MSDE

REVIEW

Check for updates

Cite this: *Mol. Syst. Des. Eng.*, 2019, **4**, 828

Deep learning for molecular design—a review of the state of the art

Daniel C. Elton, †*[a] Zois Boukouvalas,[ab] Mark D. Fuge[a] and Peter W. Chung[a]

Received 18th March 2019, Accepted 22nd May 2019

DOI: 10.1039/c9me00039a

rsc.li/molecular-engineering

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.

Design, System, Application

In this review we discuss the use of deep learning techniques to perform molecular generation and optimization, a new field which has proved to be a fertile area for development in the past three years. The broad categories of techniques which we discuss are recursive neural networks, variational autoencoders, generative adversarial networks, and reinforcement learning. These techniques can be used for generating a diverse set of leads for high throughput screening. The latent spaces of generative models can be used for optimization of molecules when coupled with a property predicting module. Alternatively, a pure reinforcement learning approach to optimization can be taken. The majority of work so far has been focused on drug design but there are numerous other application areas now appearing in the literature such as metal organic frameworks, organic LEDs, organic solar cells, and energetic materials. While currently genetic algorithms can often compete with deep learning based methods for molecular optimization, the field is rapidly developing and there are many avenues open for improvement. The current literature points to a future where deep generative modeling techniques will find utility not just for generating and optimizing molecules but also for materials and nanoscale systems.

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

[a] *Department of Mechanical Engineering, University of Maryland, College Park, Maryland, 20740, USA. E-mail: daniel.elton@nih.gov*
[b] *Department of Mathematics and Statistics, American University, Washington, D.C., 20016, USA*
† Present address: National Institutes of Health Clinical Center, Bethesda, Maryland, USA.

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

Daniel C. Elton

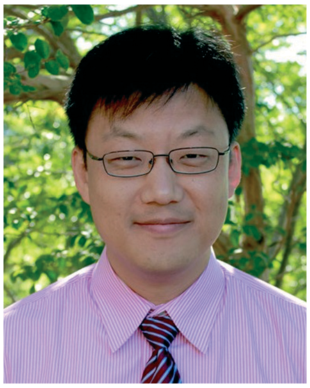
Dr. Daniel C. Elton received a B.S. degree in physics from Rensselaer Polytechnic Institute in 2009 and a Ph.D. in physics from Stony Brook University in 2016. He worked as a Postdoctoral Research Associate and later an Assistant Research Scientist at the University of Maryland, College Park between 2017–2019 where he focused on applications of deep learning to molecular design and discovery. In January 2019 he moved to work as a contractor Staff Scientist at the National Institutes of Health. His current work focuses on applications of deep learning and AI to detection and segmentation in medical images.

Zois Boukouvalas

Dr. Zois Boukouvalas received his B.S. degree in Mathematics from the University of Patras, Greece, an M.S. degree in Applied and Computational Mathematics from the Rochester Institute of Technology, and a Ph.D. degree in Applied Mathematics from University of Maryland Baltimore County in 2017. Since 2017 he has worked as a Postdoctoral Research Associate at the University of Maryland, College Park, and in August 2019 he will start as an Assistant Professor in the Department of Mathematics and Statistics at American University. His research interests include blind source separation and machine learning for big data problems.

Mark D. Fuge

Dr. Mark D. Fuge is an Assistant Professor of Mechanical Engineering at the University of Maryland, College Park. His staff and students study fundamental scientific and mathematical questions behind how humans and computers can work together to design better complex engineered systems, from the molecular scale to systems as large as aircraft and ships, by using tools from Applied Mathematics and Computer Science. He received his Ph.D. from UC Berkeley and has received a DARPA Young Faculty Award, a National Defense Science and Engineering Graduate Fellowship, and has prior/current support from NSF, NIH, DARPA, ONR, and Lockheed Martin.


Peter W. Chung

Dr. Peter W. Chung is an Associate Professor in the Department of Mechanical Engineering at the University of Maryland in College Park. He serves as the Division Lead of the Mechanics, Materials, and Manufacturing Division within the department and is also the Lead of the Energetics Group in the Center for Engineering Concepts Development.

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".[18,19] Another important technical advance was the efficient implementation of neural network training on graphics processing units (GPUs). By 2015 better 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 (Kingma *et al.*, 2013 (ref. 21)) and generative adversarial networks (Goodfellow *et al.* 2014 (ref. 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 2. 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]

Our review touches on four major issues we have observed in the field. The first is the importance and opportunities for improvement by using different molecular representations. Recent efforts have begun to depart from the use of simplified molecular-input line-entry system (SMILES) strings towards representations that are "closer to the chemical structure" and offer improved chemical accuracy, such as graph grammar based methods. The second issue is architecture selection. We discuss the pros and cons underlying different choices of model architecture and present some of their key mathematical details to better illuminate how different approaches relate to each other. This leads us to highlight the advantages of adversarial training and reinforcement learning over maximum likelihood based training. We also touch on techniques for molecular optimization using generative models, which has grown in popularity recently. The third major issue is the approaches for quantitatively evaluating different approaches for molecular generation and optimization. Fourth, and finally, we discuss is reward function design, which is crucial for the practical application of methods which use reinforcement learning. We contribute by offering novel overview of how to engineer reward function to generate a set of leads which is chemically stable, diverse, novel, has good properties, and is synthesizable.

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.

1 Molecular representation

The molecular representation refers to the digital encoding used for each molecule that serves as input for training the deep learning model. A representation scheme must capture essential structural information about each molecule. Creating an appropriate representation from a molecular structure is called featurization. Two important properties that are desirable (but not required) for representations are uniqueness

Table 1 Different representation schemes

	Method	Unique?	Invertible?
3D	Raw voxels	✗	✓
	Smoothed voxels	✗	✓
	Tensor field networks	✗	✗
2D graph	SMILES	✗	✓
	Canonical SMILES	✓	✓
	InChI	✓	✓
	MACCS keys	✓	✗
	Tensors	✗	✓
	Chemception images	✓	✓
	Fingerprinting	✓	✗

Table 2 For works that trained models separately on multiple datasets we report only the largest dataset used. Several of these datasets are described in Table 3, 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

Architecture	Representation	N_{train}	Dataset(s)	Citation(s)
RNN	SMILES	1 611 889	ZINC	Bjerrum, 2017 (ref. 72)
RNN	SMILES	541 555	ChEMBL[a]	Gupta, 2017 (ref. 73)
RNN	SMILES	350 419	DRD2	Olivecrona, 2017 (ref. 74)
RNN	SMILES	1 400 000	ChEMBL	Segler, 2017 (ref. 75)
RNN	SMILES	250 000	ZINC	Yang, 2017 (ref. 76)
RNN	SMILES	200 000	ZINC	Cherti, 2017 (ref. 77)
RNN	SMILES	1 735 442	ChEMBL	Neil, 2018 (ref. 78)
RNN	SMILES	1 500 000	ChEMBL	Popova, 2018 (ref. 79)
RNN	SMILES	13 000	PubChemQC	Sumita, 2018 (ref. 80)
RNN	SMILES	541 555	ChEMBL[a]	Merk, 2018 (ref. 81)
RNN	SMILES	541 555	ChEMBL[a]	Merk, 2018 (ref. 82)
RNN	SMILES	509 000	ChEMBL	Ertl, 2018 (ref. 83)
RNN	SMILES	1 000 000	GDB-13	Arús-Pous, 2018 (ref. 84)
RNN	SMILES	163 000	ZINC	Zheng, 2019 (ref. 85)
RNN	RG + SMILES	798 243	ChEMBL	Pogány, 2018 (ref. 86)
RNN	Graph operations	130 830	ChEMBL	Li, 2018 (ref. 59)
VAE	SMILES	249 000	ZINC/QM9	Gómez-Bombarelli, 2016 (ref. 23)
VAE	SMILES	1 200 000	ChEMBL	Blaschke, 2018 (ref. 87)
VAE	SMILES	500 000	ZINC	Lim, 2018 (ref. 88)
VAE	SMILES	300 000	ZINC	Kang, 2018 (ref. 89)
VAE	SMILES	190 000	ZINC	Harel, 2018 (ref. 90)
VAE	SMILES	1 211 352	ChEMBL23	Sattarov, 2019 (ref. 91)
GVAE	CFG (SMILES)	200 000	ZINC	Kusner, 2017 (ref. 92)
GVAE	CFG (custom)	3989	PSC	Jørgensen, 2018 (ref. 26 and 93)
SD-VAE	CFG (custom)	250 000	ZINC	Dai, 2018 (ref. 58)
JT-VAE	Graph operations	250 000	ZINC	Jin, 2018 (ref. 94)
JT-VAE	Graph operations	250 000	ZINC	Jin, 2019 (ref. 95)
CVAE	Graph	250 000	ZINC/CEPDB	Liu, 2018 (ref. 96)
MHG-VAE	Graph (MHG)	220 011	ZINC	Kajino, 2018 (ref. 97)
VAE	Graph	72 000 000	ZINC + PubChem	Winter, 2018 (ref. 98)
VAE	Graph	10 000	ZINC/QM9	Samanta, 2018 (ref. 99)
VAE	Graph (tensors)	10 000	ZINC	Samanta, 2018 (ref. 100)
VAE	Graph (tensors)	250 000	ZINC/QM9	Simonovsky, 2018 (ref. 64)
CVAE	Graph (tensors)	250 000	ZINC	Ma, 2018 (ref. 101)
VAE	3D wave transform	48 000 000	ZINC	Kuzminykh, 2018 (ref. 40)
CVAE	3D density	192 813 983	ZINC	Skalic, 2019 (ref. 41)
VAE + RL	MPNN + graph ops	133 885	QM9	Kearnes, 2019 (ref. 102)
GAN	SMILES	5000	GBD-17	Guimaraes, 2017 (ref. 103)
GAN (ANC)	SMILES	15 000	ZINC/CHEMDIV	Putin, 2018 (ref. 104)
GAN (ATNC)	SMILES	15 000	ZINC/CHEMDIV	Putin, 2018 (ref. 105)
GAN	Graph (tensors)	133 885	QM9	De Cao, 2018 (ref. 50 and 63)
GAN	MACCS (166 bit)	6252	MCF-7	Kadurin, 2017 (ref. 70)
sGAN	MACCS (166 bit)	20 000	L1000	Méndez-Lucio, 2017 (ref. 106)
CycleGAN	Graph operations	250 000	ZINC	Maziarka, 2019 (ref. 107)
AAE	MACCS (166 bit)	6252	MCF-7	Kadurin, 2017 (ref. 69)
AAE	SMILES	15 000	HCEP	Sanchez-Lengeling, 2017 (ref. 108)
CCM-AAE	Graph (tensors)	133 885	QM9	Grattarola, 2018 (ref. 109)
BMI	SMILES	16 674	PubChem	Ikebata, 2017 (ref. 110)
CAAE	SMILES	1 800 000	ZINC	Polykovskiy, 2018 (ref. 111)
GCPN	Graph	250 000	ZINC	You, 2018 (ref. 65)
Pure RL	Graph	n/a	n/a	Zhou, 2018 (ref. 66)
Pure RL	Fragments	n/a	n/a	Ståhl, 2019 (ref. 112)

Acronyms used are: AAE = adversarial autoencoder, ANC = adversarial neural computer, ATNC = adversarial threshold neural computer, BMI = Bayesian model inversion, CAAE = constrained AAE, CCM-AAE = constant-curvature Riemannian manifold AAE, CFG = context free grammar, CVAE = constrained VAE, ECC = edge-conditioned graph convolutions,[71] 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.[a] Filtered to isolate likely bioactive compounds.

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

Table 3 Some publicly available datasets

Dataset	Description	N	URL/citation
GDB-13	Combinatorially generated library	977 468 314	http://gdb.unibe.ch/downloads/[113]
ZINC15	Commercially available compounds	>750 000 000	http://zinc15.docking.org[114]
GDB-17	Combinatorially generated library	50 000 000	http://gdb.unibe.ch/downloads/[115]
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/
PubChemQC	Compounds from PubChem with property data from quantum chemistry (DFT) calculations	3 981 230	http://pubchemqc.riken.jp/[116]
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/[117]
BNPAH	B, *N*-substituted polycyclic aromatic hydrocarbons with properties calculated from *ab initio* density functional theory	33 000	https://moldis.tifrh.res.in/datasets.html[118]
DrugBank	FDA drugs and other drugs available internationally	10 500	https://www.drugbank.ca/
Energetics	Energetic molecules and simulation data collected from public domain literature	417	https://git.io/energeticmols[27]
HOPV15	Harvard organic photovoltaic dataset	350[a]	https://figshare.com/articles/HOPV15_Dataset/1610063/4 (ref. 119)

[a] Also contains numerous conformers for each molecule, for a total of 4855 structures.

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. Equilibrium coordinates can be determined from the 2D graph *via* energy minimization. This point may explain the success of machine learning based property prediction at predicting high level properties from 2D graphs, as opposed to 3D structures.[8,27,39] In so much as properties are related to equilibrium structure, machine learning infers this from the 2D graph. In this section, we review both 3D and 2D representation schemes that have been developed recently (Table 1).

1.1 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 3Dcoordinates 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. This issue can be mitigated *via* spatial smoothing (blurring) by placing spherical Gaussians or a set of decaying concentric waves around each atomic nuclei.[40] Alternatively, the van der Waals radius may be used.[41] Amidi *et al.* use this type of approach for predictive modeling with 3D convolutional neural networks (CNNs),[42] while Kuzminykh *et al.* and Skalic *et al.* use this approach with a CNN-based autoencoder for generative modeling.[40,41]

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.[40,42] Approaches which generate feature vectors from 3D coordinates that are invariant to translation and rotation are wavelet transform invariants,[43] solid harmonic wavelet scattering transforms,[44] and tensor field networks.[45] 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 (deep learning models learn to generate these representations, but if they cannot be unambiguously related to a 3D structure they are not very useful). Despite their issues with invertibility, tensor field networks have been suggested to have utility for generative modeling since it was shown they can accurately predict the location of missing atoms in molecules where one atom was removed.[45] 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.

1.2 Representations of molecular graphs

1.2.1 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 $\mathscr{G}$ with a set of edges $\mathscr{E}$ and set of vertices $\mathscr{V}$. 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[46,47]). After coordinate embedding, the most energetically favorable conformation of the molecule can be obtained by doing energy minimization with classical force fields or quantum mechanical simulation.

There are several ways to represent graphs for machine learning. The most popular way is the SMILES string representation.[48] 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.[49,50]

Many deep generative modeling techniques have been developed specifically for sequence generation, most notably recurrent neural networks (RNNs), which can be used for SMILES generation. 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 can have at least N equivalent SMILES string representations. 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*[51] or *RDKit*[52] 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.[53] 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.[53,54] 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.[53]

In addition to SMILES strings, Gómez-Bombarelli *et al.* have tried InChI strings[55] 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.[23] Interestingly, Winter *et al.* show that more physically meaningful latent spaces can be obtained by training a variational autoencoder to translate between InChI to SMILES.[56] 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).[57] The rules of the context-free grammar impose constraints based on the grammar of SMILES strings.[58] 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.[58] 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.[59] Another more recent work by Li *et al.* uses a deep neural network to decide on graph generation steps (append, connect, or terminate).[60] 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,[61] and likely will have utility for deep generative models as well.

1.2.2 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.[62] The Chemception framework takes a SMILES string in and produces an 80 × 80 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.[62] 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.[53]

1.2.3 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,[50,63] 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.*[64] use a similar approach—they take 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 isomorphism" problem, which takes N^4 operations in the worse case). Simonovsky *et al.* use an approximate graph matching algorithm to do this, but it is still computationally expensive.

1.2.4 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[65] and in recent work using pure deep reinforcement learning to generate molecules.[66] 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,[27,67] 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[68] for molecular representation with adversarial autoencoders.[69,70] 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.

2 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. High level diagrams of the major architectures are shown in Fig. 1. A basic familiarity with machine learning concepts is assumed.

2.1 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 $S_{1:T} = (S_1,\ldots, S_T)$, $S_\mathrm{T} \in \mathcal{V}$, 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, 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.[120,121] 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,[122] or a newer more computationally efficient variant called the

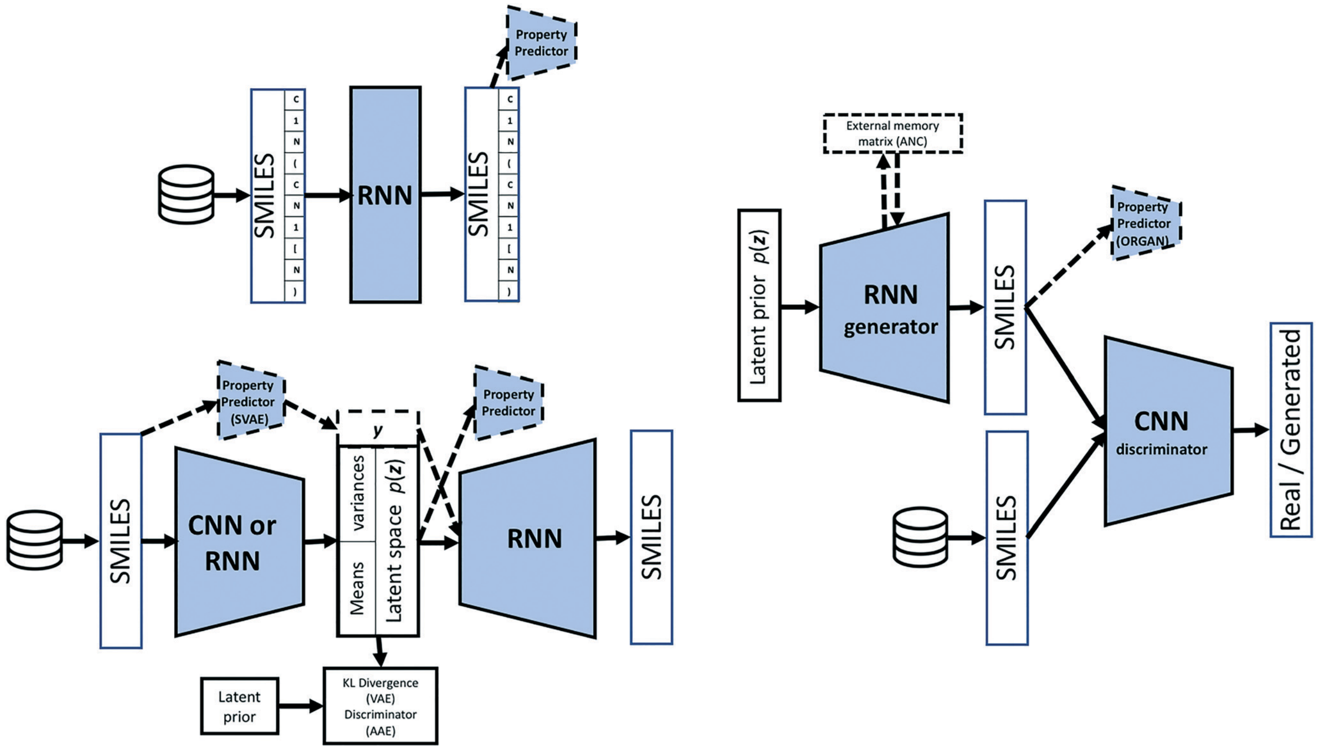


Fig. 1 Bird's eye views of three popular frameworks for generative modeling using SMILES strings, with possible variations shown with dashed lines.

gated recurrent unit (GRU).[123] 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.[122,123]

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".[124] 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.[125] DaD was extended to sequence generation with RNNs by Bengio *et al.*, who called the method "scheduled sampling".[126] 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.[127]

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 π_θ parametrized by θ is trained with the following differentiable loss:

$$L^{\text{MLE}} = -\sum_{s \in Z} \sum_{t=2}^{T} \log \pi_\theta \left(s_T | S_{1:T-1} \right) \quad (1)$$

Here 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[128] 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%.[129] 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.[121]

In the context of SMILES strings generation, to generate each character the output layer usually gives probabilities p_i for every possible SMILES string token. When running in generative mode, the typical method is to use a multinomial sampler to sample this distribution, while in training mode one typically just chooses the token with the highest probability. Using a multinomial sampler captures the model's true distribution, but because MLE training tends to focus on optimizing the peaks of the distribution and doesn't always capture the tails of distributions well. 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.

2.1.1 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*.[78] 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 of the "environment" is the sequence generated so far is $(s_0,\ldots, s_t)$ 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:

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

where θ 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 non-zero 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,[130] 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.[78]

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.[74] Additionally, most policy learning methods have been proven to lead to an optimal policy and the resulting generative models are faster to sample.[74] 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.[66]

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:[74,131]

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

Other possibilities are explored by Olivecrona *et al.*[74] 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.* sought approaches that circumvent MLE when training. 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.[66] 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.[132] Jaques *et al.* have also explored the application of deep *Q*-learning and *G*-learning to molecular optimization.[131] 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.

To give a flavor of what applications have been demonstrated, we will briefly present some representative works using RNNs. Bjerrum & Threlfall explore using an architecture consisting of 256 LSTM cells followed by a "time-distributed dense" output layer.[72] Their network achieved a SMILES validity rates of 98% and the property distributions for the properties tested matched the property distributions found in the training data (some key properties they looked at were synthetic accessibility score, molecular weight, log*P*, and total polar surface area). Popova *et al.* have shown how an RNN trained for generation can be further trained with reinforcement learning to generate molecules targeted towards a specific biological function - in their case they focused on the decree to which molecules bind and inhibit the JAK2 enzyme, for which much empirical data exists. They showed how their system could be used to either maximize or minimize inhibition with JAK2 and also independently discovered 793 commercially available compounds found in the ZINC database.[79] In a similar vein, Segler *et al.* fine tune an RNN to generate a "focused library" of molecules which are likely to target the 5-HT$_{2A}$ receptor.[75] Finally, Olivecrona *et al.* show how a generative model can be fine tuned to generate analogs of a particular drug (Celecoxib) or molecules which bind to the type 2 dopamine receptor.[74]

2.2 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.[133] 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".[134] 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]

2.2.1 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 $p(\boldsymbol{x},z) = p_\theta(\boldsymbol{x}|z)p(z)$ which consists of latent variables z drawn from a pre-specified prior $p(z)$ and passed into a decoder $p_\theta(\boldsymbol{x}|z)$ parametrized by parameters θ. To apply maximum likelihood learning to such a model we like to maximize the probability of each observed datapoint $p(\boldsymbol{x}) = \int p_\theta(\boldsymbol{x}|z)p(z)\mathrm{d}z$ for all datapoints in our training data. However for complicated models with many parameters θ (like neural networks) this integral is intractable to compute. The method of variational inference instead maximizes a lower bound on $\log p(\boldsymbol{x})$:

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

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

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

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

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

After several manipulations, eqn (10) can be written as

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

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

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

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

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

The encoder $q_\theta(z|\boldsymbol{x})$ is typically a neural network with parameters ϕ used to find the mean and variance functions in $\mathcal{N}(z,\mu(\boldsymbol{x}), \mathrm{diag}(\sigma^2(\boldsymbol{x})))$. The resulting "Gaussian VAE" has the advantage that the KL-divergence can be computed analytically. The parameters θ and ϕ 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.[21,121,135]

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 $q_\phi(z|\boldsymbol{x})$.[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%),[72,75,78] the molecular autoencoder generates far fewer valid SMILES. The valid SMILES rate was found to vary greatly between ≈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.[57] 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).[57,58] 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.*[57]). Further work by Dai *et al.* added additional "semantic" constraints on top of a CFG yielding a higher rate of valid SMILES (43.5%).[58] 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.[136]

2.2.2 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 $p(z)$.[137] 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:

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

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

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

2.2.3 Supervised VAEs/AAEs for property prediction & optimization. In supervised VAEs, target properties y for each molecule are incorporated into the generator in addition to the SMILES strings or other molecular representation. Fig. 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,[137] 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. One unique property they optimize after training such a system is the predicted HOMO–LUMO gap, which is important for determining a molecule's utility in organic solar cells. 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

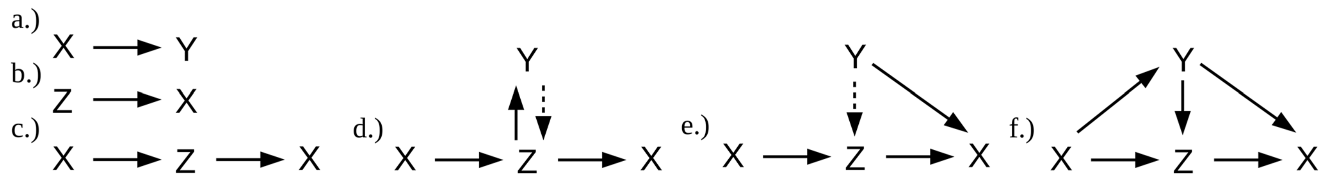


Fig. 2 Different deep learning approaches visualized as graphical models. Figure adapted from Kang *et al.*[89] Solid lines represent explicit conditional dependencies while dashed lines represent implicit conditional dependencies (arising from the relationship between X and Y inherent in the training data) for which disentanglement may be desired. a.) Regression (property prediction) b.) direct generative model c.) autoencoder d.) supervised autoencoder, type 1 (ref. 23) e.) supervised autoencoder, type 2 (ref. 88 and 111) f.) supervised/semisupervised autoencoder, type 3.[89]

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.[99,110,138]

Two other types of supervised VAEs are shown in Fig. 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:[89]

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

Kang *et al.* assume that the property values have a Gaussian distribution. Type 3 VAEs are particularly useful when y is known for only some of the training data (a semi-supervised setting). In semi-supervised VAEs, the generator is tasked with predict y and is trained on the molecules where y is known and makes a best guess prediction for the rest. In effect, when y is not known, it becomes just another latent variable and a different objective function is used (for details, see Kang *et al.*[89]).

In type 2 VAEs, property data $\boldsymbol{y}$ is embedded directly into the latent space during training.[88,111] 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 $p(z)$. Then one samples from the generator $p_\theta(\boldsymbol{x}|\boldsymbol{y},z)$. 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".[111] The issue is that when sampling we assumed that $p(z)$ is independent of $p(\boldsymbol{y})$. However, the two distributions are actually "entangled" by the implicit relationship between $\boldsymbol{x}$ and $\boldsymbol{y}$ which is in the training data (this is indicated by a dashed line in Fig. 2). For consistency, one should be sampling from $p(z|\boldsymbol{y})$. Polykovskiy *et al.* developed two "disentanglement" approaches to ameliorate this issue: learning $p(z|\boldsymbol{y})$ and forcing all $p(z|\boldsymbol{y})$ to match $p(z)$.[111]

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 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:

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

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 $\hat{\boldsymbol{x}}$.[89,139] In brief, beam search keeps the top K most likely (sub)sequences at each step of the generation.

2.3 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".[140] 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]

$$\min_G \max_D V(D,G) = \mathbb{E}_{x\in p_d(x)}\left[\log D(\boldsymbol{x})\right] + \mathbb{E}_{z\in p_z(z)}\left[\log\left(1-D\left(G(z)\right)\right)\right] \quad (19)$$

Here $p_d(x)$ 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 $\log(1 - D(G(z)))$ 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:

$$D_G^*(\boldsymbol{x}) = \frac{p_d(\boldsymbol{x})}{p_d(\boldsymbol{x}) + p_{\theta_G}(\boldsymbol{x})} \quad (20)$$

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

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

where $D_{JS}(p_d, p_{\theta_G})$ is the Jensen–Shannon divergence:

$$D_{JS}(p_d, p_{\theta_G}) = \frac{1}{2}D_{KL}\left(p_d \middle\| \frac{p_d + p_{\theta_G}}{2}\right) + \frac{1}{2}D_{KL}\left(p_{\theta_G} \middle\| \frac{p_d + p_{\theta_G}}{2}\right) \quad (22)$$

Here $D_{KL}(p,q)$ is the Kullback–Leibler (KL) divergence. Maximizing the log-likelihood is equivalent to minimizing the forward KL divergence $D_{KL}(p_d, p_{\theta_G})$.[135] To better understand what this means, we can rewrite the equation for KL divergence (eqn (11)) in a slightly different way:

$$D_{KL}(p_d, p_{\theta_G}) = \int p_d(z)\left(\log p_d(z) - \log p_{\theta_G}(z)\right)dz \quad (23)$$

This shows us that KL divergence captures the difference between p_d and p_{θ_G} weighted by p_d. Thus one of the weaknesses of maximum likelihood is that p_{θ_G} may have significant deviations from p_d when $p_d \approx 0$. To summarize, the forward KL divergence ($D_{KL}(p_d, p_{\theta_G})$) punishes models that underestimate the data distribution, while the reverse KL divergence ($D_{KL}(p_d, p_{\theta_G})$) 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_{θ_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_{θ_G} than the original GAN objective function and results in faster and more stable training.[141] 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_{θ_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:

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

$\Pi(x,y)$ 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 latest techniques, see Kurach *et al.*[142]

Several papers have emerged so far applying GANs to molecular generation—Guimaraes *et al.* (ORGAN),[103] Sanchez-Lengeling *et al.* (ORGANIC),[108] De Cao & Kipf (MolGAN),[50,63] and Putin *et al.* (RANC, ATNC).[104,105] The Objective-Reinforced GAN (ORGAN) of Guimares *et al.* uses the SMILES molecular representation and an RNN (LSTM) generator and a CNN discriminator.[103] The architecture of the ORGAN is taken from the SeqGAN of Yu *et al.*[143] 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:

$$R(S) = \lambda D(S) + (1 - \lambda)R(S) \quad (25)$$

where $\lambda \in [0, 1]$ is a tunable 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. Proof of concept applications of the ORGAN have been demonstrated in two domains – firstly for drug design it as shown how ORGAN can be used to maximize Lapinksi's rule of five metric as well as the quantitative estimate of drug likeliness metric. The second application for which ORGAN was demonstrated is maximizing the power conversion efficiency (PCE) of molecules for use in organic photovoltaics, where PCE is estimated using a machine learning based predictor that was previously developed.[108]

2.3.1 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.[70] 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[108] with its default hyperparameters suffers from a perfect discriminator problem during training, leading to a plateauing of the generator's loss.[104] To help solve these issues, Putin *et al.* implemented a differentiable neural computer (DNC) as the generator.[104] The DNC (Graves *et al.*, 2016)[144] is an extension of the neural Turing machine (Graves *et al.* 2014)[145] 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.[105]

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.

3 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.[146] 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.[140,147] 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.[148] 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,[142] and in the field of reinforcement learning it has been found that codes which purport to do the same thing will give different results.[147] 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.

3.1 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").[149,150] Interestingly, they show that these three methodologies measure different things, so good performance under one does not imply good performance under another.[150]

The "inception score" (IS) uses a third-party neural network which has been trained in a supervised manner to do classification.[149] In the original IS, Google's Inception network trained on ImageNet was used as the third-party network. IS computes the divergence between the distribution of classes predicted by the third-party neural network on generated molecules 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.[151] Inspired by this, Preuer *et al.* created the Fréchet ChemNet Distance metric for evaluating models that generate molecules.[152] 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.[142] Comparing with test data gives a more useful metric which measures generalization ability, and is advocated as a standard by Kurach *et al.*[142]

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.[153] Inspired by *MoleculeNet*, Polykovskiy and collaborators have introduced the MOlecular SEtS (MOSES) package to make it easier to build and test generative models.[154] 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, $\log P$, 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*.[155] 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.[156] 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, Im *et al.* have shown significant pitfalls to using the Inception Distance metric.[157] 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[158] and could be of use to the molecular modeling community.

3.2 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.[159] Since then, the development of better reward functions has greatly helped to mitigate such issues, but low diversity and novelty remains an issue.[160–162] 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.

3.2.1 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 $\mathscr{G}$ can be defined quite simply as:

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

A popular metric is the Tanimoto similarity between two extended-connectivity fingerprint bit vectors.[154] 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:[154]

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

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:

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

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:

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

where $|\text{set}(\mathscr{G})|$ is the number of unique molecules in the generated batch $\mathscr{G}$ and $|\mathscr{G}|$ 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. Guimaraes *et al.* define the "soft novelty" for a single molecule as:[103]

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

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 $\mathscr{T}$. Then one can look at how many molecules in a set of generated molecules $\mathscr{G}$ appear in $\mathscr{T}$ and use a novelty reward such as:[75]

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

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.*[161] use the ISOMAP method,[163] Merk *et al.*[81] use multidimensional scaling, and Segler *et al.*[75] use t-SNE projection.[164]

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 interpolation *vs.* VAEs, for reasons that are not yet fully understood. For instance with GANs trained on natural images, interpolation can be done between a z point corresponding to a frowning man to a point z' corresponding to a smiling woman, and all of the intervening points result in images which make sense.[165] Empirically most real world high dimensional datasets are found to lie on a low density manifold.[166] Ideally, the dimensionality of the latent space $p(z)$ used in a GAN, VAE, or AAE will correspond to the dimensionality of this manifold. If the dimensionality of $p(z)$ 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 a thin spherical shell. In such cases it has been found empirically that better results can be found by doing spherical linear interpolation or *slerp*.[167] 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 $0 \le t \le 1$ 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:[74]

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

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 tunable 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 run so as 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.[73] Similarly, with reinforcement learning one can simply start the agent with a particular molecule 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.[90]

3.2.2 Stability and synthesizability. Enforcement of synthesizability has thus far mainly been done using the synthetic accessibility (SA) score developed by Ertl & Schuffenhauer,[168] although other synthesizability scoring functions exist.[169,170] 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, Sanchez-Lengeling *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.[108]

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.[103] Examples of such functions include Lipinski's Rule of Five score,[5] the natural product-likeness score,[171] the quantitative estimate of drug-likeness,[172] and the Muegge metrics.[173,174] 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.[175] For energetic molecules and other niche domains an analogous set of functional groups to avoid has yet to be developed.

Many software packages exist for checking molecule's stability and synthesizability 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.[79] Bjerrum *et al.* use the Wiley ChemPlanner software post-training to find synthesis routes for 25–55% of the molecules generated by their RNN.[72] Finally, Sumita *et al.* check for previously discovered synthetic routes for their generated molecules using a SciFinder literature search.[80]

It is worth mentioning that deep learning is now being used for the prediction of chemical reactions[11,12] and synthesis planning. Segler *et al.* trained a deep reinforcement learning system on a large set of reactions that have been published in organic chemistry and demonstrated a system that could predict chemical reactions and plan synthetic routes at the level of expert chemists.[13]

3.2.3 **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 ($\log P$), and the quantitative measure of drug-likeness (QED),[172] both of which can be found in the open source package RDKit.[52]

Since physics based prediction is usually very computationally intensive, a popular approach is to train a property predicting machine learning model ahead of time. There is now an enormous literature on deep learning for property prediction demonstrating success in multiple areas.[8] Impressive results have been obtained, such as systems which can predict molecular energies at DFT accuracy,[176] and highly accurate systems which can transfer between many types of molecules.[177] While the literature on quantum property prediction is perhaps the most developed, success has been seen in other areas, such as calculating high level properties of energetic molecular crystals (such as sensitivity and detonation velocity).[10,27] Many predictive models are now published for "off the shelf" use, for instance a collection of predictive models for ADME (absorption, distribution, metabolism, and excretion) called "SwissADME" was recently published.[176]

It has also been demonstrated 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 λ^* is the target wavelength, they used the following:[80]

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

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.[80]

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

$$J(S) = \log P(S) - \mathrm{SA}(S) - \text{Ring Penalty}(S) \quad (35)$$

Yang *et al.* add an additional penalty for generating invalid SMILES which could be used more broadly:[76]

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

In the context of training the ORGAN architecture, Guimaraes *et al.* found that rotating the reward function metric from epoch to epoch had some advantages to using all metrics at once.[103] 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.

4 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.[178] With RNNs, recently developed attention mechanisms and external memory cells offer a possible avenue to improve SMILES string generation.[179]

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,[57] and ChemTS[76]) for optimizing the "penalized $\log P$ score" (eqn (35)). They found that with computer time fixed to eight hours, their method performed better or comparable to the deep learning methods. In a similar vein, Jensen found that a genetic algorithm performed better than a SMILES based RNN + Bayesian optimization, the ChemTS

RNN, and a CVAE with 100x lower computational cost.[180] It appears that genetic algorithms can explore chemical space in a very efficient manner.

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.[181] 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_{train} values in Table 2. 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 drug design and other specialized applications the 3D geometry of molecules is 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.[66]

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,[94] ORGAN,[103] and Graph Convolutional Policy Network[65] approaches when optimizing the $\log P$ and QED metrics.[66] 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[182,183]) 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[184] and microstructures.[185–187] Hierarchical GANs[188] 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 used, a process called "closing the loop". More work is needed to develop workflows and methods to interface and integrate generative models into laboratory platforms to allow for rapid feedback and cycling. A key challenge is developing useful software and cyberinfrastructure for computational screening and data management.[189] The potential for efficiency improvements *via* automated AI-assisted synthesis planning and "self-driving" robotic laboratories is quite profound.[14,15,190–192]

Conflicts of interest

There are no conflicts to declare.

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 J. A. DiMasi, H. G. Grabowski and R. W. Hansen, *J. Health. Med. Econ.*, 2016, **47**, 20–33.
2 S. M. Paul, D. S. Mytelka, C. T. Dunwiddie, C. C. Persinger, B. H. Munos, S. R. Lindborg and A. L. Schacht, *Nat. Rev. Drug Discovery*, 2010, **9**, 203–214.
3 A. Homburg, *Propellants, Explos., Pyrotech.*, 2017, **42**, 851–853.
4 P. G. Polishchuk, T. I. Madzhidov and A. Varnek, *J. Comput.-Aided Mol. Des.*, 2013, **27**, 675–679.
5 C. A. Lipinski, F. Lombardo, B. W. Dominy and P. J. Feeney, *Adv. Drug Delivery Rev.*, 1997, **23**, 3–25.
6 R. Macarron, M. N. Banks, D. Bojanic, D. J. Burns, D. A. Cirovic, T. Garyantes, D. V. S. Green, R. P. Hertzberg, W. P. Janzen, J. W. Paslay, U. Schopfer and G. S. Sittampalam, *Nat. Rev. Drug Discovery*, 2011, **10**, 188–195.
7 E. O. Pyzer-Knapp, C. Suh, R. Gómez-Bombarelli, J. Aguilera-Iparraguirre and A. Aspuru-Guzik, *Annu. Rev. Mater. Res.*, 2015, **45**, 195–216.
8 K. T. Butler, D. W. Davies, H. Cartwright, O. Isayev and A. Walsh, *Nature*, 2018, **559**, 547–555.

9 P. Raccuglia, K. C. Elbert, P. D. F. Adler, C. Falk, M. B. Wenny, A. Mollo, M. Zeller, S. A. Friedler, J. Schrier and A. J. Norquist, *Nature*, 2016, **533**, 73–76.

10 B. C. Barnes, D. C. Elton, Z. Boukouvalas, D. E. Taylor, W. D. Mattson, M. D. Fuge and P. W. Chung, 2018, arXiv e-prints:1807.06156.

11 D. Fooshee, A. Mood, E. Gutman, M. Tavakoli, G. Urban, F. Liu, N. Huynh, D. V. Vranken and P. Baldi, *Mol. Syst. Des. Eng.*, 2018, **3**, 442–452.

12 P. Schwaller, T. Gaudin, D. Lányi, C. Bekas and T. Laino, *Chem. Sci.*, 2018, **9**, 6091–6098.

13 M. H. S. Segler, M. Preuss and M. P. Waller, *Nature*, 2018, **555**, 604–610.

14 A. B. Henson, P. S. Gromski and L. Cronin, *ACS Cent. Sci.*, 2018, **4**, 793–804.

15 L. M. Roch, F. Häse, C. Kreisbeck, T. Tamayo-Mendoza, L. P. E. Yunker, J. E. Hein and A. Aspuru-Guzik, *Sci. Robot.*, 2018, **3**, eaat5559.

16 D. Cireşan, U. Meier and J. Schmidhuber, *2012 IEEE Conference on Computer Vision and Pattern Recognition*, 2012, pp. 3642–3649.

17 A. Krizhevsky, I. Sutskever and G. E. Hinton, in *Advances in Neural Information Processing Systems 25*, ed. F. Pereira, C. J. C. Burges, L. Bottou and K. Q. Weinberger, Curran Associates, Inc., 2012, pp. 1097–1105.

18 G. E. Dahl, N. Jaitly and R. Salakhutdinov, 2014, arXiv e-prints:1406.1231.

19 G. E. Hinton, N. Srivastava, A. Krizhevsky, I. Sutskever and R. R. Salakhutdinov, 2012, arXiv e-prints:1207.0580.

20 A. Krizhevsky, I. Sutskever and G. E. Hinton, *Commun. ACM*, 2017, **60**, 84–90.

21 D. P. Kingma and M. Welling, 2013, arXiv e-prints: 1312.6114.

22 I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville and Y. Bengio, in *Advances in Neural Information Processing Systems 27*, ed. Z. Ghahramani, M. Welling, C. Cortes, N. D. Lawrence and K. Q. Weinberger, Curran Associates, Inc., 2014, pp. 2672–2680.

23 R. Gómez-Bombarelli, J. N. Wei, D. Duvenaud, J. M. Hernández-Lobato, B. Sánchez-Lengeling, D. Sheberla, J. Aguilera-Iparraguirre, T. D. Hirzel, R. P. Adams and A. Aspuru-Guzik, *ACS Cent. Sci.*, 2018, **4**, 268–276.

24 E. J. Griffen, A. G. Dossetter, A. G. Leach and S. Montague, *Drug Discovery Today*, 2018, **23**, 1373–1384.

25 R. Gómez-Bombarelli, J. Aguilera-Iparraguirre, T. D. Hirzel, D. Duvenaud, D. Maclaurin, M. A. Blood-Forsythe, H. S. Chae, M. Einzinger, D.-G. Ha, T. Wu, G. Markopoulos, S. Jeon, H. Kang, H. Miyazaki, M. Numata, S. Kim, W. Huang, S. I. Hong, M. Baldo, R. P. Adams and A. Aspuru-Guzik, *Nat. Mater.*, 2016, **15**, 1120–1127.

26 P. B. Jørgensen, M. Mesta, S. Shil, J. M. G. Lastra, K. W. Jacobsen, K. S. Thygesen and M. N. Schmidt, *J. Chem. Phys.*, 2018, **148**, 241735.

27 D. C. Elton, Z. Boukouvalas, M. S. Butrico, M. D. Fuge and P. W. Chung, *Sci. Rep.*, 2018, **8**, 9059.

28 B. C. Rinderspacher and J. M. Elward, *Mol. Syst. Des. Eng.*, 2018, **3**, 485–495.

29 H. Li, C. R. Collins, T. G. Ribelli, K. Matyjaszewski, G. J. Gordon, T. Kowalewski and D. J. Yaron, *Mol. Syst. Des. Eng.*, 2018, **3**, 496–508.

30 D. Nagarajan, T. Nagarajan, N. Roy, O. Kulkarni, S. Ravichandran, M. Mishra, D. Chakravortty and N. Chandra, *J. Biol. Chem.*, 2017, **293**, 3492–3509.

31 A. T. Müller, J. A. Hiss and G. Schneider, *J. Chem. Inf. Model.*, 2018, **58**, 472–479.

32 F. Grisoni, C. S. Neuhaus, G. Gabernet, A. T. Müller, J. A. Hiss and G. Schneider, *ChemMedChem*, 2018, **13**, 1300–1302.

33 X. Shen, T. Zhang, S. Broderick and K. Rajan, *Mol. Syst. Des. Eng.*, 2018, **3**, 826–838.

34 Y. He, E. D. Cubuk, M. D. Allendorf and E. J. Reed, *J. Phys. Chem. Lett.*, 2018, **9**, 4562–4569.

35 B. Pirard, *Expert Opin. Drug Discovery*, 2011, **6**, 225–231.

36 L.-P. Wang, A. Titov, R. McGibbon, F. Liu, V. S. Pande and T. J. Martínez, *Nat. Chem.*, 2014, **6**, 1044–1048.

37 J. Besnard, G. F. Ruda, V. Setola, K. Abecassis, R. M. Rodriguiz, X.-P. Huang, S. Norval, M. F. Sassano, A. I. Shin, L. A. Webster, F. R. C. Simeons, L. Stojanovski, A. Prat, N. G. Seidah, D. B. Constam, G. R. Bickerton, K. D. Read, W. C. Wetsel, I. H. Gilbert, B. L. Roth and A. L. Hopkins, *Nature*, 2012, **492**, 215–220.

38 N. Yoshikawa, K. Terayama, M. Sumita, T. Homma, K. Oono and K. Tsuda, *Chem. Lett.*, 2018, **47**, 1431–1434.

39 A. Daina, O. Michielin and V. Zoete, *Sci. Rep.*, 2017, **7**, 2717.

40 D. Kuzminykh, D. Polykovskiy, A. Kadurin, A. Zhebrak, I. Baskov, S. Nikolenko, R. Shayakhmetov and A. Zhavoronkov, *Mol. Pharmaceutics*, 2018, 4378–4385.

41 M. Skalic, J. Jiménez Luna, D. Sabbadin and G. De Fabritiis, *J. Chem. Inf. Model.*, 2019, **59**(3), 1205–1214.

42 A. Amidi, S. Amidi, D. Vlachakis, V. Megalooikonomou, N. Paragios and E. I. Zacharaki, *PeerJ*, 2018, **6**, e4750.

43 M. Hirn, S. Mallat and N. Poilvert, *Multiscale Model. Simul.*, 2017, **15**, 827–863.

44 M. Eickenberg, G. Exarchakis, M. Hirn and S. Mallat, in *Advances in Neural Information Processing Systems 30*, ed. I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan and R. Garnett, Curran Associates, Inc., 2017, pp. 6540–6549.

45 N. Thomas, T. Smidt, S. Kearnes, L. Yang, L. Li, K. Kohlhoff and P. Riley, *Tensor field networks: Rotation- and translation-equivariant neural networks for 3D point clouds*, 2018.

46 N. M. O'Boyle, M. Banck, C. A. James, C. Morley, T. Vandermeersch and G. R. Hutchison, *J. Cheminf.*, 2011, **3**, 33.

47 *The Open Babel Package*, http://www.openbabel.org.

48 D. Weininger, *J. Chem. Inf. Model.*, 1988, **28**(1), 31–36.

49 E. Jang, S. Gu and B. Poole, 2016, arXiv e-prints:1611.01144.

50 N. De Cao and T. Kipf, *ICML 2018 workshop on Theoretical Foundations and Applications of Deep Generative Models*, 2018.

51 M. Swain, *MolVS*, https://github.com/mcs07/MolVS.
52 G. Landrum, *RDKit: Open-source cheminformatics*, http://www.rdkit.org.
53 E. J. Bjerrum and B. Sattarov, 2018, arXiv e-prints:1806.09300.
54 E. J. Bjerrum, 2017, arXiv e-prints:1703.07076.
55 S. Heller, A. McNaught, S. Stein, D. Tchekhovskoi and I. Pletnev, *J. Cheminf.*, 2013, **5**, 7.
56 R. Winter, F. Montanari, F. Noé and D.-A. Clevert, *Chem. Sci.*, 2019, **10**, 1692–1701.
57 M. J. Kusner, B. Paige and J. M. Hernández-Lobato, 2017, arXiv e-prints:1703.01925.
58 H. Dai, Y. Tian, B. Dai, S. Skiena and L. Song, 2018, arXiv e-prints:1802.08786.
59 Y. Li, O. Vinyals, C. Dyer, R. Pascanu and P. Battaglia, 2018, arXiv e-prints:1803.03324.
60 Y. Li, L. Zhang and Z. Liu, *J. Cheminf.*, 2018, **10**, 33.
61 M. Suzuki, H. Nagamochi and T. Akutsu, *J. Cheminf.*, 2014, **6**, 31.
62 G. B. Goh, C. Siegel, A. Vishnu, N. O. Hodas and N. Baker, 2017, arXiv e-prints:1706.06689.
63 N. De Cao and T. Kipf, 2018, arXiv e-prints:1805.11973.
64 M. Simonovsky and N. Komodakis, 2018, arXiv e-prints:1802.03480.
65 J. You, B. Liu, R. Ying, V. Pande and J. Leskovec, 2018, arXiv e-prints:-1806.02473.
66 Z. Zhou, S. Kearnes, L. Li, R. N. Zare and P. Riley, 2018, arXiv e-prints:1810.08678.
67 Z. Boukouvalas, D. C. Elton, P. W. Chung and M. D. Fuge, 2018, arXiv e-prints:1811.00628.
68 J. L. Durant, B. A. Leland, D. R. Henry and J. G. Nourse, *J. Chem. Inf. Comput. Sci.*, 2002, **42**, 1273–1280.
69 A. Kadurin, A. Aliper, A. Kazennov, P. Mamoshina, Q. Vanhaelen, K. Khrabrov and A. Zhavoronkov, *Oncotarget*, 2016, **8**(7), 10883–10890.
70 A. Kadurin, S. Nikolenko, K. Khrabrov, A. Aliper and A. Zhavoronkov, *Mol. Pharmaceutics*, 2017, **14**, 3098–3104.
71 M. Simonovsky and N. Komodakis, *2017 IEEE Conference on Computer Vision and Pattern Recognition*, CVPR 2017, Honolulu, HI, USA, July 21–26, 2017, vol. 2017, pp. 29–38.
72 E. J. Bjerrum and R. Threlfall, 2017, arXiv e-prints:1705.04612.
73 A. Gupta, A. T. Müller, B. J. H. Huisman, J. A. Fuchs, P. Schneider and G. Schneider, *Mol. Inf.*, 2017, **37**, 1700111.
74 M. Olivecrona, T. Blaschke, O. Engkvist and H. Chen, *J. Cheminf.*, 2017, **9**, 48.
75 M. H. S. Segler, T. Kogej, C. Tyrchan and M. P. Waller, *ACS Cent. Sci.*, 2017, **4**, 120–131.
76 X. Yang, J. Zhang, K. Yoshizoe, K. Terayama and K. Tsuda, *Sci. Technol. Adv. Mater.*, 2017, **18**, 972–976.
77 M. Cherti, B. Kégl and A. Kazakçı, *International Conference on Learning Representations, workshop track*, Toulon, France, 2017.
78 D. Neil, M. Segler, L. Guasch, M. Ahmed, D. Plumbley, M. Sellwood and N. Brown, *International Conference on Learning Representations*, 2018.
79 M. Popova, O. Isayev and A. Tropsha, *Sci. Adv.*, 2018, **4**, eaap7885.
80 M. Sumita, X. Yang, S. Ishihara, R. Tamura and K. Tsuda, *ACS Cent. Sci.*, 2018, **4**, 1126–1133.
81 D. Merk, L. Friedrich, F. Grisoni and G. Schneider, *Mol. Inf.*, 2018, **37**, 1700153.
82 D. Merk, F. Grisoni, L. Friedrich and G. Schneider, *Communications Chemistry*, 2018, **1**, 68.
83 P. Ertl, R. Lewis, E. Martin and V. Polyakov, 2017, arXiv e-prints:1712.07449.
84 J. Arús-Pous, T. Blaschke, S. Ulander, J.-L. Reymond, H. Chen and O. Engkvist, *ChemRxiv preprint*, 2018.
85 S. Zheng, X. Yan, Q. Gu, Y. Yang, Y. Du, Y. Lu and J. Xu, *J. Cheminf.*, 2019, **11**, 5.
86 P. Pogány, N. Arad, S. Genway and S. D. Pickett, *J. Chem. Inf. Model.*, 2018, **59**(3), 1136–1146.
87 T. Blaschke, M. Olivecrona, O. Engkvist, J. Bajorath and H. Chen, *Mol. Inf.*, 2017, **37**, 1700123.
88 J. Lim, S. Ryu, J. W. Kim and W. Y. Kim, *J. Cheminf.*, 2018, **10**, 31.
89 S. Kang and K. Cho, *J. Chem. Inf. Model.*, 2018, **59**(1), 43–52.
90 S. Harel and K. Radinsky, *Mol. Pharmaceutics*, 2018, **15**, 4406–4416.
91 B. Sattarov, I. I. Baskin, D. Horvath, G. Marcou, E. J. Bjerrum and A. Varnek, *J. Chem. Inf. Model.*, 2019, **59**(3), 1182–1196.
92 M. J. Kusner, B. Paige and J. M. Hernández-Lobato, 2017, arXiv e-prints:1703.01925.
93 P. B. Jørgensen, M. N. Schmidt and O. Winther, *Mol. Inf.*, 2018, **37**, 1700133.
94 W. Jin, R. Barzilay and T. S. Jaakkola, *International Conference on Learning Representations*, 2018.
95 W. Jin, K. Yang, R. Barzilay and T. Jaakkola, *International Conference on Learning Representations*, 2019.
96 Q. Liu, M. Allamanis, M. Brockschmidt and A. L. Gaunt, 2018, arXiv e-prints:1805.09076.
97 H. Kajino, 2018, arXiv e-prints:1803.03324.
98 R. Winter, F. Montanari, F. Noé and D.-A. Clevert, *ChemRxiv preprint*, 2018.
99 B. Samanta, A. De, N. Ganguly and M. Gomez-Rodriguez, 2018, arXiv e-prints:1802.05283.
100 B. Samanta, A. De, G. Jana, P. K. Chattaraj, N. Ganguly and M. Gomez-Rodriguez, 2018, arXiv e-prints:1802.05283.
101 T. Ma, J. Chen and C. Xiao, in *Advances in Neural Information Processing Systems 32*, 2018.
102 S. M. Kearnes, L. Li and P. Riley, 2019, arXiv e-prints:1904.08915.
103 G. Lima Guimaraes, B. Sanchez-Lengeling, C. Outeiral, P. L. Cunha Farias and A. Aspuru-Guzik, 2017, arXiv e-prints:1705.10843.
104 E. Putin, A. Asadulaev, Y. Ivanenkov, V. Aladinskiy, B. Sanchez-Lengeling, A. Aspuru-Guzik and A. Zhavoronkov, *J. Chem. Inf. Model.*, 2018, **58**, 1194–1204.
105 E. Putin, A. Asadulaev, Q. Vanhaelen, Y. Ivanenkov, A. V. Aladinskaya, A. Aliper and A. Zhavoronkov, *Mol. Pharmaceutics*, 2018, **15**(10), 4386–4397.
106 O. Méndez-Lucio, B. Baillif, D.-A. Clevert, D. Rouquié and J. Wichard, *ChemRxiv preprint*, 2018.

107 L. Maziarka, A. Pocha, J. Kaczmarczyk, K. Rataj and M. Warchoł, *Mol-CycleGAN - a generative model for molecular optimization*, 2019.

108 B. Sanchez-Lengeling, C. Outeiral, G. L. Guimaraes and A. Aspuru-Guzik, *ChemRxiv preprint*, 2017.

109 D. Grattarola, L. Livi and C. Alippi, 2018, arXiv e-prints:1812.04314.

110 H. Ikebata, K. Hongo, T. Isomura, R. Maezono and R. Yoshida, *J. Comput.-Aided Mol. Des.*, 2017, **31**, 379–391.

111 D. Polykovskiy, A. Zhebrak, D. Vetrov, Y. Ivanenkov, V. Aladinskiy, P. Mamoshina, M. Bozdaganyan, A. Aliper, A. Zhavoronkov and A. Kadurin, *Mol. Pharmaceutics*, 2018, **15**(10), 4398–4405.

112 N. Ståhl, G. Falkman, A. Karlsson, G. Mathiason and J. Bostrom, *ChemRxiv e-print*, 2019.

113 L. C. Blum and J.-L. Reymond, *J. Am. Chem. Soc.*, 2009, **131**, 8732–8733.

114 T. Sterling and J. J. Irwin, *J. Chem. Inf. Model.*, 2015, **55**, 2324–2337.

115 L. Ruddigkeit, R. van Deursen, L. C. Blum and J.-L. Reymond, *J. Chem. Inf. Model.*, 2012, **52**, 2864–2875.

116 M. Nakata and T. Shimazaki, *J. Chem. Inf. Model.*, 2017, **57**, 1300–1308.

117 R. Ramakrishnan, P. O. Dral, M. Rupp and O. A. von Lilienfeld, *Sci. Data*, 2014, **1**, 14022.

118 S. Chakraborty, P. Kayastha and R. Ramakrishnan, *J. Chem. Phys.*, 2019, **150**, 114106.

119 S. A. Lopez, E. O. Pyzer-Knapp, G. N. Simm, T. Lutzow, K. Li, L. R. Seress, J. Hachmann and A. Aspuru-Guzik, *Sci. Data*, 2016, **3**, 160086.

120 A. Géron, *Hands-On Machine Learning with Scikit-Learn and TensorFlow: Concepts, Tools, and Techniques to Build Intelligent Systems*, O'Reilly Media, Inc., 2017, 1st edn.

121 I. Goodfellow, Y. Bengio and A. Courville, *Deep Learning*, MIT Press, 2016.

122 S. Hochreiter and J. Schmidhuber, *Neural Comput.*, 1997, **9**, 1735–1780.

123 K. Cho, B. van Merrienboer, C. Gulcehre, D. Bahdanau, F. Bougares, H. Schwenk and Y. Bengio, *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2014, pp. 1724–1734.

124 M. Ranzato, S. Chopra, M. Auli and W. Zaremba, 2015, arXiv:abs/1511.06732.

125 A. Venkatraman, M. Hebert and J. A. Bagnell, *Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence*, 2015, pp. 3024–3030.

126 S. Bengio, O. Vinyals, N. Jaitly and N. Shazeer, *Proceedings of the 28th International Conference on Neural Information Processing Systems*, Cambridge, MA, USA, 2015, vol. 1, pp. 1171–1179.

127 F. Huszár, 2015, arXiv e-prints:1511.05101.

128 R. J. Williams and D. Zipser, *Neural Comput.*, 1989, **1**, 270–280.

129 R. Gómez-Bombarelli, *Broad Institute Models, Inference, & Algorithms talk:* "*Deep learning chemical space*", 2017, https://www.youtube.com/watch?v=ieZhnnvjyWU.

130 R. J. Williams, *Machine Learning*, 1992, vol. 8, pp. 229–256.

131 N. Jaques, S. Gu, D. Bahdanau, J. M. Hernádez-Lobato, R. E. Turner and D. Eck, *International Conference on Machine Learning*, 2017.

132 V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski, S. Petersen, C. Beattie, A. Sadik, I. Antonoglou, H. King, D. Kumaran, D. Wierstra, S. Legg and D. Hassabis, *Nature*, 2015, **518**, 529–533.

133 G. Hinton and R. Salakhutdinov, *Science*, 2006, **313**, 504–507.

134 R. Salakhutdinov and G. Hinton, *Proceedings of the Twelth International Conference on Artificial Intelligence and Statistics*, Hilton Clearwater Beach Resort, Clearwater Beach, Florida USA, 2009, pp. 448–455.

135 P. Mehta, M. Bukov, C.-H. Wang, A. G. R. Day, C. Richardson, C. K. Fisher and D. J. Schwab, 2018, arXiv e-prints:1803.08823.

136 D. Janz, J. van der Westhuizen and J. M. Hernández-Lobato, 2017, arXiv e-prints:1708.04465.

137 A. Makhzani, J. Shlens, N. Jaitly and I. Goodfellow, *International Conference on Learning Representations*, 2016.

138 R.-R. Griffiths and J. M. Hernández-Lobato, 2017, arXiv e-prints:1709.05501.

139 I. Sutskever, O. Vinyals and Q. V. Le, 2014, arXiv e-prints: 1409.3215.

140 M. Lucic, K. Kurach, M. Michalski, S. Gelly and O. Bousquet, 2017, arXiv e-prints:1711.10337.

141 M. Arjovsky, S. Chintala and L. Bottou, 2017, arXiv e-prints: 1701.07875.

142 K. Kurach, M. Lucic, X. Zhai, M. Michalski and S. Gelly, 2018, arXiv e-prints:1807.04720.

143 L. Yu, W. Zhang, J. Wang and Y. Yu, *Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence*, February 4–9, 2017, San Francisco, California, USA., 2017, pp. 2852–2858.

144 A. Graves, G. Wayne, M. Reynolds, T. Harley, I. Danihelka, A. Grabska-Barwińska, S. G. Colmenarejo, E. Grefenstette, T. Ramalho, J. Agapiou, A. P. Badia, K. M. Hermann, Y. Zwols, G. Ostrovski, A. Cain, H. King, C. Summerfield, P. Blunsom, K. Kavukcuoglu and D. Hassabis, *Nature*, 2016, **538**, 471–476.

145 A. Graves, G. Wayne and I. Danihelka, 2014, arXiv e-prints: 1807.06156.

146 D. Sculley, J. Snoek, A. Wiltschko and A. Rahimi, *Sixth International Conference on Learning Representations - Workshop Track*, 2018.

147 P. Henderson, R. Islam, P. Bachman, J. Pineau, D. Precup and D. Meger, *Thirthy-Second AAAI Conference On Artificial Intelligence*, 2018.

148 G. Melis, C. Dyer and P. Blunsom, 2017, arXiv e-prints:1707.05589.

149 T. Salimans, I. J. Goodfellow, W. Zaremba, V. Cheung, A. Radford and X. Chen, 2016, arXiv e-prints:1606.03498.

150 L. Theis, A. van den Oord and M. Bethge, *International Conference on Learning Representations*, 2016.

151 M. Heusel, H. Ramsauer, T. Unterthiner, B. Nessler and S. Hochreiter, *GANs Trained by a Two Time-Scale Update Rule Converge to a Local Nash Equilibrium*, 2017.

152 K. Preuer, P. Renz, T. Unterthiner, S. Hochreiter and G. Klambauer, *J. Chem. Inf. Model.*, 2018, **58**, 1736–1741.

153 Z. Wu, B. Ramsundar, E. N. Feinberg, J. Gomes, C. Geniesse, A. S. Pappu, K. Leswing and V. Pande, *Chem. Sci.*, 2018, **9**, 513–530.

154 D. Polykovskiy, A. Zhebrak, B. Sanchez-Lengeling, S. Golovanov, O. Tatanov, S. Belyaev, R. Kurbanov, A. Artamonov, V. Aladinskiy, M. Veselov, A. Kadurin, S. Nikolenko, A. Aspuru-Guzik and A. Zhavoronkov, 2018, arXiv e-prints:: arXiv:1811.12823.

155 M. Benhenda, E. J. Bjerrum, H. Yi and C. Zaveri, *Authorea preprint*, 2018.

156 N. Brown, M. Fiscato, M. H. Segler and A. C. Vaucher, *J. Chem. Inf. Model.*, 2019, **59**(3), 1096–1108.

157 D. J. Im, A. H. Ma, G. W. Taylor and K. Branson, *International Conference on Learning Representations*, 2018.

158 I. Gulrajani, C. Raffel and L. Metz, *International Conference on Learning Representations*, 2019.

159 D. 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.

160 M. Benhenda, 2017, arXiv e-prints:1703.01925.

161 N. Yoshikawa, K. Terayama, T. Honma, K. Oono and K. Tsuda, 2018, arXiv e-prints:1804.02134.

162 J. Panteleev, H. Gao and L. Jia, *Bioorg. Med. Chem. Lett.*, 2018, **28**, 2807–2815.

163 J. B. Tenenbaum, *Science*, 2000, **290**, 2319–2323.

164 L. van der Maaten and G. Hinton, *J. Mach. Learn. Res.*, 2008, **9**, 2579–2605.

165 A. Radford, L. Metz and S. Chintala, *International Conference on Learning Representations*, 2016.

166 P. Domingos, *Commun. ACM*, 2012, **55**, 78–87.

167 T. White, 2016, arXiv e-prints:1609.04468.

168 P. Ertl and A. Schuffenhauer, *J. Cheminf.*, 2009, **1**, 8.

169 Y. Podolyan, M. A. Walters and G. Karypis, *J. Chem. Inf. Model.*, 2010, **50**, 979–991.

170 Y. Fukunishi, T. Kurosawa, Y. Mikami and H. Nakamura, *J. Chem. Inf. Model.*, 2014, **54**, 3259–3267.

171 P. Ertl, S. Roggo and A. Schuffenhauer, *J. Chem. Inf. Model.*, 2008, **48**, 68–74.

172 G. R. Bickerton, G. V. Paolini, J. Besnard, S. Muresan and A. L. Hopkins, *Nat. Chem.*, 2012, **4**, 90–98.

173 I. Muegge, S. L. Heald and D. Brittelli, *J. Med. Chem.*, 2001, **44**, 1841–1846.

174 I. Muegge, *Med. Res. Rev.*, 2003, **23**, 302–321.

175 A. Kalgutkar, I. Gardner, R. Obach, C. Shaffer, E. Callegari, K. Henne, A. Mutlib, D. Dalvie, J. Lee, Y. Nakai, J. O'Donnell, J. Boer and S. Harriman, *Curr. Drug Metab.*, 2005, **6**, 161–225.

176 F. A. Faber, L. Hutchison, B. Huang, J. Gilmer, S. S. Schoenholz, G. E. Dahl, O. Vinyals, S. Kearnes, P. F. Riley and O. A. von Lilienfeld, *J. Chem. Theory Comput.*, 2017, **13**, 5255–5264.

177 L. Cheng, M. Welborn, A. S. Christensen and T. F. Miller III, *J. Chem. Phys.*, 2019, **150**, 131103.

178 T. Che, Y. Li, R. Zhang, R. D. Hjelm, W. Li, Y. Song and Y. Bengio, *Maximum-Likelihood Augmented Discrete Generative Adversarial Networks*, 2017.

179 C. Olah and S. Carter, *Distill*, 2016, **1**.

180 J. H. Jensen, *Chem. Sci.*, 2019, **10**, 3567–3572.

181 Z. Boukouvalas, 2018, arXiv e-prints: 1801.08600.

182 E. C. Alley, G. Khimulya, S. Biswas, M. AlQuraishi and G. M. Church, 2019, bioRxiv e-prints:10.1101/589333v1.

183 N. Anand and P. Huang, in *Advances in Neural Information Processing Systems 31*, ed. S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi and R. Garnett, Curran Associates, Inc., 2018, pp. 7494–7505.

184 A. Nouira, N. Sokolovska and J.-C. Crivello, 2018, arXiv e-prints:1810.11203.

185 X. Li, Z. Yang, L. C. Brinson, A. Choudhary, A. Agrawal and W. Chen, *Volume 2B: 44th Design Automation Conference*, 2018.

186 R. Singh, V. Shah, B. Pokuri, S. Sarkar, B. Ganapathysubramanian and C. Hegde, 2018, arXiv e-prints:1811.09669.

187 Z. Yang, X. Li, L. C. Brinson, A. N. Choudhary, W. Chen and A. Agrawal, 2018, arXiv e-prints:1805.02791.

188 W. Chen, A. Jeyaseelan and M. D. Fuge, *ASME 2016 International Design Engineering Technical Conferences and Computers and Information in Engineering Conference*, Quebec City, Canada, 2018.

189 J. Hachmann, M. A. F. Afzal, M. Haghighatlari and Y. Pal, *Mol. Simul.*, 2018, **44**, 921–929.

190 S. K. Saikin, C. Kreisbeck, D. Sheberla, J. S. Becker and A. Aspuru-Guzik, *Expert Opin. Drug Discovery*, 2018, **14**, 1–4.

191 P. S. Gromski, A. B. Henson, J. M. Granda and L. Cronin, *Nat. Rev. Chem.*, 2019, **3**, 119–128.

192 D. P. Tabor, L. M. Roch, S. K. Saikin, C. Kreisbeck, D. Sheberla, J. H. Montoya, S. Dwaraknath, M. Aykol, C. Ortiz, H. Tribukait, C. Amador-Bedolla, C. J. Brabec, B. Maruyama, K. A. Persson and A. Aspuru-Guzik, *Nat. Rev. Mater.*, 2018, **3**, 5–20.