objective reinforcement learning is also a promising approach.[65]

**Pure reinforcement learning approaches** The deep reinforcement learning work of Zhou et al. demonstrated superior molecular optimization performance when compared with the Junction Tree VAE,[91] ORGAN,[95] and Graph Convolutional Policy Network[64] approaches when optimizing the logP and QED metrics.[65] The work of Zhou et al. is notable as it is the first to take a pure RL approach with no pretrained generator. We believe much future work in molecular optimization will proceed in this direction since many application areas have limited training data available.

**Hierarchical modeling** Hierarchical representation schemes will allow for efficient generative modeling of large molecules (such as proteins) as well as complex systems such as polymers, metal organic frameworks, and molecular crystals. Generative modeling techniques will also be useful not just for optimizing molecules but also optimizing the structures and systems in which they are embedded. GANs have recently been applied to the generation of crystal structures[169] and microstructures.[170–172] Hierarchical GANs[173] may be useful for the generation of many-molecule complexes or for the simultaneous optimization of both material and geometry in nanomaterials and metamaterials.

**Closing the loop** After the synthesis and characterization of new lead compounds the data obtained can be fed back to improve the models, a process called "closing the loop". More work is needed to develop workflows and methods to interface and integrate generative models into laboratory platforms allow for rapid feedback and cycling. A key challenge is developing useful software and cyberinfrastructure for computational screening.[174] The potential for efficiency improvements via automated AI-assisted synthesis planning and "self-driving" robotic laboratories is quite profound.[14,15,175–177]

## ACKNOWLEDGEMENTS

Support for this work is gratefully acknowledged from the U.S. Office of Naval Research under grant number N00014-17-1-2108 and from the Energetics Technology Center under project number 2044-001. Partial support is also acknowledged from the Center for Engineering Concepts Development in the Department of Mechanical Engineering at the University of Maryland, College Park. We thank Dr. Ruth M. Doherty, Dr. William Wilson, and Dr. Andrey Gorlin for their input and for proofreading the manuscript.

## REFERENCES


[1] Joseph A. DiMasi, Henry G. Grabowski, and Ronald W. Hansen. Innovation in the pharmaceutical industry: New estimates of R&D costs. *Journal of Health Economics*, 47:20 – 33, 2016.

[2] Steven M. Paul, Daniel S. Mytelka, Christopher T. Dunwiddie, Charles C. Persinger, Bernard H. Munos, Stacy R. Lindborg, and Aaron L. Schacht. How to improve R&D productivity: the pharmaceutical industry's grand challenge. *Nature Reviews Drug Discovery*, 9(3):203–214, feb 2010.

[3] Axel Homburg. Remarks on the evolution of explosives. *Propellants, Explosives, Pyrotechnics*, 42(8):851–853, aug 2017.

[4] P. G. Polishchuk, T. I. Madzhidov, and A. Varnek. Estimation of the size of drug-like chemical space based on GDB-17 data. *Journal of Computer-Aided Molecular Design*, 27(8):675–679, Aug 2013.

[5] Christopher A. Lipinski, Franco Lombardo, Beryl W. Dominy, and Paul J. Feeney. Experimental and computational approaches to estimate solubility and permeability in drug discovery and development settings. *Advanced Drug Delivery Reviews*, 23(1-3):3–25, jan 1997.

[6] Ricardo Macarron, Martyn N. Banks, Dejan Bojanic, David J. Burns, Dragan A. Cirovic, Tina Garyantes, Darren V. S. Green, Robert P. Hertzberg, William P. Janzen, Jeff W. Paslay, Ulrich Schopfer, and G. Sitta Sittampalam. Impact of high-throughput screening in biomedical research. *Nature Reviews Drug Discovery*, 10(3):188–195, mar 2011.

[7] Edward O. Pyzer-Knapp, Changwon Suh, Rafael Gómez-Bombarelli, Jorge Aguilera-Iparraguirre, and Alán Aspuru-Guzik. What is high-throughput virtual screening? a perspective from organic materials discovery. *Annual Review of Materials Research*, 45(1):195–216, jul 2015.

[8] Keith T. Butler, Daniel W. Davies, Hugh Cartwright, Olexandr Isayev, and Aron Walsh. Machine learning for molecular and materials science. *Nature*, 559(7715):547–555, jul 2018.

[9] Paul Raccuglia, Katherine C. Elbert, Philip D. F. Adler, Casey Falk, Malia B. Wenny, Aurelio Mollo, Matthias Zeller, Sorelle A. Friedler, Joshua Schrier, and Alexander J. Norquist. Machine-learning-assisted materials discovery using failed experiments. *Nature*, 533(7601):73–76, may 2016.

[10] B. C. Barnes, D. C. Elton, Z. Boukouvalas, D. E. Taylor, W. D. Mattson, M. D. Fuge, and P. W. Chung. Machine Learning of Energetic Material Properties. *arXiv e-prints: 1807.06156*, July 2018.

[11] David Fooshee, Aaron Mood, Eugene Gutman, Mohammadamin Tavakoli, Gregor Urban, Frances Liu, Nancy Huynh, David Van Vranken, and Pierre Baldi. Deep learning for chemical reaction prediction. *Molecular Systems Design & Engineering*, 3(3):442–452, 2018.

[12] Philippe Schwaller, Théophile Gaudin, Dávid Lányi, Costas Bekas, and Teodoro Laino. "found in translation": predicting outcomes of complex organic chemistry reactions using neural sequence-to-sequence models. *Chemical Science*, 9(28):6091–6098, 2018.

[13] Marwin H. S. Segler, Mike Preuss, and Mark P. Waller. Planning chemical syntheses with deep neural networks and symbolic AI. *Nature*, 555(7698):604–610, mar 2018.

[14] Alon B. Henson, Piotr S. Gromski, and Leroy Cronin. Designing algorithms to aid discovery by chemical robots. *ACS Central Science*, 4(7):793–804, jul 2018.

[15] Loïc M. Roch, Florian Häse, Christoph Kreisbeck, Teresa Tamayo-Mendoza, Lars P. E. Yunker, Jason E. Hein, and Alán Aspuru-Guzik. ChemOS: Orchestrating autonomous experimentation. *Science Robotics*, 3(19):eaat5559, jun 2018.

[16] D. Cireşan, U. Meier, and J. Schmidhuber. Multi-column deep neural networks for image classification. In *2012 IEEE Conference on Computer Vision and Pattern Recognition*, pages 3642–3649, June 2012.

[17] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In F. Pereira, C. J. C. Burges, L. Bottou, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 25*, pages 1097–1105. Curran Associates, Inc., 2012.

[18] George E. Dahl, Navdeep Jaitly, and Ruslan Salakhutdinov. Multi-task Neural Networks for QSAR Predictions. *arXiv e-prints:1406.1231*, June 2014.

[19] Geoffrey E. Hinton, Nitish Srivastava, Alex Krizhevsky, Ilya Sutskever, and Ruslan R. Salakhutdinov. Improving neural networks by preventing co-adaptation of feature detectors. *arXiv e-prints: 1207.0580*, July 2012.

[20] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton. Imagenet classification with deep convolutional neural networks. *Commun. ACM*, 60(6):84–90, May 2017.

[21] D. P Kingma and M. Welling. Auto-Encoding Variational Bayes. *arXiv e-prints: 1312.6114*, December 2013.

[22] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In Z. Ghahramani, M. Welling, C. Cortes, N. D. Lawrence, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 27*, pages 2672–2680. Curran Associates, Inc., 2014.

[23] Rafael Gómez-Bombarelli, Jennifer N. Wei, David Duvenaud, José Miguel Hernández-Lobato, Benjamín Sánchez-Lengeling, Dennis Sheberla, Jorge Aguilera-Iparraguirre, Timothy D. Hirzel, Ryan P. Adams, and Alán Aspuru-Guzik. Automatic chemical design using a data-driven continuous representation of molecules. *ACS Central Science*, 4(2):268–276, jan 2018.

[24] Edward J. Griffen, Alexander G. Dossetter, Andrew G. Leach, and Shane Montague. Can we accelerate medicinal chemistry by augmenting the chemist with big data and artificial intelligence? *Drug Discovery Today*, 23(7):1373–1384, jul 2018.

[25] Rafael Gómez-Bombarelli, Jorge Aguilera-Iparraguirre, Timothy D. Hirzel, David Duvenaud, Dougal Maclaurin, Martin A. Blood-Forsythe, Hyun Sik Chae, Markus Einzinger, Dong-Gwang Ha, Tony Wu, Georgios Markopoulos, Soonok Jeon, Hosuk Kang, Hiroshi Miyazaki, Masaki Numata, Sunghan Kim, Wenliang Huang, Seong Ik Hong, Marc Baldo, Ryan P. Adams, and Alán Aspuru-Guzik. Design of efficient molecular organic light-emitting diodes by a high-throughput virtual screening and experimental approach. *Nature Materials*, 15(10):1120–1127, aug 2016.

[26] Peter Bjørn Jørgensen, Murat Mesta, Suranjan Shil, Juan Maria García Lastra, Karsten Wedel Jacobsen, Kristian Sommer Thygesen, and Mikkel N. Schmidt. Machine learning-based screening of complex molecules for polymer solar cells. *The Journal of Chemical Physics*, 148(24):241735, jun 2018.

[27] Daniel C. Elton, Zois Boukouvalas, Mark S. Butrico, Mark D. Fuge, and Peter W. Chung. Applying machine learning techniques to predict the properties of energetic materials. *Scientific Reports*, 8(1), jun 2018.

[28] B. Christopher Rinderspacher and Jennifer M. Elward. Enriched optimization of molecular properties under constraints: an electrochromic example. *Molecular Systems Design & Engineering*, 3(3):485–495, 2018.

[29] Haichen Li, Christopher R. Collins, Thomas G. Ribelli, Krzysztof Matyjaszewski, Geoffrey J. Gordon, Tomasz Kowalewski, and David J. Yaron. Tuning the molecular weight distribution from atom transfer radical polymerization using deep reinforcement learning. *Molecular Systems Design & Engineering*, 3(3):496–508, 2018.

[30] Deepesh Nagarajan, Tushar Nagarajan, Natasha Roy, Omkar Kulkarni, Sathyabaarathi Ravichandran, Madhulika Mishra, Dipshikha Chakravortty, and Nagasuma Chandra. Computational antimicrobial peptide design and evaluation against multidrug-resistant clinical isolates of bacteria. *Journal of Biological Chemistry*, 293(10):3492–3509, dec 2017.

[31] Alex T. Müller, Jan A. Hiss, and Gisbert Schneider. Recurrent neural network model for constructive peptide design. *Jour-*

*nal of Chemical Information and Modeling*, 58(2):472–479, jan 2018.

[32] Francesca Grisoni, Claudia S. Neuhaus, Gisela Gabernet, Alex T. Müller, Jan A. Hiss, and Gisbert Schneider. Designing anticancer peptides by constructive machine learning. *ChemMedChem*, 13(13):1300–1302, may 2018.

[33] Xiaozhou Shen, Tianmu Zhang, Scott Broderick, and Krishna Rajan. Correlative analysis of metal organic framework structures through manifold learning of hirshfeld surfaces. *Molecular Systems Design & Engineering*, 3(5):826–838, 2018.

[34] Yuping He, Ekin D. Cubuk, Mark D. Allendorf, and Evan J. Reed. Metallic metal–organic frameworks predicted by the combination of machine learning methods and ab initio calculations. *The Journal of Physical Chemistry Letters*, 9(16):4562–4569, jul 2018.

[35] Bernard Pirard. The quest for novel chemical matter and the contribution of computer-aidedde novodesign. *Expert Opinion on Drug Discovery*, 6(3):225–231, feb 2011.

[36] Lee-Ping Wang, Alexey Titov, Robert McGibbon, Fang Liu, Vijay S. Pande, and Todd J. Martínez. Discovering chemistry with an ab initio nanoreactor. *Nature Chemistry*, 6(12):1044–1048, nov 2014.

[37] Jérémy Besnard, Gian Filippo Ruda, Vincent Setola, Keren Abecassis, Ramona M. Rodriguiz, Xi-Ping Huang, Suzanne Norval, Maria F. Sassano, Antony I. Shin, Lauren A. Webster, Frederick R. C. Simeons, Laste Stojanovski, Annik Prat, Nabil G. Seidah, Daniel B. Constam, G. Richard Bickerton, Kevin D. Read, William C. Wetsel, Ian H. Gilbert, Bryan L. Roth, and Andrew L. Hopkins. Automated design of ligands to polypharmacological profiles. *Nature*, 492(7428):215–220, dec 2012.

[38] Naruki Yoshikawa, Kei Terayama, Masato Sumita, Teruki Homma, Kenta Oono, and Koji Tsuda. Population-based de novo molecule generation, using grammatical evolution. *Chemistry Letters*, 47(11):1431–1434, nov 2018.

[39] Denis Kuzminykh, Daniil Polykovskiy, Artur Kadurin, Alexander Zhebrak, Ivan Baskov, Sergey Nikolenko, Rim Shayakhmetov, and Alex Zhavoronkov. 3d molecular representations based on the wave transform for convolutional neural networks. *Molecular Pharmaceutics*, mar 2018.

[40] Miha Skalic, José Jiménez Luna, Davide Sabbadin, and Gianni De Fabritiis. Shape-based generative modeling for de-novo drug design. *Journal of Chemical Information and Modeling*, (ja), 2019.

[41] Afshine Amidi, Shervine Amidi, Dimitrios Vlachakis, Vasileios Megalooikonomou, Nikos Paragios, and Evangelia I. Zacharaki. EnzyNet: enzyme classification using 3D convolutional neural networks on spatial representation. *PeerJ*, 6:e4750, may 2018.

[42] Matthew Hirn, Stéphane Mallat, and Nicolas Poilvert. Wavelet scattering regression of quantum chemical energies. *Multiscale Modeling & Simulation*, 15(2):827–863, jan 2017.

[43] Michael Eickenberg, Georgios Exarchakis, Matthew Hirn, and Stephane Mallat. Solid harmonic wavelet scattering: Predicting quantum molecular energy from invariant descriptors of 3D electronic densities. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems 30*, pages 6540–6549. Curran Associates, Inc., 2017.

[44] Nathaniel Thomas, Tess Smidt, Steven Kearnes, Lusann Yang, Li Li, Kai Kohlhoff, and Patrick Riley. Tensor field networks: Rotation- and translation-equivariant neural networks for 3D point clouds, 2018.

[45] Noel M O'Boyle, Michael Banck, Craig A James, Chris Morley, Tim Vandermeersch, and Geoffrey R Hutchison. Open Babel: An open chemical toolbox. *Journal of Cheminformatics*, 3(1):33, 2011.

[46] The Open Babel Package. `http://www.openbabel.org`.

[47] D. Weininger. Smiles, a chemical language and information system. 1. introduction to methodology and encoding rules. *J. Chem. Inf. Comp. Sci.*, 28, 1988.

[48] Eric Jang, Shixiang Gu, and Ben Poole. Categorical Reparameterization with Gumbel-Softmax. *arXiv e-prints:1611.01144*, November 2016.

[49] Nicola De Cao and Thomas Kipf. MolGAN: An implicit generative model for small molecular graphs. *ICML 2018 workshop on Theoretical Foundations and Applications of Deep Generative Models*, 2018.

[50] M. Swain. MolVS. `https://github.com/mcs07/MolVS`.

[51] Greg Landrum. RDKit: Open-source cheminformatics. `http://www.rdkit.org`.

[52] E. Jannik Bjerrum and B. Sattarov. Improving Chemical Autoencoder Latent Space and Molecular De novo Generation Diversity with Heteroencoders. *arXiv e-prints:1806.09300*, June 2018.

[53] E. Jannik Bjerrum. SMILES Enumeration as Data Augmentation for Neural Network Modeling of Molecules. *arXiv e-prints:1703.07076*, March 2017.

[54] Stephen Heller, Alan McNaught, Stephen Stein, Dmitrii Tchekhovskoi, and Igor Pletnev. InChI - the worldwide chemical structure identifier standard. *Journal of Cheminformatics*, 5(1):7, 2013.

[55] Robin Winter, Floriane Montanari, Frank Noe, and Djork-Arn Clevert. Learning Continuous and Data-Driven Molecular Descriptors by Translating Equivalent Chemical Representations. *ChemRxiv preprint*, 7 2018.

[56] M. J. Kusner, B. Paige, and J. M. Hernández-Lobato. Grammar Variational Autoencoder. *arXiv e-prints:1703.01925*, March 2017.

[57] Hanjun Dai, Yingtao Tian, Bo Dai, Steven Skiena, and Le Song. Syntax-directed variational autoencoder for structured data. *arXiv e-prints:1802.08786*, 2018.

[58] Y. Li, O. Vinyals, C. Dyer, R. Pascanu, and P. Battaglia. Learning Deep Generative Models of Graphs. *arXiv e-prints:1803.03324*, March 2018.

[59] Yibo Li, Liangren Zhang, and Zhenming Liu. Multi-objective de novo drug design with conditional graph generative model. *Journal of Cheminformatics*, 10(1), jul 2018.

[60] Masaki Suzuki, Hiroshi Nagamochi, and Tatsuya Akutsu. Efficient enumeration of monocyclic chemical graphs with given path frequencies. *Journal of Cheminformatics*, 6(1), may 2014.

[61] G. B. Goh, C. Siegel, A. Vishnu, N. O. Hodas, and N. Baker. Chemception: A Deep Neural Network with Minimal Chemistry Knowledge Matches the Performance of Expert-developed QSAR/QSPR Models. *arXiv e-prints:1706.06689*, June 2017.

[62] N. De Cao and T. Kipf. MolGAN: An implicit generative model for small molecular graphs. *arXiv e-prints:1805.11973*, May 2018.

[63] Martin Simonovsky and Nikos Komodakis. Graphvae: Towards generation of small graphs using variational autoencoders. *arXiv e-prints:1802.03480*, 2018.

[64] J. You, B. Liu, R. Ying, V. Pande, and J. Leskovec. Graph Convolutional Policy Network for Goal-Directed Molecular Graph Generation. *arXiv e-prints: - 1806.02473*, June 2018.

[65] Z. Zhou, S. Kearnes, L. Li, R. N. Zare, and P. Riley. Optimization of Molecules via Deep Reinforcement Learning. *arXiv e-prints:1810.08678*, October 2018.

[66] Z. Boukouvalas, D. C. Elton, P. W. Chung, and M. D. Fuge. Independent Vector Analysis for Data Fusion Prior to Molecular Property Prediction with Machine Learning. *arXiv e-prints:: 1811.00628*, November 2018.

[67] Joseph L. Durant, Burton A. Leland, Douglas R. Henry, and James G. Nourse. Reoptimization of mdl keys for use in drug discovery. *Journal of Chemical Information and Computer Sciences*, 42(6):1273–1280, 2002.

[68] Artur Kadurin, Alexander Aliper, Andrey Kazennov, Polina Mamoshina, Quentin Vanhaelen, Kuzma Khrabrov, and Alex Zhavoronkov. The cornucopia of meaningful leads: Applying deep adversarial autoencoders for new molecule development in oncology. *Oncotarget*, December 2016.

[69] Artur Kadurin, Sergey Nikolenko, Kuzma Khrabrov, Alex Aliper, and Alex Zhavoronkov. druGAN: An advanced generative adversarial autoencoder model for de novo generation of new molecules with desired molecular properties in silico. *Molecular Pharmaceutics*, 14(9):3098–3104, 2017.

[70] E. Jannik Bjerrum and R. Threlfall. Molecular Generation with Recurrent Neural Networks (RNNs). *arXiv e-prints:1705.04612*, May 2017.

[71] Anvita Gupta, Alex T. Müller, Berend J. H. Huisman, Jens A. Fuchs, Petra Schneider, and Gisbert Schneider. Generative recurrent networks for de novo drug design. *Molecular Informatics*, 37(1-2):1700111, nov 2017.

[72] Marcus Olivecrona, Thomas Blaschke, Ola Engkvist, and Hongming Chen. Molecular de-novo design through deep reinforcement learning. *Journal of Cheminformatics*, 9(1), sep 2017.

[73] Marwin H. S. Segler, Thierry Kogej, Christian Tyrchan, and Mark P. Waller. Generating focused molecule libraries for drug discovery with recurrent neural networks. *ACS Central Science*, 4(1):120–131, dec 2017.

[74] Xiufeng Yang, Jinzhe Zhang, Kazuki Yoshizoe, Kei Terayama, and Koji Tsuda. ChemTS: an efficient python library for de novo molecular generation. *Science and Technology of Advanced Materials*, 18(1):972–976, 2017.

[75] Mehdi Cherti, Balázs Kégl, and Akın Kazakçı. De novo drug design with deep generative models: an empirical study. In *International Conference on Learning Representations*, Toulon, France, 2017.

[76] Daniel Neil, Marwin Segler, Laura Guasch, Mohamed Ahmed, Dean Plumbley, Matthew Sellwood, and Nathan Brown. Exploring deep recurrent models with reinforcement learning for molecule design. In *International Conference on Learning Representations*, 2018.

[77] Mariya Popova, Olexandr Isayev, and Alexander Tropsha. Deep reinforcement learning for de novo drug design. *Science Advances*, 4(7):eaap7885, jul 2018.

[78] Masato Sumita, Xiufeng Yang, Shinsuke Ishihara, Ryo Tamura, and Koji Tsuda. Hunting for organic molecules with artificial intelligence: Molecules optimized for desired excitation energies. *ACS Central Science*, 4(9):1126–1133, 2018.

[79] Daniel Merk, Lukas Friedrich, Francesca Grisoni, and Gisbert Schneider. De novo design of bioactive small molecules by artificial intelligence. *Molecular Informatics*, 37(1-2):1700153, jan 2018.

[80] Daniel Merk, Francesca Grisoni, Lukas Friedrich, and Gisbert Schneider. Tuning artificial intelligence on the de novo design of natural-product-inspired retinoid X receptor modulators. *Communications Chemistry*, 1(1), oct 2018.

[81] P. Ertl, R. Lewis, E. Martin, and V. Polyakov. In silico generation of novel, drug-like chemical matter using the LSTM neural network. *arXiv e-prints:1712.07449*, 2017.

[82] Josep Arús-Pous, Thomas Blaschke, Silas Ulander, Jean-Louis Reymond, Hongming Chen, and Ola Engkvist. Exploring the GDB-13 Chemical Space Using Deep Generative Models. *ChemRxiv preprint*, 10 2018.

[83] Shuangjia Zheng, Xin Yan, Qiong Gu, Yuedong Yang, Yunfei Du, Yutong Lu, and Jun Xu. QBMG: quasi-biogenic molecule generator with deep recurrent neural network. *Journal of Cheminformatics*, 11(1):5, Jan 2019.

[84] Peter Pogány, Navot Arad, Sam Genway, and Stephen D. Pickett. De novo molecule design by translating from reduced graphs to SMILES. *Journal of Chemical Information and Modeling*, dec 2018.

[85] Thomas Blaschke, Marcus Olivecrona, Ola Engkvist, Jürgen Bajorath, and Hongming Chen. Application of generative autoencoder in de novo molecular design. *Molecular Informatics*, 37(1-2):1700123, dec 2017.

[86] Jaechang Lim, Seongok Ryu, Jin Woo Kim, and Woo Youn Kim. Molecular generative model based on conditional variational autoencoder for de novo molecular design. *Journal of Cheminformatics*, 10(1), jul 2018.

[87] Seokho Kang and Kyunghyun Cho. Conditional molecular design with deep generative models. *Journal of Chemical Information and Modeling*, jul 2018.

[88] Shahar Harel and Kira Radinsky. Prototype-based compound discovery using deep generative models. *Molecular Pharmaceutics*, 15(10):4406–4416, jul 2018.

[89] M. J. Kusner, B. Paige, and J. M. Hernández-Lobato. Grammar Variational Autoencoder. *arXiv e-prints:1703.01925*, March 2017.

[90] Peter B. Jørgensen, Mikkel N. Schmidt, and Ole Winther. Deep generative models for molecular science. *Molecular Informatics*, 37(1-2):1700133, jan 2018.

[91] Wengong Jin, Regina Barzilay, and Tommi S. Jaakkola. Junction tree variational autoencoder for molecular graph generation. *arXiv e-prints: 1802.04364*, 2018.

[92] Qi Liu, Miltiadis Allamanis, Marc Brockschmidt, and Alexander L. Gaunt. Constrained graph variational autoencoders for molecule design. *arXiv e-prints:1805.09076*, 2018.

[93] H. Kajino. Molecular Hypergraph Grammar with its Application to Molecular Optimization. *arXiv e-prints:1803.03324*, September 2018.

[94] Bidisha Samanta, Abir De, Niloy Ganguly, and Manuel Gomez-Rodriguez. Designing random graph models using variational autoencoders with applications to chemical design. *arXiv e-prints:1802.05283*, 2018.

[95] G. Lima Guimaraes, B. Sanchez-Lengeling, C. Outeiral, P. L. Cunha Farias, and A. Aspuru-Guzik. Objective-Reinforced Generative Adversarial Networks (ORGAN) for Sequence Generation Models. *arXiv e-prints:1705.10843*, May 2017.

[96] Evgeny Putin, Arip Asadulaev, Yan Ivanenkov, Vladimir Aladinskiy, Benjamin Sanchez-Lengeling, Alán Aspuru-Guzik, and Alex Zhavoronkov. Reinforced adversarial neural computer for de novo molecular design. *Journal of Chemical Information and Modeling*, 58(6):1194–1204, may 2018.

[97] Evgeny Putin, Arip Asadulaev, Quentin Vanhaelen, Yan Ivanenkov, Anastasia V. Aladinskaya, Alex Aliper, and Alex Zhavoronkov. Adversarial threshold neural computer for molecular de novo design. *Molecular Pharmaceutics*, 2018.

[98] Oscar Méndez-Lucio, Benoit Baillif, Djork-Arné Clevert, David Rouquié, and Joerg Wichard. De novo generation of hit-like molecules from gene expression signatures using artificial intelligence. *ChemRxiv preprint*, nov 2018.

[99] Łukasz Maziarka, Agnieszka Pocha, Jan Kaczmarczyk, Krzysztof Rataj, and Micha Warcho. Mol-cyclegan - a generative model for molecular optimization, 2019.

[100] Benjamin Sanchez-Lengeling, Carlos Outeiral, Gabriel L. Guimaraes, and Alan Aspuru-Guzik. Optimizing distributions over molecular space. An Objective-Reinforced Generative Adversarial Network for Inverse-design Chemistry (ORGANIC). *ChemRxiv preprint*, 8 2017.

[101] Hisaki Ikebata, Kenta Hongo, Tetsu Isomura, Ryo Maezono, and Ryo Yoshida. Bayesian molecular design with a chemical language model. *Journal of Computer-Aided Molecular Design*, 31(4):379–391, mar 2017.

[102] Daniil Polykovskiy, Alexander Zhebrak, Dmitry Vetrov, Yan Ivanenkov, Vladimir Aladinskiy, Polina Mamoshina, Marine Bozdaganyan, Alexander Aliper, Alex Zhavoronkov, and Artur Kadurin. Entangled conditional adversarial autoencoder for de novo drug discovery. *Molecular Pharmaceutics*, sep 2018.

[103] Martin Simonovsky and Nikos Komodakis. Dynamic edge-conditioned filters in convolutional neural networks on graphs. In *2017 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017, Honolulu, HI, USA, July 21-26, 2017*, pages 29–38, 2017.

[104] Lorenz C. Blum and Jean-Louis Reymond. 970 million druglike small molecules for virtual screening in the chemical universe database GDB-13. *Journal of the American Chemical Society*, 131(25):8732–8733, 2009.

[105] Teague Sterling and John J. Irwin. ZINC 15 – ligand discovery for everyone. *Journal of Chemical Information and Modeling*,

55(11):2324–2337, nov 2015.

[106] Lars Ruddigkeit, Ruud van Deursen, Lorenz C. Blum, and Jean-Louis Reymond. Enumeration of 166 billion organic small molecules in the chemical universe database GDB-17. *Journal of Chemical Information and Modeling*, 52(11):2864–2875, 2012.

[107] Raghunathan Ramakrishnan, Pavlo O Dral, Matthias Rupp, and O Anatole von Lilienfeld. Quantum chemistry structures and properties of 134 kilo molecules. *Scientific Data*, 1, 2014.

[108] Steven A. Lopez, Edward O. Pyzer-Knapp, Gregor N. Simm, Trevor Lutzow, Kewei Li, Laszlo R. Seress, Johannes Hachmann, and Alán Aspuru-Guzik. The harvard organic photovoltaic dataset. *Scientific Data*, 3:160086, sep 2016.

[109] Aurélien Géron. *Hands-On Machine Learning with Scikit-Learn and TensorFlow: Concepts, Tools, and Techniques to Build Intelligent Systems*. O'Reilly Media, Inc., 1st edition, 2017.

[110] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT Press, 2016. `http://www.deeplearningbook.org`.

[111] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural Computation*, 9(8):1735–1780, nov 1997.

[112] Kyunghyun Cho, Bart van Merrienboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. Learning phrase representations using rnn encoder–decoder for statistical machine translation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1724–1734. Association for Computational Linguistics, 2014.

[113] Marc'Aurelio Ranzato, Sumit Chopra, Michael Auli, and Wojciech Zaremba. Sequence level training with recurrent neural networks. *arXiv*, abs/1511.06732, 2015.

[114] Arun Venkatraman, Martial Hebert, and J. Andrew Bagnell. Improving multi-step prediction of learned time series models. In *Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence*, AAAI'15, pages 3024–3030. AAAI Press, 2015.

[115] Samy Bengio, Oriol Vinyals, Navdeep Jaitly, and Noam Shazeer. Scheduled sampling for sequence prediction with recurrent neural networks. In *Proceedings of the 28th International Conference on Neural Information Processing Systems - Volume 1*, NIPS'15, pages 1171–1179, Cambridge, MA, USA, 2015. MIT Press.

[116] F. Huszár. How (not) to Train your Generative Model: Scheduled Sampling, Likelihood, Adversary? *arXiv e-prints:1511.05101*, November 2015.

[117] Ronald J. Williams and David Zipser. A learning algorithm for continually running fully recurrent neural networks. *Neural Computation*, 1(2):270–280, jun 1989.

[118] Rafa Gómez-Bombarelli. Broad institute models, inference, & algorithms talk: "deep learning chemical space", 2017.

[119] Ronald J. Williams. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine Learning*, 8(3-4):229–256, may 1992.

[120] Natasha Jaques, Shixiang Gu, Dzmitry Bahdanau, Jos Miguel Hernndez-Lobato, Richard E. Turner, and Douglas Eck. Sequence tutor: Conservative fine-tuning of sequence generation models with KL-control. In *International Conference on Machine Learning*, 2017.

[121] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A. Rusu, Joel Veness, Marc G. Bellemare, Alex Graves, Martin Riedmiller, Andreas K. Fidjeland, Georg Ostrovski, Stig Petersen, Charles Beattie, Amir Sadik, Ioannis Antonoglou, Helen King, Dharshan Kumaran, Daan Wierstra, Shane Legg, and Demis Hassabis. Human-level control through deep reinforcement learning. *Nature*, 518(7540):529–533, feb 2015.

[122] Geoffrey Hinton and Ruslan Salakhutdinov. Reducing the dimensionality of data with neural networks. *Science*, 313(5786):504 – 507, 2006.

[123] Ruslan Salakhutdinov and Geoffrey Hinton. Deep Boltzmann machines. In David van Dyk and Max Welling, editors, *Proceedings of the Twelth International Conference on Artificial Intelligence and Statistics*, volume 5 of *Proceedings of Machine Learning Research*, pages 448–455, Hilton Clearwater Beach Resort, Clearwater Beach, Florida USA, 16–18 Apr 2009. PMLR.

[124] P. Mehta, M. Bukov, C.-H. Wang, A. G. R. Day, C. Richardson, C. K. Fisher, and D. J. Schwab. A high-bias, low-variance introduction to Machine Learning for physicists. *arXiv e-prints:1803.08823*, March 2018.

[125] D. Janz, J. van der Westhuizen, and J. M. Hernández-Lobato. Actively Learning what makes a Discrete Sequence Valid. *arXiv e-prints:1708.04465*, August 2017.

[126] Alireza Makhzani, Jonathon Shlens, Navdeep Jaitly, and Ian Goodfellow. Adversarial autoencoders. In *International Conference on Learning Representations*, 2016.

[127] R.-R. Griffiths and J. M. Hernández-Lobato. Constrained Bayesian Optimization for Automatic Chemical Design. *arXiv e-prints:1709.05501*, September 2017.

[128] Ilya Sutskever, Oriol Vinyals, and Quoc V. Le. Sequence to Sequence Learning with Neural Networks. *arXiv e-prints: 1409.3215*, September 2014.

[129] Mario Lucic, Karol Kurach, Marcin Michalski, Sylvain Gelly, and Olivier Bousquet. Are GANs Created Equal? A Large-Scale Study. *arXiv e-prints:1711.10337*, November 2017.

[130] M. Arjovsky, S. Chintala, and L. Bottou. Wasserstein GAN. *arXiv e-prints: 1701.07875*, January 2017.

[131] Karol Kurach, Mario Lucic, Xiaohua Zhai, Marcin Michalski, and Sylvain Gelly. The GAN Landscape: Losses, Architectures, Regularization, and Normalization. *arXiv e-prints:1807.04720*, July 2018.

[132] Lantao Yu, Weinan Zhang, Jun Wang, and Yong Yu. Seqgan: Sequence generative adversarial nets with policy gradient. In *Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence, February 4-9, 2017, San Francisco, California, USA.*, pages 2852–2858, 2017.

[133] Alex Graves, Greg Wayne, Malcolm Reynolds, Tim Harley, Ivo Danihelka, Agnieszka Grabska-Barwińska, Sergio Gómez Colmenarejo, Edward Grefenstette, Tiago Ramalho, John Agapiou, Adrià Puigdomènech Badia, Karl Moritz Hermann, Yori Zwols, Georg Ostrovski, Adam Cain, Helen King, Christopher Summerfield, Phil Blunsom, Koray Kavukcuoglu, and Demis Hassabis. Hybrid computing using a neural network with dynamic external memory. *Nature*, 538(7626):471–476, oct 2016.

[134] A. Graves, G. Wayne, and I. Danihelka. Neural Turing Machines. *arXiv e-prints: 1807.06156*, October 2014.

[135] D. Sculley, Jasper Snoek, Alex Wiltschko, and Ali Rahimi. Winner's curse? On pace, progress, and empirical rigor. In *Sixth International Conference on Learning Representations - Workshop Track*, 2018.

[136] Peter Henderson, Riashat Islam, Philip Bachman, Joelle Pineau, Doina Precup, and David Meger. Deep reinforcement learning that matters. In *Thirthy-Second AAAI Conference On Artificial Intelligence*, 2018.

[137] Gábor Melis, Chris Dyer, and Phil Blunsom. On the State of the Art of Evaluation in Neural Language Models. *arXiv e-prints:1707.05589*, July 2017.

[138] Tim Salimans, Ian J. Goodfellow, Wojciech Zaremba, Vicki Cheung, Alec Radford, and Xi Chen. Improved techniques for training gans. *arXiv e-prints:1606.03498*, 2016.

[139] L. Theis, A. van den Oord, and M. Bethge. A note on the evaluation of generative models. Apr 2016.

[140] Martin Heusel, Hubert Ramsauer, Thomas Unterthiner, Bernhard Nessler, and Sepp Hochreiter. Gans trained by a two time-scale update rule converge to a local nash equilibrium, 2017.

[141] Kristina Preuer, Philipp Renz, Thomas Unterthiner, Sepp Hochreiter, and Günter Klambauer. Fréchet ChemNet distance: A metric for generative models for molecules in drug discovery. *Journal of Chemical Information and Modeling*, 58(9):1736–1741, aug 2018.

[142] Zhenqin Wu, Bharath Ramsundar, Evan N. Feinberg, Joseph Gomes, Caleb Geniesse, Aneesh S. Pappu, Karl Leswing, and Vijay Pande. MoleculeNet: a benchmark for molecular machine

learning. *Chemical Science*, 9(2):513–530, 2018.

[143]Daniil Polykovskiy, Alexander Zhebrak, Benjamin Sanchez-Lengeling, Sergey Golovanov, Oktai Tatanov, Stanislav Belyaev, Rauf Kurbanov, Aleksey Artamonov, Vladimir Aladinskiy, Mark Veselov, Artur Kadurin, Sergey Nikolenko, Alan Aspuru-Guzik, and Alex Zhavoronkov. Molecular Sets (MOSES): A Benchmarking Platform for Molecular Generation Models. *arXiv e-prints:: arXiv:1811.12823*, November 2018.

[144]Mostapha Benhenda, Esben Jannik Bjerrum, Hsiao Yi, and Chintan Zaveri. DiversityNet: a collaborative benchmark for generative ai models in chemistry. *Authorea preprint*, 2018.

[145]Nathan Brown, Marco Fiscato, Marwin H. S. Segler, and Alain C. Vaucher. GuacaMol: Benchmarking Models for De Novo Molecular Design. *arXiv e-prints:1811.09621*, November 2018.

[146]Daniel Jiwoong Im, Alllan He Ma, Graham W. Taylor, and Kristin Branson. Quantitatively evaluating GANs with divergences proposed for training. In *International Conference on Learning Representations*, 2018.

[147]Ishaan Gulrajani, Colin Raffel, and Luke Metz. Towards GAN benchmarks which require generalization. In *International Conference on Learning Representations*, 2019.

[148]Derek Lowe. Calculating a few too many new compounds, 2016. `http://blogs.sciencemag.org/pipeline/archives/2016/11/08/calculating-a-few-too-many-new-compounds`.

[149]M. Benhenda. ChemGAN challenge for drug discovery: can AI reproduce natural chemical diversity? *arXiv e-prints:1703.01925*, August 2017.

[150]N. Yoshikawa, K. Terayama, T. Honma, K. Oono, and K. Tsuda. Population-based de novo molecule generation, using grammatical evolution. *arXiv e-prints:1804.02134*, April 2018.

[151]Jane Panteleev, Hua Gao, and Lei Jia. Recent applications of machine learning in medicinal chemistry. *Bioorganic & Medicinal Chemistry Letters*, 28(17):2807–2815, sep 2018.

[152]J. B. Tenenbaum. A global geometric framework for nonlinear dimensionality reduction. *Science*, 290(5500):2319–2323, dec 2000.

[153]Laurens van der Maaten and Geoffrey Hinton. Visualizing data using t-SNE. *Journal of Machine Learning Research*, 9:2579–2605, 2008.

[154]Alec Radford, Luke Metz, and Soumith Chintala. Unsupervised representation learning with deep convolutional generative adversarial networks. In *International Conference on Learning Representations*, 2016.

[155]Pedro Domingos. A few useful things to know about machine learning. *Commun. ACM*, 55(10):78–87, October 2012.

[156]Tom White. Sampling generative networks: Notes on a few effective techniques. *arXiv e-prints:1609.04468*, 2016.

[157]Peter Ertl and Ansgar Schuffenhauer. Estimation of synthetic accessibility score of drug-like molecules based on molecular complexity and fragment contributions. *Journal of Cheminformatics*, 1(1):8, Jun 2009.

[158]Yevgeniy Podolyan, Michael A. Walters, and George Karypis. Assessing synthetic accessibility of chemical compounds using machine learning methods. *Journal of Chemical Information and Modeling*, 50(6):979–991, jun 2010.

[159]Yoshifumi Fukunishi, Takashi Kurosawa, Yoshiaki Mikami, and Haruki Nakamura. Prediction of synthetic accessibility based on commercially available compound databases. *Journal of Chemical Information and Modeling*, 54(12):3259–3267, dec 2014.

[160]Peter Ertl, Silvio Roggo, and Ansgar Schuffenhauer. Natural product-likeness score and its application for prioritization of compound libraries. *Journal of Chemical Information and Modeling*, 48(1):68–74, jan 2008.

[161]G. Richard Bickerton, Gaia V. Paolini, Jérémy Besnard, Sorel Muresan, and Andrew L. Hopkins. Quantifying the chemical beauty of drugs. *Nature Chemistry*, 4(2):90–98, feb 2012.

[162]Ingo Muegge, Sarah L. Heald, and David Brittelli. Simple selection criteria for drug-like chemical matter. *Journal of Medicinal Chemistry*, 44(12):1841–1846, jun 2001.

[163]Ingo Muegge. Selection criteria for drug-like compounds. *Medicinal Research Reviews*, 23(3):302–321, mar 2003.

[164]Amit Kalgutkar, Iain Gardner, R. Obach, Christopher Shaffer, Ernesto Callegari, Kirk Henne, Abdul Mutlib, Deepak Dalvie, Jae Lee, Yasuhiro Nakai, John O'Donnell, Jason Boer, and Shawn Harriman. A comprehensive listing of bioactivation pathways of organic functional groups. *Current Drug Metabolism*, 6(3):161–225, jun 2005.

[165]Antoine Daina, Olivier Michielin, and Vincent Zoete. SwissADME: a free web tool to evaluate pharmacokinetics, drug-likeness and medicinal chemistry friendliness of small molecules. *Scientific Reports*, 7(1), mar 2017.

[166]Tong Che, Yanran Li, Ruixiang Zhang, R Devon Hjelm, Wenjie Li, Yangqiu Song, and Yoshua Bengio. Maximum-likelihood augmented discrete generative adversarial networks, 2017.

[167]Chris Olah and Shan Carter. Attention and augmented recurrent neural networks. *Distill*, 2016.

[168]Zois Boukouvalas. Development of ICA and IVA Algorithms with Application to Medical Image Analysis. *arXiv e-prints: 1801.08600*, January 2018.

[169]A. Nouira, N. Sokolovska, and J.-C. Crivello. CrystalGAN: Learning to Discover Crystallographic Structures with Generative Adversarial Networks. *arXiv e-prints: 1810.11203*, October 2018.

[170]Xiaolin Li, Zijiang Yang, L. Catherine Brinson, Alok Choudhary, Ankit Agrawal, and Wei Chen. A deep adversarial learning methodology for designing microstructural material systems. In *Volume 2B: 44th Design Automation Conference*. ASME, aug 2018.

[171]Rahul Singh, Viraj Shah, Balaji Pokuri, Soumik Sarkar, Baskar Ganapathysubramanian, and Chinmay Hegde. Physics-aware Deep Generative Models for Creating Synthetic Microstructures. *arXiv e-prints:1811.09669*, November 2018.

[172]Z. Yang, X. Li, L. C. Brinson, A. N. Choudhary, W. Chen, and A. Agrawal. Microstructural Materials Design via Deep Adversarial Learning Methodology. *arXiv e-prints:1805.02791*, May 2018.

[173]Ashwin Jeyaseelan Chen, Wei and Mark D. Fuge. Synthesizing designs with inter-part dependencies using hierarchical generative adversarial networks. In *ASME 2016 International Design Engineering Technical Conferences and Computers and Information in Engineering Conference*, Quebec City, Canada, Aug 2018. ASME.

[174]Johannes Hachmann, Mohammad Atif Faiz Afzal, Mojtaba Haghighatlari, and Yudhajit Pal. Building and deploying a cyberinfrastructure for the data-driven design of chemical systems and the exploration of chemical space. *Molecular Simulation*, 44(11):921–929, may 2018.

[175]Semion K. Saikin, Christoph Kreisbeck, Dennis Sheberla, Jill S. Becker, and Aspuru-Guzik A. Closed-loop discovery platform integration is needed for artificial intelligence to make an impact in drug discovery. *Expert Opinion on Drug Discovery*, 14(1):1–4, nov 2018.

[176]Gisbert Schneider. Automating drug discovery. *Nature Reviews Drug Discovery*, 17(2):97–113, dec 2017.

[177]Daniel P. Tabor, Loc M. Roch, Semion K. Saikin, Christoph Kreisbeck, Dennis Sheberla, Joseph H. Montoya, Shyam Dwaraknath, Muratahan Aykol, Carlos Ortiz, Hermann Tribukait, Carlos Amador-Bedolla, Christoph J. Brabec, Benji Maruyama, Kristin A. Persson, and Alán Aspuru-Guzik. Accelerating the discovery of materials for clean energy in the era of smart automation. *Nature Reviews Materials*, 3(5):5–20, apr 2018.