Review

Data-driven algorithms for inverse design of polymers

Kianoosh Sattari[a], Yunchao Xie*[a], Jian Lin*[a, b]

Received 00th January 20xx,
Accepted 00th January 20xx

DOI: 10.1039/x0xx00000x

The ever-increasing demand for novel polymers with superior properties requires a deeper understanding and exploration of the chemical space. Recently, data-driven approaches to explore the chemical space for polymer design are emerging. Among them, inverse design strategies for designing polymers with specific properties have evolved to be a significant materials informatics platform via learning hidden knowledge from materials data as well as smartly navigating the chemical space in an optimized way. In this review, we first summarize the progress on the representation of polymers, a prerequisite step for the inverse design of polymers. Then, we systematically introduce three data-driven strategies implemented for the inverse design of polymers, i.e., high-throughput virtual screening, global optimization, and generative models. Finally, we discuss the challenges and opportunities of the data-driven strategies as well as optimization algorithms employed in the inverse design of polymers.

Keywords: Machine learning, deep learning, inverse design, polymers, representation, generative models

1. Introduction

Polymers have become deeply integrated into both human daily life and high technology due to a plethora of attractive physical, chemical, and electrical properties. These ubiquitous and highly tunable properties of polymers mainly arise from extraordinary diversity at both micro and macro scales.[1-4] Though only containing few elements in the periodic table, polymers exhibit versatile functionality *via* finely tuning the atomic-level connectivity, chain packing, crystallinity, phases, and morphology. Benefitting from these properties, polymers have found widespread applications including biology, medicine, and engineering.[5]

The design of novel polymer materials has been gone through three stages of development. In the first stage, scientists rely on experimentally-driven trial-and-error approaches to invent materials, such as penicillin, Vaseline, and Teflon.[6] A trial-and-error approach involves significant domain knowledge. It starts from defining a problem or hypothesis followed by testing with a proposed solution, finally learning from failure for the next iteration.[7] Using the domain knowledge, the scientists narrow down the design space to limited amount of candidates for validation. However, the involved strategy in this stage has limitations, such as by-chance discovery and preparation from common chemical compounds found in nature, thus limiting their potential for the next innovations. Moreover, they are extremely time-, labor-, and cost-consuming.[8-11] In the second stage, researchers adopt high-throughput experiments or virtual screening to determine the relevant properties of enormous targets, and they choose the best ones for further optimization.[12-15] Even though those approaches have been improved by high-throughput simulations[16], high-performance computing (HPC)[17], and GPU accelerated modules,[18] such a research strategy still lags the pace of the ever-increasing demands on the polymers with superior properties. Even for small molecules, the number of structures is estimated to be on the order of 10^{60}, making an efficient and thorough search impossible by traditional experiment and computation-based approaches.[19]Hence, it is urgent to solve these problems to accelerate the design of polymers to meet the ever-increasing demands. In the third stage, a research paradigm tackles the ‘materials-property’ problem in an ‘inverted’ manner, which approaches the ‘desired properties-to-appropriate materials’ procedure, or called “inverse design”, instead of a forward ‘structure-to-property’ procedure. With advances in machine learning (ML) and deep learning (DL), inverse design, a new research paradigm, has emerged as an efficient tool to navigate the design space. AI is being used for predicting properties of polymers, seeking a mapping function relating a structure to the property of choice.[6, 20-28] Deep generative models seek to learn the underlying probability distribution of structures and their corresponding properties for connecting them in a nonlinear way.[6] The DL algorithms can also act as the recommender systems for hypothesis generation about experimental conditions that are likely to produce polymers,[29, 30] which, however, is not the focus of this review.

For polymers, stochastic macromolecules, establishing the exact recipes of polymer chains especially those possessing

[a] Department of Mechanical and Aerospace Engineering, University of Missouri, Columbia, MO 65211, USA
[b] Department of Electrical Engineering and Computer Science, University of Missouri, Columbia, MO 65211, USA
*E-mails: linjian@missouri.edu or yxpx3@mail.missouri.edu

cross-links or network interpenetration is impractical. Indeed, defining all the atoms in complex polymers is not practical since the input representations are computationally expensive. Instead of directly using all sequenced atoms in a polymer chain as the source of feature representations, alternatives, such as chemical compounds or functional groups, can be more efficient to represent polymers.[29] Even for complicated polymers, one needs to start with designing monomers or building blocks since many characteristics of polymers are transferred by their building blocks. There exist several works on inverse molecule design using different architectures,[31-35] as well as thorough reviews in this area.[6, 36, 37] Polymer inverse design, however, is still in its infancy and will bring up increased attention like other complex materials such as crystalline porous materials in the future.[38] Ferguson and Ranganathan reviewed improvements in data-driven protein design, one other member of macromolecules, which can be useful for polymer design studies.[30] Sherman et. al. reviewed recent advances in inverse design of soft materials.[39] They particularly addressed methodological limitations and computational challenges that constrain the size and complexity of materials that can be designed.

A typical flowchart of inverse design of polymers using DL can be described as the following four steps. 1) Data preparation. In polymer research, it is still a challenge to find or generate a sufficient volume of data. Such data can be created from experiments. Or high throughput computations using first-principle theory, density functional theory (DFT), classical MD, and coarse-grained (CG) modeling can be also used to generate polymer data.[17, 40, 41] Webb et al. used CG modeling to simulate polymers to construct a database for developing machine learning models.[41] Another source of data can be mined from scientific literature or publicly available patents.[40] For instance, PoLyInfo, an open-source database, includes information of different polymers homopolymers, copolymers, and polymer blends.[42] 2) Polymer representations. Followed by data collection is the numerical representation of both structures and properties of polymers. Representations can use the approaches from a complex and expensive one such as 3D coordinates to a compact and cheap string-based one such as SMILES. 3) Development of the DL algorithms for inverse design. ML-based prediction models can be used in the inverse design process to direct the generator toward the best candidates. 4) Validation. Validation of the best candidates can be through either computation or experiment or both. Computational validations in different scales are faster and cost less compared to experimental evaluation. After validation with simulation, one can choose the best candidates for experimental evaluation.

We will mainly focus on the state-of-the-art data-driven algorithms for inverse design of polymers, reviewing several promising case studies, and elaborating future opportunities in chemical, biomedical, and materials science fields. The review focuses on Steps 2 and 3 from the mentioned workflow. Although the importance of the predictors in the inverse design process cannot be overemphasized, in this review, we mainly focus on deep learning and optimization algorithms that can efficiently navigate the design space. Their correlation is schematically represented in Fig. 1. The schematic shows two different directions of forward and inverse design. One may transfer knowledge that is obtained from well-studied ML and DL algorithms for molecular property prediction and inverse molecular design to the polymer field. If successful, a new research paradigm for complex polymer design can be shifted from an intuitive one to an on-demand and determinative one.

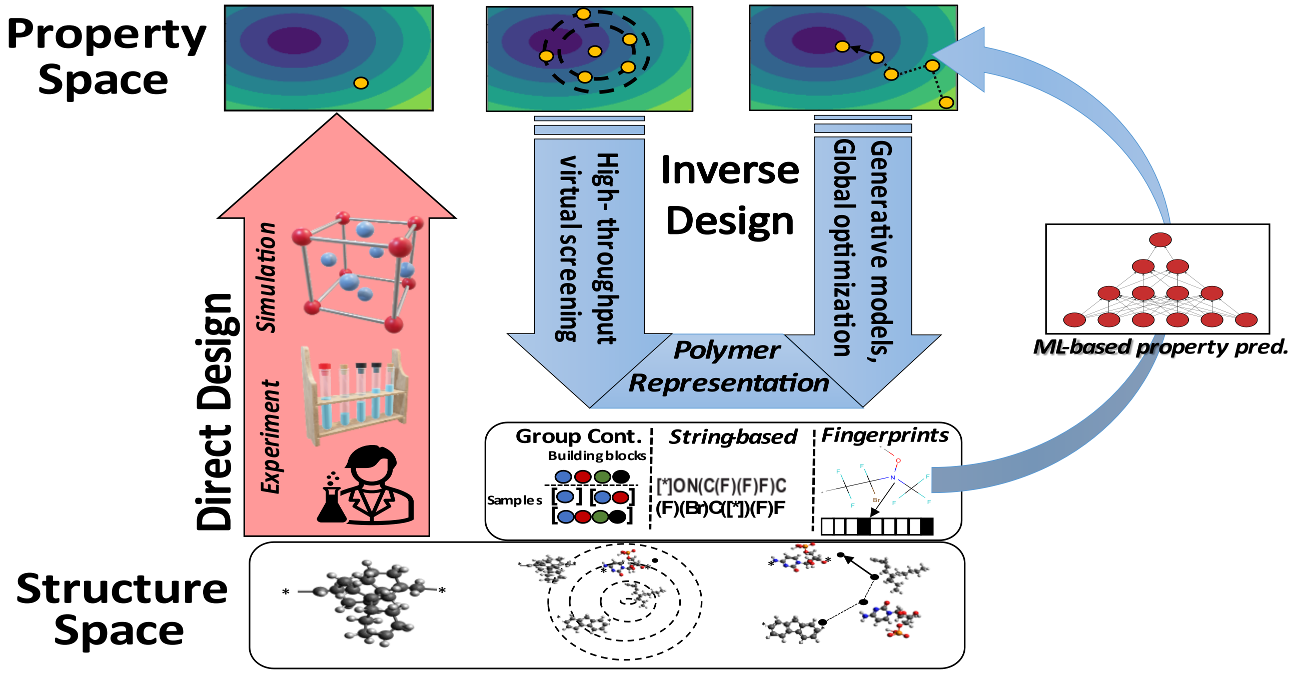

Fig. 1 Schematic of forward and inverse materials design. Experiment and simulation from direct design map the structures to the properties. Inverse design starts with desired properties and generates candidates. Polymer representation is used to numerically introduce the polymers for ML-based models.

2. Search/design space

As human researchers, we can operate in an unconstrained design space.[43] The design space can be defined by discrete or continuous variables.[43] To realize the goal of inverse material design, one needs to define the design space by deciding both the input representation (descriptors or features as defined in Section 3) and a model family (e.g. deep neural networks as discussed in Section 4). If all possible input parameters were considered, the design space would be massive, while, in most cases, the final model is only restricted to a defined space trained from random initialization. Thus, defining an appropriate design space would influence both the search process and results.[44] Algorithms that can efficiently navigate the design space are very desired, especially for polymer design which involves massive possibilities, making the exhaustive testing not practical.[43] In the following sections, we will explain how researchers define the design space for specific problems and discuss applications of data-driven algorithms in inverse polymer design.

3. Representations and Fingerprints of Polymers

The prerequisite for inverse design of polymers is to numerically represent the polymers to be read and processed by computers. These fingerprints or called descriptors should possess adequate chemo-structural information of the materials while satisfying computational rules with as small size as possible.[45] Since the total energy of a molecule is constant with rotations, translations, and symmetry operations such as mirror reflections of a molecule in a 3D space, a valid representation should be invariant to these operations. When chosen appropriately, representations can accurately correlate structures to properties.[27]

Application of the representations developed for molecules to polymer or macromolecular systems is not straightforward because of the chemical, topological, and morphological complexities of the polymers.[41] In two recently published works, Lengeling and Guzik[6] and Elton et al.[36] reviewed various molecular representations that can be used. Dong et al. created a freely available web-based platform, called ChemDes, to integrate multiple state-of-the-art packages (i.e., Pybel,[46] CDK,[47] RDKit,[48] BlueDesc,[49] Chemopy,[50] PaDEL,[51] and jCompoundMapper[52]) for computing molecular descriptors and fingerprints.[53] ChemDes provides a friendly web interface to relieve users from tedious programming work as well as offering three useful tools for format converting, MOPAC optimization, and fingerprint similarity calculation.[53] Molecular Orbital PACkage (MOPAC) is a program of implementing semi-empirical quantum chemistry computation. MOPAC is mostly used with a graphical user interface.[54] When 3D molecular descriptors are used in the calculations, MOPAC can optimize the chemical structures to obtain relaxed 3D coordinates.[53] In a study of ML-assisted design of high-performance organic photovoltaic materials, Sun et al. employed ChemDes to extract various descriptors and fingerprints for their ML models to identify the best choice of representation.[24] The need for this kind of integrated web-based platform for polymers descriptor and fingerprint computation is much needed.

This review focuses on representations that are specific to polymers and macromolecules. They have been used as input for DL models in inverse design and virtual high-throughput screening tasks. As emphasized by Chen et al, designing polymers fingerprints that convey both chemical and morphological information, as well as their synthesis information, is an open challenge.[40] With the fast development of new chemistry, materials informatics, and data-driven algorithms, a universally applicable polymer representation system is becoming urgent.[45]

3.1. String-based representations from 2D graphs

A system of molecules with atoms and bonds can be considered as graphs with edges and vertices.[36] Obviously, such graphs cannot transfer information about 3D conformations and bond angles and lengths. However, for most of the properties of the structures, such 3D information is not needed. Thus, most generative models have not employed 3D coordinates but instead worked with 2D graphs. After a polymer structure is designed, the most energetically favorable conformation can be extracted using classical forcefields or quantum mechanical approaches.[36] There are several string-based methods to represent graphs for ML/DL-based models that will be reviewed in this review.

Simplified molecular-input line-entry system (*SMILES)*[55] is widely used to represent molecules and polymers.[45, 56-58] After representing atoms and bonds by SMILES symbols, one needs to represent raw characters as one hot encode matrices to perform computation. The first step for that transformation is tokenization from natural language, dividing the whole string into characters. The second step is to use one-hot encoding to represent each character. After deciding the dataset, one needs to extract a pool of unique characters that are present in SMILES sequences, and then assign a numerical value to each character within a sequence. To make the SMILES representations compatible with ML models, one needs to encode the assigned values to one-hot vectors, although the one-hot encoded vectors are larger and increase the computational cost.[24] As an example, if we assign 5 to "C" representing carbon and 6 to "O" representing oxygen, a machine learning model needs to assign a natural ordering between the characters. However, in case of the SMILES representations, there is no ordinal relationship between the characters, making one-hot encoding easier. Technically, all strings should be represented by the same length in ML models. For that, researchers add special characters at the end of the stings to have the same size for all the inputs.[24] Atom and bond matrices can be extracted from SMILES representations.[59] An atom matrix represents the atoms with their atomic numbers and can be one-hot encoded. A bond matrix is usually a 4th order tensor showing information of structures with no bond, single, double, or triple bonds between atoms. These matrices are sometimes named the adjacency matrices and contain the same information as represented by SMILES.

SMILES can be extended to polymers by representing the repeat units of polymers and specifying the connecting points of those repeat units.[21, 26] The transition from molecules to polymers representations can be challenging due largely to increased complexity. For degree-1 polymers (i.e., monomers), the regular SMILES representation can be used with small modifications. Unlike common SMILES strings for small molecules, these degree-1 polymer-SMILES strings contain distinct symbols of "*" to indicate the polymerization points of monomers, which is used for wildcard atom in molecule representation.[60] For relatively simple polymers such as linear chain polymers with two connecting points or ladder polymers with four connecting points in each repeat unit, Tran et al. used *SMILES* to represent these two groups of polymers.[21]

The major challenge in using *SMILES* for DL-based inverse design algorithms is that a large fraction of string combinations does not correspond to valid representations. Invalidity can be syntactic or semantic. In molecule representations, Guzik and colleagues represented a modified version of SMILES with a 100% validity, a representation named *SELFIES.*[61] Employing derivation rules, SELFIES uses different characters from the ones that are used in SMILES to show chains and branches in molecules. The derivation of a single symbol depends on the state of the derivation. They tried SELFIES in the molecule inverse design models.[6, 62] All the generated SELFIES were valid. One sample molecule is shown in both SMILES and SELFIES in Fig. 2. Thiede et al. employed SELFIES representation in their curiosity algorithm powered by deep reinforcement learning for efficient exploration of chemical space to find new molecules.[63] Utilizing a predictor inside their framework, they use the error of the prediction to reward the generator to explore more unknown candidates.

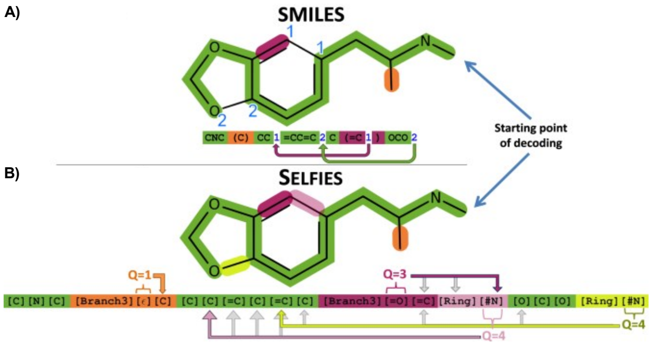
A)
SMILES
1
1
2
2
O
O
N
CNC (C) CC 1 =CC=C 2 C (=C 1) OCO 2
Starting point of decoding
B)
SELFIES
O
O
N
Q=1
Q=3
[C] [N] [C] [Branch3] [ϵ] [C] [C] [C] [=C] [C] [=C] [C] [Branch3] [=O] [=C] [Ring] [#N] [O] [C] [O] [Ring] [#N]
Q=4
Q=4

Fig. 2 String-based representation of a molecular graph. A small organic molecule 3,4-Methyleintenedioxymethamphetamine is used as an example. (**A**) SMILES representation. The main line of atoms in green is completed with branches (opening and closing brackets) and rings (unique numbers after the atoms that are connected). If there is an open parenthesis without closing or only one number for a ring, that would be an invalid structure. (**B**) SELFIES representation. A set of rules that restrict any of the strings from avoiding chemical rules were used (refer to the original paper for details). Reproduced from Ref.[61] published under the terms of Creative Commons Attribution 4.0 license.

Proposed by O'Boyle and Dalke, DeepSMILES is another modification of SMILES in a way to improve the validity of the generated strings. Unlike SELFIES, DeepSMILES does not provide 100% validity, but it improves a higher validity than original SMILES.[64] There is an opportunity for future studies on string-based polymer representations that are valid for any combinations.

Ramprasad and co-workers employed modified SMILES for polymers, in which endpoints or connection points of repeat units were represented using special symbols.[21, 26] As shown in Fig. 3, they used [*] to represent connecting points between the repeat units.[21] Polymer chain, repeat unit, and SMILES of two polymers from linear and ladder groups are shown in Fig. 3. Although low-level representations such as SMILES can depict explicit polymer structures, the strings have large lengths and hard to parse. To represent polyurethane with a chain of length 30 for example, one needs 600 characters that are computationally expensive.[65] Thus, low-level SMILES-based representation is not suitable for large polymers.[65]

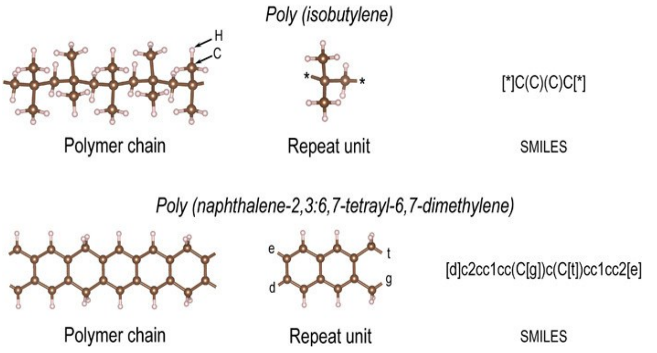

Fig. 3 Polymer chains, repeat units, and SMILES representations of linear polymer poly(isobutylene) and a ladder polymer poly(naphthalene-2,3:6,7-tetrayl-6,7-dimethylene). The connection points are shown with "*". Reproduced from Ref.[21] with the permission from AIP publishing.

Trying to modify the SMILES to fit polymers, Lin et al. introduced BigSMILES as a compact yet structurally robust identifier or a representation system.[58] As shown in Fig. 4, BigSMILES can be used for different organic materials, including homopolymers, random copolymers, and block copolymers with various molecular connectivity, from linear and ring polymers to branched polymers.[45] They used two kinds of bonding descriptors. The first type is AA type bonding that can happen between any two bonding moieties. The second type of bonding, AB bonding, like DNA rules, a bonding moiety cannot connect directly to another from the same group but can connect to one from a different conjugate group. This is the situation in monomers polymerized with condensation reactions.[45] Besides using all the strings in SMILES, BigSMILES uses extra strings to handle the stochastic nature of polymers. There are many details about their descriptors, which can be referred in their paper.[45] They proposed a descriptor system to represents many kinds of polymers, but they did not test it for developing ML/DL for materials design. Trying this representation in a DL-based inverse design is an opportunity for future research. However, as this representation approach relies on the predefined fragments extracted from a training dataset, the fragments of a generated structure is limited to the predefined ones. Although no implementation of SELFIES and DeepSMILES in representing polymers is reported, they can be modified in the same way as BigSMILES was modified from SMILES for polymer representations. Unlike low-level representations such as SMILES, high-level approaches such as Big SMILES are suitable for large polymers. However, they are so high-level that they cannot convey explicit information about the complete polymer structures.[65]

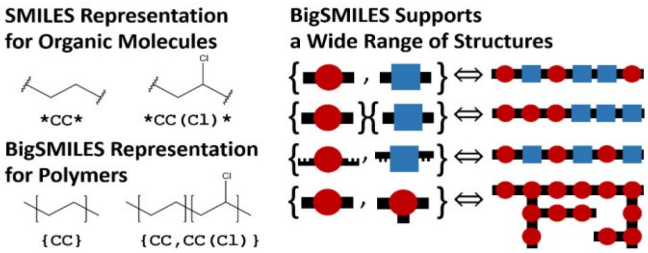
SMILES Representation for Organic Molecules
Cl
CC
CC(Cl)
BigSMILES Representation for Polymers
Cl
{CC}
{CC,CC(Cl)}
BigSMILES Supports a Wide Range of Structures

Fig. 4 Schematic of BigSMILES. Curly brackets separate repeat units that include multiple monomers. Reproduced from Ref.[58], Copyright 2019 American Chemical Society.

Guo et al. recently reported PolyGrammar, a parametric context-sensitive grammar (CSG), to solve limitations of SMILES and BigSMILES for polymer representation.[65] CSG is a formal grammar that defines how to build strings from a language's alphabet obeying a set of production rules (see left side of Fig. 5).[65] PolyGrammar represents a molecular chain structure as a string of symbols, each of which refers to a particular molecular fragment in the polymer chain. The generation process begins with an initial symbol. At each iteration, each non-terminal symbol in the string is replaced by a successor whose predecessor matches the symbol until the string does not have any non-terminal symbols (see Fig. 5, center). The hypergraph is used to translate the resulting symbol string to a polymer chain (see right side of Fig. 5). In an ordinary hypergraph, nodes and edges between the nodes represent atoms and bonds, respectively.[66] The hypergraph allows individual nodes to join any other nodes. An edge that connects a subset of the nodes in the hypergraph is called hyperedge.[67] These production rules make them appropriate to represent many classes of polymers for valid structural generation. In their studies, polyurethane was tested as a proof-of-concept. Nevertheless, further studies are needed to make PolyGrammar generable to generate valid strings of more classes of polymers.

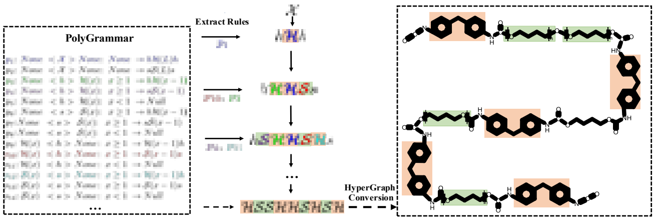

Fig. 5 Schematic of chemistry design model, PolyGrammar. In centre, molecular chain structure as a string of symbols is shown. PolyGrammar has a set of production rules shown on the left. The generation process begins with an initial symbol χ and substitutes each non-terminal symbol (h, s or χ) at each iteration by the successor of a production rule whose predecessor matches the symbol. The process stops when there is no non-terminal symbol. Reproduced from Ref[65] with permission.

All the mentioned string-based representations mainly considered element composition and simplified structures of the polymers. They quite ignore architectures, stochastic nature (PDI), and the processing history of the polymers. These are critical factors in determining their properties. Thermal conductivity, for example, can be significantly different in the same type of a polymer but processed into different forms, such as laminated films or spun fibers due to anisotropic molecular orientation.[68] Wu et al. found that the thermal conductivity significantly depends on the processing history of the polymers. As such information has not been experimentally reported, they failed to derive a predictive model for thermal conductivity directly from the given data. Thus, they considered proxy properties—related to thermal conductivity—such as glass transition temperatures and melting temperatures as the alternative targets.

3.2. 2D/3D information

The Hohenberg-Kohn theorem of DFT proves that the electronic charge density of a system is a universal representation with the total of the information about the system.[69] The material fingerprints can be chemo-structural descriptors or as fundamental as electronic charge density.[28] Using electronic charge density is the most accurate way to represent a system but is not feasible for a large system such as polymers. Pilania et al. conducted a similarity-based machine learning model to extract fingerprints to replace the complicated and cumbersome rule based on Schrödinger's or Kohn-Sham equation.[28]

Using SMILES as input, polymers are either directly fingerprinted by employing hierarchical polymer fingerprints,[21, 26] or represented by molecular fingerprints.[68, 70] Usual kernels extract features of the molecules, hash those features, and utilize the hashed features to determine bits that should be set. Generally, kernels are functions that take two objects (data points, structures) as the input and assign a scalar output value to compare the similarity of the two objects.[71] Typical fingerprint sizes are between 1K to 4K bits. Barnett et al. utilized a Daylight-like fingerprinting algorithm from the RDKit package[48] in their ML-based framework to design exceptional polymer membranes for gas separation.[70] Daylight is a software that delivers a state-of-the-art chemical information processing method. Daylight molecular fingerprints contain a) a pattern representing each atom and its closest neighbors and the bonds that connect them; b) a pattern corresponding to each group of atoms and bonds connected by paths up to seven bonds. Their topology-based approach analyzed the various fragments of a molecule consisting of a certain number of bonds and hashed each fragment to a binary fingerprint.[70] They broke a polymer's repeat unit down into fragments containing between 1 and 7 units and the structure was hashed into a 2048 bits fingerprint to encode all the possible connectivity pathways of the monomer.[70]

Another promising way named hierarchical fingerprints to represent polymers has been introduced by Kim et al. in an ML-model for polymer property prediction.[26] They introduced three levels of descriptors at different length scales (Fig. 6). At the atomic-scale level, the existence of a fixed set of atomic fragments or motifs is tracked. As an example, a triplet of "O1-C3-C4" shows oxygen connects to one atom, a Carbon connected to three atoms, and another Carbon connected to 4 atoms in the same order. They extracted 108 such components from the dataset they used.[26] Next, in a larger level from an RDKit Python library,[48] they used van der Waals surface area,[72] the topological polar surface area (TPSA),[73] the ratio of atoms in rings to the total atoms, and the fraction of rotatable bonds.[26] Each of the mentioned descriptors in QSPR is crucial for accurately predicting properties. For example, TPSA is the sum of surfaces of polar atoms in the molecule that is a key descriptor for T_g and density. Lastly, "morphological descriptor", the highest length-scale descriptor, includes descriptors such as the shortest topological distance between rings, and the length of the largest side-chain.[26] They also considered a recursive feature elimination (RFE) algorithm to remove the least important features. Lightstone et al. utilized this hierarchical fingerprint system to build an ML model for predicting the Refractive index of polymers.[22] This hierarchical fingerprint system can also be used in generative models. Very recently, Kuenneth et al. modified this approach to represent copolymers, an attempt to extend the polymer informatics beyond monopolymer.[74] To do that, first, fingerprints of the repetitive units of a copolymer were extracted. After that, these fingerprints were weighed according to the ratio of the monomers in the copolymer. For instance, C1 and C2 are the ratios of each monomer (unit) in a two-monomer copolymer. If one of the ratios is zero, it indicates a homopolymer[74].

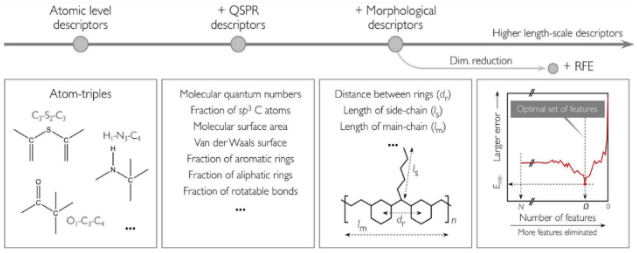

Fig. 6 A hierarchical fingerprint system. This classifies descriptors according to the physical scale and chemical characteristics and RFE process to remove unnecessary features. Reproduced from Ref.[26] with permission. Copyright 2018, American Chemical Society.

In another recently published work, Ramprasad and co-workers introduced a general atomic neighborhood fingerprint method to represent polymers.[75] They incorporated basic components, rotational invariants, and structural features in the representation system. To represent basic components, they employed grid-based representation for the local atomic environment, which includes a hierarchy of features capturing various aspects of the atomic neighborhood (semi-local). To fingerprint rotationally invariant components, they considered some transformation of basic components to make them rotationally invariant to cover cases involving directionless quantities.[75] Finally, they conducted structural fingerprints from predefined components. Based on the application, one can increase the sophistication of the proposed fingerprint to obtain a desired level of accuracy. As an example, Huan et al. investigated the use of just the vector components from basic component category to develop force fields for elemental Al, Cu, C, and more.[76]

After fingerprinting polymers, one can define a suitable measure of chemical distance to quantify the degree of (dis)similarity between two defined fingerprints for developing an ML model with high accuracy, which was demonstrated in Pilania et al.'s work.[28] For example, Kernel Ridge Regression (KRR) is a non-linear regression model that can determine of similarity of input objects.[77] KRR combines ridge regression and classification with kernel machines.[78] The Kernel machines are a class of models originally developed for pattern analysis. They require a user-defined kernel and a similarity function to perform tasks of clustering, rankings, and regression.[79] Using the hierarchical fingerprint system for developing ML-based models for polymer property prediction is quite successful.[21, 26] However, introduction of the fingerprints needs extraction of a pool of components that make the distinguished fragments of polymers. This process requires pre-processing of training datasets. Disadvantage of this method is that one needs to define the pool for each dataset, which make it not generable and cannot be used for generating new polymers consisting of the fragments outside the existing pool.

3.3. Group contribution

A group contribution approach was demonstrated by Van Krevelen and co-workers, where a polymer is broken down into its fragments (groups). From these fragments, the property of the polymer can be predicted.[80] The group contribution methods assume any property is a sum of contributions from building blocks that are independent of each other. This is referred to as quantitative structure-property relationship (QSPR).[80] [10, 27] The group of representations are fast and easy to be interpreted.[27] However, since this approach relies on the available fragment library, for truly novel polymers (outside the predefined library) that are generated by inverse design, group contribution techniques are powerless.[40] Thus, the group contribution methods may not be optimal for new materials discovery but can be useful for feature extraction and property prediction of many polymers.[10] They can be also used to generate low-fidelity data, which although noisy, can be combined with high-fidelity data by multi-fidelity information fusion schemes such as multi-fidelity co-kriging.[81]

By the group contribution techniques, researchers fingerprint the predefined building blocks of polymers.[38] Webb et al. employed a hybrid approach, by which all polymers are constructed from four possible coarse-grained (CG) beads (α, β, δ, and γ). α and β were used to form the backbone of the polymers, while δ and γ were used to form pendant groups that adorn the backbone.[41] They defined 10 different building blocks out of these beads. Within this defined chemical space, they defined three different classes of polymers. Class (I) includes regular polymers with up to four building blocks. Class (II) includes random copolymers with up to four unique building blocks in the polymer sequence. Class (III) is similar to Class (I) but with up to eight building blocks.[41] All the bead types and topologies of polymers are represented in Fig. 7A. They considered three classes of polymers created from these building blocks (Fig. 7B). They then used one-hot encoding (OHE) and property coloring that reflects polymer compositions to extract feature vectors. These vectors were later fed to a deep neural network (DNN) model. To extract property features, the polymer was encoded as an image with each bead of the polymer represented by a pixel (**Fig. 7**C). The coloring of the markers represents the polymer composition. In this way, the application of the data-driven models was extended from homopolymers to copolymers.

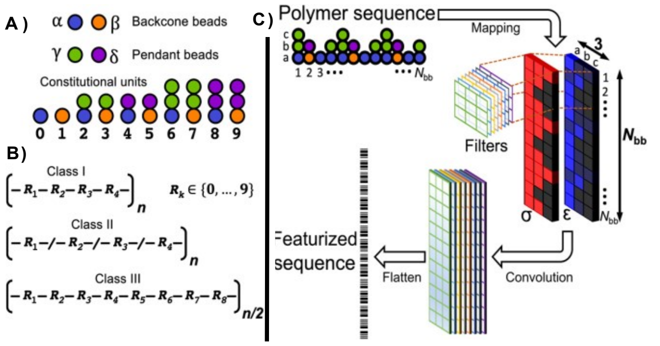

Fig. 7 Schematic of CG polymer presentation and property coloring featurization. (**A**) Bead types and topologies of polymers. α and β are backbone while γ and δ are pendant beads that can form 10 different building blocks (BBs). (**B**) Three classes of polymers. Class I represent regular copolymer with four BBs. Class II shows random polymers with four BBs. Class III are regular polymers with a repeat pattern of eight BBs. (**C**) Filters are used to produce a convolved image that is then flattered to a feature vector. Reproduced from Ref.[41] with permission, Copyright 2020, AAAS.

4. Strategies for Inverse Design of Materials

The traditional materials research paradigm heavily relies on a forward design principle where the properties of materials are predicted from given structures. However, this process is time- and labor-intensive and cannot meet the ever-increasing demands of developing novel materials cost-effectively and speedily. Inverse design, on the other hand, inverts this paradigm *via* receiving desired functionality or properties as inputs for generating the desired structures.[6] This process can be done in two different ways. The first way is called the high throughput virtual screening (HTVS), one of the earliest efforts in inverse design.[7] HTVS can narrow the hypothesized chemical space to find the best candidates possessing targeted properties.[7] The second way includes smart searching algorithms, i.e., global optimization (GO) to navigate the chemical space and DL-based generative models (GMs) to learn hidden knowledge from the training data.

4.1. High throughput virtual screening (HTVS)

By high throughput virtual screening approaches, one needs to narrow the chemical space by defining specific building blocks and bonding rules. The model can then make hypothesized candidates, and those candidates can be tested with the help of an ML-based predictor or high-throughput simulation, such as DFT and MD.[82] Here, the user defines the inputs and ensures that any combination of these inputs (fragments or building blocks of polymers) is valid. Although HTVS seems like a version of the direct approach for material design, its core philosophy is different.[7, 13] First, it focuses on the data-driven discovery that includes automation and time-critical performance.[7] Second, HTVS possesses a computational funnel with promising candidates assessed by more expensive methodologies.[7] Feedback between theory and experiment is a crucial ingredient. It is true that the validity of the generated structures by HTVS is higher than that of the ones generated from GM, but the generation is limited to the hypothesized chemical space.[14, 82]

To generate novel polyimides (PIs) with exceptional refractive index (RI), Afzal *et al*. defined 29 building blocks for PIs' core structures.[82] Definition of 29 building blocks (see Fig. 8B) and their bonding rules are shown in Fig. 8A. They initially generated 6.6 billion compounds. To restrict the search among a more manageable number of candidates, they chose only the most promising 100 R_1 and 100 R_2 with high RI values, resulting in 10,000 PI candidates. R1 and R2 are arranged in the polyimide structures (Fig. 8A). The possible molecular building blocks used to create R1 and R2 are represented in Fig. 8B. R1, represented by green shapes, are linkers and can be chosen from 6 possible linkers in the polyimide structure. R2, shown by blue shapes, are moieties and can be chosen from 23 possible hetero-aromatic moieties in the polyimide structures. Also, R in molecular building blocks (in Fig. 8B) defines allowed sites for linking. Finally, they utilized the HTVS approach to screen them for the best candidates with the highest IR.

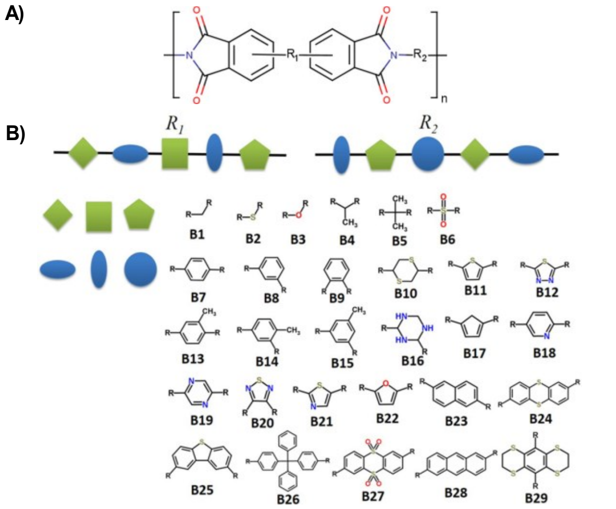

Fig. 8 Genral Polyimide structure and molecular building blocks. (**A**) A Polyimide (PI) core structure with residues R1 and R2. (**B**) Molecular building blocks used for R1 and R2. R in building blocks shows allowed sites for linking. (B1-B6) are linkers marked in green, and blue ones. B7-B29 are hetero-aromatic moieties. Reproduced from Ref.[82] with permission. Copyright 2019, American Chemical Society.

Moreover, we can employ simulation results to provide feedback for chosen candidates. Accordingly, with guidance from a high throughput hierarchal modeling scheme that is involved combinatorial exploration based on DFT followed by successive screening, Treich et al. synthesized novel dielectric materials with high energy density for film capacitors. They considered the organic polymers that were formed by linear combinations of seven basic chemical building blocks.[83]

When experienced chemists have hypotheses that can define a narrowed screening space, they employ HTVS to exploit the space.[84] Manually performing a HTVS is computationally expensive and even impossible for many cases as it requires computational capabilities that allow a large number of calculations to run parallelly.[13] Going beyond the existing hypotheses and broadening the search space need more intelligent approaches. As proposed by Knapp et al., automation is a potential solution.[13] In the next section, we review some advanced algorithms, i.e., GO and GMs, for the inverse design of polymers. They can catch hidden information from a structure-property-paired database for generating novel structures that do not exist in the database.

4.2. Global optimization (GO)

GO, including but not limited to Bayesian optimization (BO), particle swarm optimization (PSO), and genetic algorithm (GA), finds optimal solution of the target objective function and can be employed in the inverse design of polymers.[84] Multi-objective optimization needs a fitness function to consider how the global objective is created by the individual objectives. The evaluation of polymer candidates to check whether they meet the desired property objectives, i.e., computation of fitness function, is a crucial component of GO-based algorithms.[85] One consideration when defining a fitness function is to normalize the objectives to minimize their differences.

4.2.1. Bayesian optimization (BO)

Bayesian optimization (BO) is a sequential design strategy without assumption of any functional forms. Many material tasks can be considered as the optimization problems where controllable parameters must be updated to reach desired objectives. A proper optimization algorithm should be noise-tolerant, global, and convergent with as few inputs as possible. Satisfying these requirements, BO is a systematic approach to find a global optimum of an unknown function f which is expensive to be evaluated.86 87-89

BO is constructed by Bayes' theorem where a joint distribution can be decomposed hierarchically into product of conditional and marginal distributions in the following formula:

$$P_{posterior}(S|Y \in U) \propto P_{likelihood}(Y \in U|S) P_{prior}(S) \qquad (1)$$

Where $P_{posterior}(S|Y)$ is the posterior probability of a model, hypothesis, or theory *S* given input data (observations) *Y*. It is proportional to the likelihood of *Y* given *S* multiplied by the prior probability of *S*.[90] When specifically applied to the polymer design, *S* can be a polymer structure for which the polymeric properties *Y* lie in a desired region *U*.[68] With a desired region *U* given *Y*, it affords $P_{likelihood}(Y \in U|S)$, the probability that defines goodness of fit of *S* with respect to the property requirement. $P_{prior}(S)$ can be used to reduce the occurrence of chemically unfavorable or unrealistic structures and then assign lower probability to them.

Wang et al. proposed an ML-assisted coarse-grained molecular dynamic (CGMD) model to design highly conductive polymer electrolytes.[88] They created a continuous high-dimensional design space from a discrete chemical space by coarse-graining the chemical species (Step 1 and Step 2 shown in Fig. 9). They then employed a BO algorithm to efficiently explore this space *via* autonomous CGMD simulations to predict the relationships between the transport properties and the associated CG parameters (Step 2 and Step 3 shown in Fig. 9). The constructed design space and the corresponding material properties served as the input and output of the model, respectively. They then employed a BO algorithm to efficiently explore this space *via* autonomous CGMD simulations to predict the relationships between the transport properties and the associated CG parameters (from 2 to 3 in Fig. 9). The constructed design space is input, and the target material property is the output of the model.

The procedure of running the BO algorithm includes the following steps: (1) select a prior for the possible space of function *f*; (2) estimate the posterior given the prior and current simulation data; (3) employ the posterior to decide the next calculation to evaluate according to an acquisition function; (4) obtain the new data from the simulation. They iterated 2-4 steps to explore the CG design space until convergence.

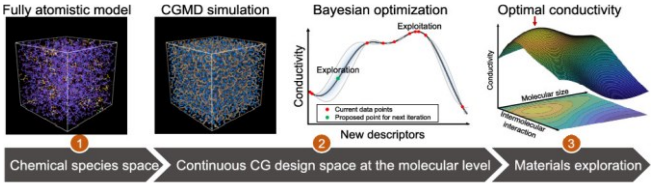

Fig. 9 Illustration of a CGMD-BO framework. A coarse-graining process transforms the chemical space to a continuous space composed of CG parameters (from 1 to 2). BO algorithm explores the space to predict the properties with given CG parameters (from 2 to 3). Reproduced form Ref.[88] Copyright 2020, American Chemical Society.

Accessing large high-quality data in polymer research is still a big challenge, sometimes making it difficult to simply use just one GO for inverse polymer design. To tackle this challenge, Wu et al. employed a combination of BO and a sequential Monte Carlo (SMC) method for the discovery of polymers with high thermal conductivity.[68] Their model creates a chemical space S (encoded by SMILES symbols) consisting of polymer repeat units (monomers), for which nth polymeric properties $\boldsymbol{Y} = (Y_1, \dots, Y_n)$ lie in a desired region U. They then employed Bayes' law to invert the forward model ($S \rightarrow \boldsymbol{Y}$) to obtain a backward model $p(S|\boldsymbol{Y} \in U)(\boldsymbol{Y} \rightarrow S)$. They. then used a sequential Monte Carlo (SMC) method to draw random samples represented by the SMILES strings (S) from high-probability regions of the backward model. Since the experimental thermal conductivity data was limited, when constructing the BO model, they considered proxy properties of glass temperature (T_g) and melting temperature (T_m) which are in correlation with the thermal conductivity as the alternative targets. In addition, they use extended connectivity fingerprints of the SMILES as the input of their prediction model. They designed the monomers but with smaller training datasets compared to other molecular generative models using standard SMILES representation.[31, 35, 91]

4.2.2. Particle swarm optimization (PSO)

In PSO, a bunch of optimizers (particles or agents) moves in a D-dimensional search space. Each agent is composed of four vectors, namely position, velocity, the best position found by itself based on the objective function, and the best position found by its neighbors.

Multiblock polymers are a class of soft materials with spontaneous self-assembly into a variety of ordered mesophases at the nanoscale.[92] Khadilkar *et al*. employed PSO as a global optimizer combined with a forward prediction engine to the inverse design of polymers that have target bulk morphologies.[92] The relevant variables are the polymer architecture parameters, namely chain block fractions, blend fractions, and interaction strength. They employed PSO in multicomponent search spaces. They used PSO for homopolymers and diblock copolymers. The 4-dimensional search space is restricted to only the block fraction of the diblocks. One can refer to their paper for the details on the optimization approach and parameter selection. One way to broaden the use of PSO is by directly targeting properties instead of through structures that were conducted in their research. Kumar et al. conducted high-accuracy tunning of poly(2-oxazoline) cloud point via machine learning techniques. They defined a design space of four repeating units and a range of molecular masses. [93] They performed inverse design via PSO with design selection using a group of neural networks, designing, and synthesizing 17 polymers at 4 target cloud points from 37 to 80 °C.

4.2.3. Genetic algorithm (GA)

Genetic algorithm (GA) is an evolution-based search algorithm that can tackle the problem of inverse polymer design. It uses the idea of natural selection with steps of crossover, mutation, and selection. GA is a type of evolutionary algorithm that mimics the “survival of the fittest” to design or optimize a desired structure with target properties.[94] Meenakshisundaram *et al*. conducted a GA to design sequence-specific copolymers from data generated by molecular dynamic (MD) simulations.[94] The copolymers consist of 20 repetitive units of two types of monomers, which are represented by 0 and 1 binary numbers. The GA determined the fitness of each candidate by analyzing the results calculated from the MD simulations.

Kim *et al*. combined GA with ML-based predictive models to design polymers possessing useful property criteria.[85] To do that, first, they used hierarchical polymer fingerprinting (explained in the representation part) to represent the polymers followed by a Gaussian process regression to map the structures to properties.[87] They then use GA to evolve generations of polymer candidates toward targeted objectives. To design polymers with target properties of glass transition temperature (T_g) of > 500 K and bandgap (E_g) of > 6 ev, T_g and E_g are included in the fitness function. Later, the ML-based predictive models can check the candidates from this fitness function. The GA process follows three steps.

1. Beginning with a randomly generated polymer candidates, they used crossover and mutation to produce new polymer candidates by changing the chemical building blocks and their sequence (Fig. 10A). They extracted 3,045 building blocks with 1 to 4 endpoints from ~ 12,000 reference polymers (Fig. 10B). Endpoints represented by “*” act as a connection between chemical building blocks.[85] For example, one homopolymer has a monomer with two endpoints. They initiated 100 polymers consisting of 8 building blocks in their repeat units. During crossover, offspring were generated from two parent polymers with one random segment. The mutation was also utilized to diversify the “gene pool”. During the evolution, offspring polymers that do not follow chemical rules or polymer assembling rules were removed.
2. The ML models were used to predict the properties of the generated candidates and evaluate their fitness outcome from the proposed fitness function.
3. The best candidates as parent polymers in each generation were kept for the next-iteration evolution.

The mentioned steps were iterated until enough polymer candidates with desired properties were generated. They used two properties T_g and E_g for evaluation purposes (shown in Fig. 10C).

GA starts with a randomly generated initial population with no prior knowledge, while they can improve the generated candidates with the feedback from ML-based prediction models.[85] Obviously, the prediction models need labeled data to learn how to map the structures to specific properties. To accelerate the optimizations and evolutions, one can bias the initial population towards the favorable building blocks with the assistance of prior knowledge to narrow the searching space.[85] Although GAs are general-purpose, stochastic, evolutionary search and optimize strategies, there is no guarantee of their convergence.[95] Moreover, their performance depends on the internal parameters that need trial and error to be tuned.[96]

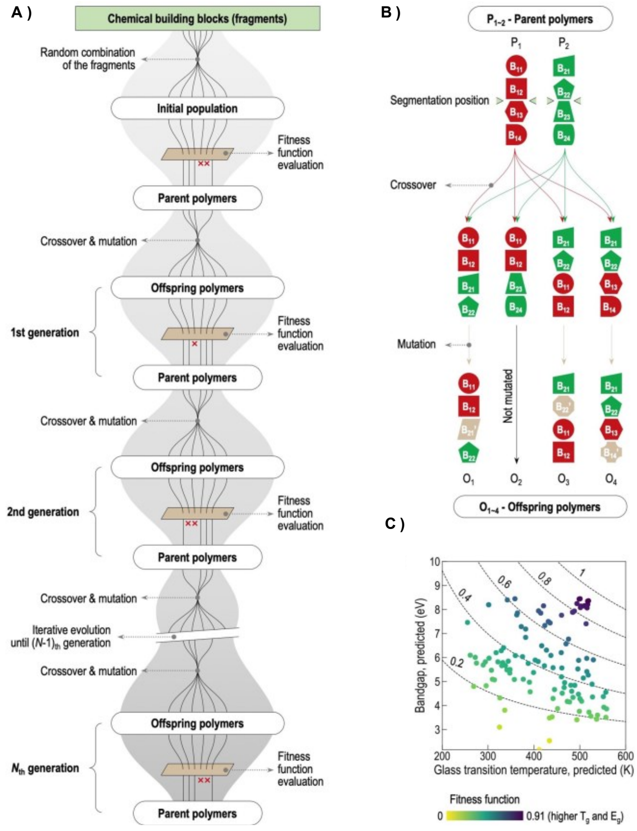

Fig. 10 A GA framework for polymer design. (**A**) Iterative evolution of polymer generation (**B**) demonstration of polymers with four chemical building blocks (fragments) through crossover and mutation. (**C**) improvement of generated polymers possessing higher combination of E_g and T_g. 10 of the best offspring polymers kept as parents for the next iteration. Reproduced from Ref.[85] with permission. Copyright 2020, Elsevier.

4.3. Generative models (GMs)

Recent advances in ML have introduced powerful probabilistic generative models (GMs) capable of generating realistic synthetic samples after being trained on real samples.[6] From a statistical point of view, with an observable variable *X* and a target variable *Y*, a GM estimates a joint probability distribution of *X* and *Y*, *P(X, Y)*. *P(X, Y)* can later be used to generate new data similar to the existing data.[97] GMs can encode the high-dimensional chemical space into the continuous latent space with a lower dimensionality, from which the new data is generated.[6] In this section, we summarize the state-of-the-art deep learning approaches that have been used for inversely designing polymers with targeted properties. Fig. 11 represents schemes of four DL-based GMs, namely recurrent neural networks (RNNs), variational autoencoder (VAE), reinforcement learning (RL), and generative adversarial networks (GAN).

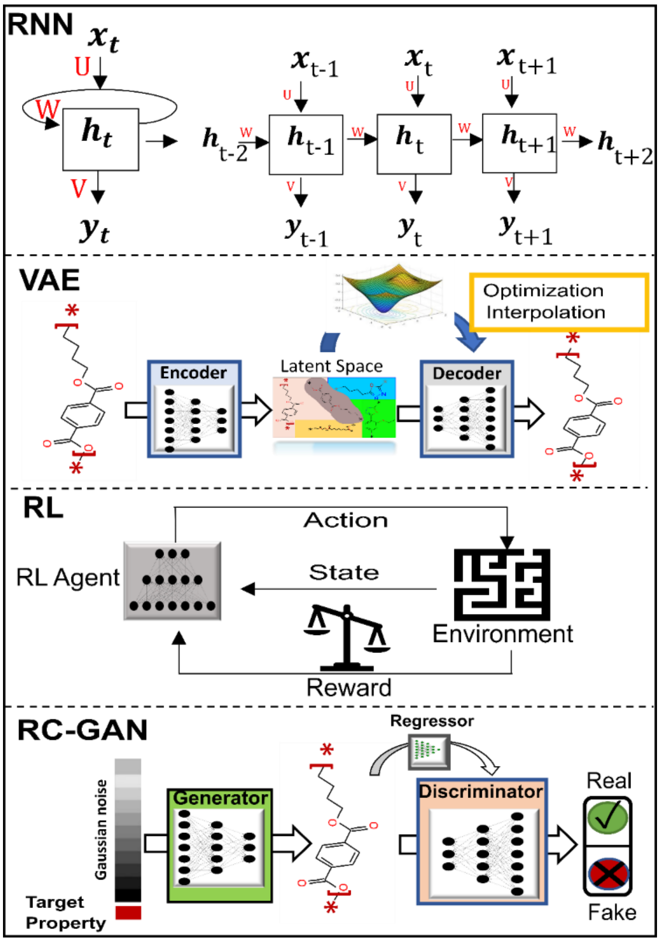

Fig. 11 DL-based algorithms for GMs. From top to bottom: Recurrent Neural Network (RNN), Variational Autoencoder (VAE), Reinforcement Learning (RL), and Generative Adversarial Network (GAN).

4.3.1. Recurrent Neural Network (RNN)

Recurrent neural network (RNN) is designed to predict the future event based on the current and past information, as shown in Fig. 11.[98] Unlike other feed-forward networks that need static input data, RNN can handle arbitrary input sequences.[99] The current input vector, $x_{(t)}$, and the past knowledge, $h_{(t-1)}$, are concatenated to a complete input vector at the time step *t*. Learning the information from the previous iterations makes RNN suitable for generating sequential data, where the information about the future is highly conditioned on the past information and current input.[60 100, 101] RNNs have been widely and successfully employed in molecular drug design.[100, 102-105]

One challenge of applying RNNs to the polymer design is the large size of the polymer sequence. Polymers have long, complex structures. For a generative model, it should enable capturing the long-term temporal dependencies during the generation procedure. RNNs can remember previous information, such as previous characters if polymer chains are represented by SMILES, to learn dynamic behavior for the future generation steps. The original vanilla RNNs (Fig. 10), however, suffer from issues of vanishing and exploding gradients, limiting their ability in learning long-term temporal dependencies.[106] The gradients include information used to update the parameters of the RNNs. Vanishing gradients happen when the updates are insignificant, resulting in no real learning. Exploding gradients, on the other hand, happen when the updated parameters are too large, making the model unstable.

By applying a gradient clipping technique, one can limit the magnitude of gradients to prevent exploding gradients, while the vanishing gradients can be addressed by several gating mechanisms.[106] These mechanisms are implemented in two well-known variants of RNNs: long short-term memory (LSTM)[107] and a gated recurrent unit (GRU)[103].[102] An LSTM network has three gates to regulate the flow of information, namely forget gate, input gate, and output gate.[107] Given the new information, the forget gate decides what information the cell state should forget. The input gate determines the newly encoded information from the new inputs. Finally, the output controls what information should be sent to the next step.[107] The cell state derivative prevents the LSTM gradients from being vanished. GRU has a similar mechanism as the LSTM but with only two gates: the update gate and the reset gate.[99] These two gates decide which hidden state information should be updated. In both LSTM and GRU, the networks learn to skip irrelevant temporary information. Cheng et al. provided in-depth discussion of LSTM and GRU by empirically comparing their performance.[103]

LSTM and GRU have been used to predict protein functions with given sequences as well as the aqueous solubility of drug-like compounds.[106] Popova et al. employed a Stack-RNN with a newly defined cell structure added to the regular GRU cell to learn long-term interdependencies with a target of designing new molecules.[108] With the development of LSTM and GRU, RNNs have shown increased power for polymer design. Ma and Luo employed an RNN for the generation of 1-degree polymers (i.e., monomers) using SMILES representations.[60] As shown in Fig. 12, the future output (o-cell) is the result of the hidden state (h-state) using the previous step (memory about the past) and the current step (present input).[60] They repeat the loop for many iterations, and the performance of RNN in each iteration is assessed by the ratio of the valid samples. However, their work has two limitations. First, it can only be used for generating simple polymers (i.e., monomers). Second, their generation process is not considered inverse design since they did not target any property in advance.

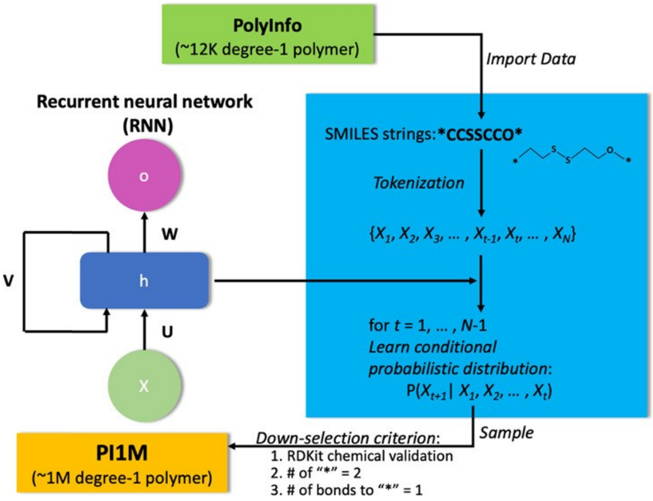

Fig. 12 An RNN architecture for the generation of homopolymers. In an RNN, O-cell generates future output, while h-cell (hidden state) is memory about the past, and X-cell is present input, where U, V, and W are parameters. Reproduced from Ref.[60] with permission, Copyright 2020, American Chemical Society.

4.3.2. Variational Autoencoder (VAE)

A variational autoencoder (VAE) proposed by Kingma *et al.*[109] employs a variational inference framework to estimate the input data distribution $p(x)$ and can be trained with gradient-based methods.[98] It uses an encoder-decoder architecture to reconstruct the input features (or material representations) x and the output $\hat{x}$ in a two-step process (Fig. 11).[6] The encoder constructs a continuous vector in the latent space from the input features, while the decoder converts these continuous vectors back to the input features. A continuous representation allows better usage of powerful gradient-based optimization models to decode random vectors and interpolate structures. Then novel and valid chemical structures can be generated by simple operations in the latent space, such as interpolating between the sampled random vectors of the chemical structures.[6] Furthermore, a continuous representation allows the usage of powerful gradient-based optimization approaches to decode random vectors and interpolate structures more smartly.[6] Bombarelli et al. employed the VAE framework to ensure that samples in the latent space correspond to valid and novel molecular structures.[6]

VAEs can be utilized for the inverse design of materials as they bridge the gap between neural networks and probability models for a large and complicated dataset.[106] Jørgensen et al. proposed a grammar variational autoencoder (GrammarVAE) for inverse design of a class of donor-acceptor polymers.[110] They used SMILES representations combined with grammar rules to increase the validity of the generated SMILES. The grammar rules are changed by the decoder so that it can only generate syntactically valid strings.

Batra et al. utilized a syntax-directed VAE combined with Gaussian process regression (GPR) predictive models to discover polymers with targeted properties. In this work, they introduced crucial modifications in SMILES grammar and polymer-specific semantics to increase the validity of the generated structures.[111] To do that, they first converted the SMILES strings to parse trees. They then utilized context-free-grammar parse trees as input for the encoder to convert them to continuous latent vectors. The derived latent vectors containing chemical and structural information help to build accurate predictive models for property predictions. To design innovative polymers possessing targeted properties, they employed simple enumeration followed by a generative interpolation approach.

4.3.3. Reinforcement learning (RL)

Reinforcement learning (RL), designed to tackle dynamic decision challenges,[108] includes analysis of possible actions and approximation of the statistical relationship between the actions and possible outcomes. They are reinforced by the determination of a treatment regime that is optimized towards the most desirable outcomes.[112] Very recently, RL achieved better performance than humans in the game of Go,[113] which has the complexity of 10^{140} possibilities.[114] It is analogous to the complexity of chemical space, which makes RL-based networks suitable to be applied to the inverse design of materials.[108]

As an example of the most successful works in RL for materials design,[63, 115, 116] Popova *et al*. proposed a deep RL (DRL) for generating chemical compounds with desired physical, chemical, and activity properties (see **Fig. 13**).[108] They combined two deep neural networks (a generative model (G) and a predictive model (P)) in the DRL framework. Playing the role of an agent, G generates novel molecules. Playing the role of a critic, P outputs the properties of the novel structures and assigns a numerical reward/penalty to the candidates. G learns to maximize the reward by improving the generated structures with properties close to desired ones.

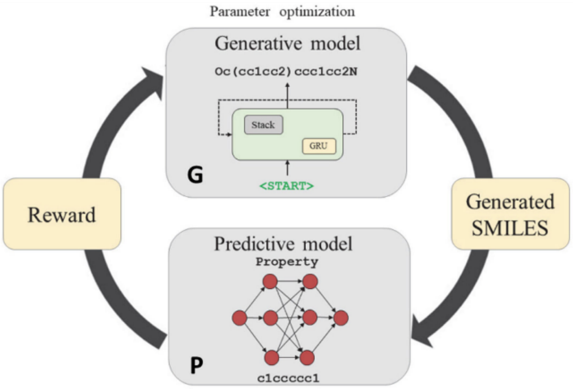

Fig. 13 A workflow of an RL algorithm for a compound generation. Reproduced with permission from Ref.[108], AAAS.

4.3.4. Generative adversarial networks

A generative adversarial network (GAN) includes two competing networks of a generator and a discriminator.[117] The generator generates sample data from random noise, while the discriminator examines the data to judge whether it is synthesized (fake) or sampled from the training dataset (real).[117] Competition of the generator and the discriminator improves both networks in such a way that the generator can generate so real data that the discriminator cannot distinguish them.[98] GANs are well known for their ability to learn complex high dimensional data and reproduce them by following similar distributions.[62] Among various DL algorithms, GANs bring in a breakthrough for materials discovery.[118] GANs can utilize different architectures such as CNNs,[34] AEs, and RNNs to implement the algorithms.[62] Meanwhile, GANs also suffer from a serious issue of mode collapse. Among various solutions, minibatch discrimination and feature mapping have been introduced to solve this issue.[119] Another way to avoid mode collapse is to penalize the model if it generates repetitive (non-unique) sequences.[62] Although fully-connected networks have been used for the original GAN model,[117] recent studies have utilized different architectures such as CNNs,[34] AEs, and RNNs.[62]

To enable on-demand data generation, the unsupervised GAN model can be modified by adding labeled information as the input condition, which is named the conditional GAN (CGAN).[120] Following CGAN, auxiliary classifier GAN (ACGAN) adopted discrete and qualitative labels in the objective function for training the ACGAN, which makes the model suitable for discrete and qualitative labels.[121] Improving ACGAN, a semi-supervised reg-GAN was developed for generating images from quantitative labels. However, the reg-GAN distinguishes the synthesized data from the real data by predicting the label first, then compares the difference between the predicted and the desired ones. To do that, a pre-set range of numbers is needed, which requires human intervention. Since their birth, GANs have transformed various fields ranging from image, speech, to materials science.[122] Nevertheless, these aforementioned GANs do not meet the criteria for generating material structures with explicitly given properties (represented by continuous labels) due to the lack of a mechanism of generating data in a regressional and conditional manner. In a study proposed by Dong et al,[34] to overcome the limitations in the previous GANs, they demonstrated a regressional and conditional GAN (RCGAN), which meets two criteria for inverse design of materials: 1) it generates distinguished structures from the real structures used for training; 2) it can accurately perform a generation task based on input quantitative labels. RCGAN can be potentially used for inversely designing molecules and polymers. As RCGAN uses a convolutional neural network (CNN) architecture, the generator generates all structures at once. But in an RNN architecture that has been employed in most GANs for the molecular inverse design, the generator generates a single character of a SMILES string at once. CNN-based GANs are more suitable for bigger systems such as polymers. Although RNN-based models may generate structures with higher validity, they are much more expensive for computing polymeric systems.

4.3.5. Hybrid architectures

Some hybrid architectures that combine GANs with other algorithms, e.g., RL, to tackle the challenge of inverse design of polymers have been proposed. Although GANs have been widely employed in drug and molecule inverses design, their application in polymers design faces grand obstacles.[26] First, even with a properly defined polymer representation, the input data is larger and more computationally expensive than that of molecules. Second, one needs to consider the polymer architecture that defines the way of branching or networking of the polymer chains.[123] With a longer sequence of data, one needs to modify the architecture of a generator to handle this challenge.[95, 96] For a GAN model, for example, it is more difficult for the generator to mimic the real data in a way that the discriminator cannot distinguish them from the real structures.[98] RLs, on the other hand, can be used to tune the properties of the generated samples toward desired values. Researchers combined various GANs structures with RL components in a way to direct the generator to generate molecules with targeted properties (see ORGANIC framework in Fig. 14).[35, 62, 124] The RL components add a reward to the discriminator to bias the employed RNN generator to create structures with a single or a set of target properties. The focus of this kind of hybrid model (combination of GANs and RL) is to generate a bunch of samples that follow a targeted range of properties (a proper distribution). So far, mentioned hybrid models were conducted for molecule design. It is envisioned that such hybrid architectures will emerge for inverse polymer design.

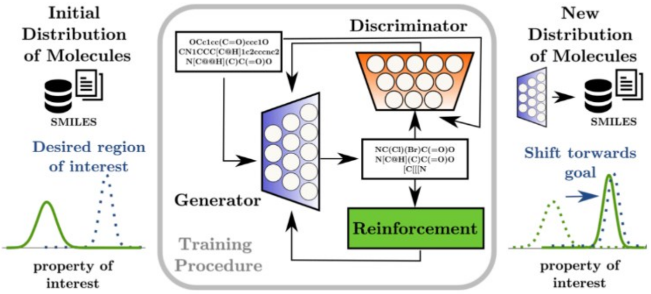

Fig. 14 Schematic of hybrid architecture of ORGANIC, with three fundamental components: a generator, a discriminator, and a reinforcement metric. Reproduced from Ref.[35]

5. Conclusion

Within this review, we have systematically surveyed the recent progress on the inverse design of polymers. First, the prerequisite, i.e., numerical representations of polymers that save as much as structural and topological information, was summarized. Then, three mainstream data-driven algorithms including HTVS, GO, and GMs for inverse design were outlined and their advantages and disadvantages were discussed. Although the inverse design has been advanced in the past decade, many challenges remain to be addressed. Two main ones as follows are considered as the most interesting and pressing.

5.1. From homopolymers to complex polymers

Polymer informatics tools have been recently growing for efficiently designing new polymers possessing targeted properties. However, as we discussed in the previous sections, most of the data-driven algorithms focus on molecules or homopolymers.[74] With simple modifications, molecular representations, such as SMILES, can be used to represent homopolymers.[105, 125, 126] However, for more complex polymers such as copolymers, polymer blends, and polymers with additives, the simple extension may not be applicable.[40] Very recently, Kuenneth et al. attempted to address the issue by developing new representations for predicting properties of copolymers, which opens a new route to developing state-of-art deep learning algorithms for copolymers design.

Most of the computational data for polymers are based on DFT calculations of their monomers or small oligomeric species.[4, 14] Polymers as macromolecules, however, contain more structural and conformational information. Direct first-principle calculations of the whole macromolecule chains are not possible. Webb et al. proposed a targeted sequence design for copolymers in an attempt to use coarse-grained (CG) classical modeling for data generation.[41] They predefined building blocks and employed feature extraction approaches to build the input representations for their deep learning model, which afforded quite impressive results.

5.2. Architectures of polymers

Defining design space of polymers is critical for polymer design. In most works of inverse polymer design, researchers consider a simplified and restricted design space while ignoring the structural complexity of polymers such as their architectures.[123] Architectural features such as branches, stars, and bottlebrushes of the polymers can largely affect their physical properties, including solubility in different solvents, glass transition temperature. They can be even crucial for some biopolymers such as DNA polymerized from four different monomers. Srinivasan et al. employed a genetic algorithm (GA) to design DNA-grafted particles that self-assemble into desired crystalline structures.[95] The employed GA framework initiates the DNA-grafted particle population for predicting superstructures formed using these building blocks.

5.3. Active learning

One significant challenge of applying data-driven algorithms of inverse materials design is the lack of sufficient high-quality and labeled data. To tackle this challenge, one can employ active learning, a paradigm in which the ML models direct the learning procedure themselves through dynamic suggestions for the next iteration of operation.[127, 128] Kim et al. employed active learning for the discovery of polymers with high glass transition temperatures (T_g). Starting with an initial small dataset of polymers, they use an ML-based predictive model in conjunction with an active-learning framework to iteratively add the new candidates. The active learning model decides the range of exploitation and exploration for selecting the next experiment. In this design, having an accurate predictive model is important. In addition, employing a suitable representation system for the polymers is crucial. Active learning for inverse design of polymers begins with utilizing hybrid GMs, elaborated in previous sections, to generate candidates possessing targeted properties. Then an active learning architecture can be used to provide feedback to guide the model to generate innovative structures with properties outside the range of the training dataset. This can be a method of doing extrapolation.

Author Contributions

J.L. conceived the review subject idea and supervised the process. K.S. performed the literature review and wrote the first draft. Y.X. and J.L. reviewed and corrected the manuscript.

Acknowledgment

This work was financially supported by grants from the National Science Foundation (award numbers: 1825352 and 1933861), the United States Department of Agriculture (award number: 2020-67030-31336), and Sony Research Award Program 2019.

Conflicts of interest

The authors declare no conflicts of interest.

References

1. S. J. Garcia, *Eur. Polym. J.*, 2014, **53**, 118-125. https://www.sciencedirect.com/science/article/pii/S0014305714000366.
2. A. C. Rinkenauer, S. Schubert, A. Traeger and U. S. Schubert, *J. Mater. Chem. B*, 2015, **3**, 7477-7493.
3. D. Paramelle, S. Gorelik, Y. Liu and J. Kumar, *Chem. Commun.*, 2016, **52**, 9897-9900.
4. A. Mannodi-Kanakkithodi, G. Pilania, T. D. Huan, T. Lookman and R. Ramprasad, *Sci. Rep.*, 2016, **6**, 1-10.
5. M. Tamasi, S. Kosuri, J. DiStefano, R. Chapman and A. J. Gormley, *Adv. Intell. Syst.*, 2020, **2**, 1900126.
6. 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.
7. B. Sanchez-Lengeling and A. Aspuru-Guzik, *Science*, 2018, **361**, 360-365.
8. G. Chen, Z. Shen, A. Iyer, U. F. Ghumman, S. Tang, J. Bi, W. Chen and Y. Li, *Polymers*, 2020, **12**, 163.
9. T. E. Gartner and A. Jayaraman, *Macromolecules*, 2019, **52**, 755-786. https://doi.org/10.1021/acs.macromol.8b01836.
10. S. Venkatram, R. Batra, L. Chen, C. Kim, M. Shelton and R. Ramprasad, *J. Phys. Chem. B*, 2020, **124**, 6046-6054.
11. H. D. Tran, C. Kim, L. Chen, A. Chandrasekaran, R. Batra, S. Venkatram, D. Kamal, J. P. Lightstone, R. Gurnani, P. Shetty, M. Ramprasad, J. Laws, M. Shelton and R. Ramprasad, *J. Appl. Phys.*, 2020, **128**, 171104. https://aip.scitation.org/doi/abs/10.1063/5.0023759.
12. H. Deng, C. Zhang, K. Sattari, Y. Ling, J.-W. Su, Z. Yan and J. Lin, *ACS Appl. Mater. Interfaces*, 2020.
13. E. O. Pyzer-Knapp, C. Suh, R. Gómez-Bombarelli, J. Aguilera-Iparraguirre and A. Aspuru-Guzik, *Annu. Rev. Mater. Res.*, 2015, **45**, 195-216.
14. P. C. St. John, C. Phillips, T. W. Kemper, A. N. Wilson, Y. Guan, M. F. Crowley, M. R. Nimlos and R. E. Larsen, *J. Chem. Phys.*, 2019, **150**, 234111.
15. J. Carrete, W. Li, N. Mingo, S. Wang and S. Curtarolo, *Phys. Rev. X*, 2014, **4**, 011019.
16. H. Deng, K. Sattari, Y. Xie, P. Liao, Z. Yan and J. Lin, *Nat. Commun.*, 2020, **11**, 1-10.
17. K. Sattari, M.S., Saint Louis University, 2019.
18. J. Glaser, T. D. Nguyen, J. A. Anderson, P. Lui, F. Spiga, J. A. Millan, D. C. Morse and S. C. Glotzer, *Comput. Phys. Commun.*, 2015, **192**, 97-107.

19. A. M. Virshup, J. Contreras-García, P. Wipf, W. Yang and D. N. Beratan, *J. Am. Chem. Soc.*, 2013, **135**, 7296-7303.

20. L. Chen, C. Kim, R. Batra, J. P. Lightstone, C. Wu, Z. Li, A. A. Deshmukh, Y. Wang, H. D. Tran and P. Vashishta, *npj Comput. Mater.*, 2020, **6**, 1-9.

21. H. Doan Tran, C. Kim, L. Chen, A. Chandrasekaran, R. Batra, S. Venkatram, D. Kamal, J. P. Lightstone, R. Gurnani and P. Shetty, *J. Appl. Phys.*, 2020, **128**, 171104.

22. J. P. Lightstone, L. Chen, C. Kim, R. Batra and R. Ramprasad, *J. Appl. Phys.*, 2020, **127**, 215105.

23. M. Glavatskikh, J. Leguy, G. Hunault, T. Cauchy and B. Da Mota, *J. Cheminformatics*, 2019, **11**, 69.

24. W. Sun, Y. Zheng, K. Yang, Q. Zhang, A. A. Shah, Z. Wu, Y. Sun, L. Feng, D. Chen and Z. Xiao, *Sci. Adv.*, 2019, **5**, eaay4275.

25. A. Jha, A. Chandrasekaran, C. Kim and R. Ramprasad, *Modell. Simul. Mater. Sci. Eng.*, 2019, **27**, 024002.

26. C. Kim, A. Chandrasekaran, T. D. Huan, D. Das and R. Ramprasad, *J. Phys. Chem. C*, 2018, **122**, 17575-17585.

27. K. Wu, N. Sukumar, N. Lanzillo, C. Wang, R. "Rampi" Ramprasad, R. Ma, A. Baldwin, G. Sotzing and C. Breneman, *J. Polym. Sci., Part B: Polym. Phys.*, 2016, **54**, 2082-2091.

28. G. Pilania, C. Wang, X. Jiang, S. Rajasekaran and R. Ramprasad, *Sci. Rep.*, 2013, **3**, 1-6.

29. N. H. Park, D. Y. Zubarev, J. L. Hedrick, V. Kiyek, C. Corbet and S. Lottier, *Macromolecules*, 2020, **53**, 10847-10854.

30. A. L. Ferguson and R. Ranganathan, *ACS Macro Lett.*, 2021, **10**, 327-340.

31. C. Shen, M. Krenn, S. Eppel and A. Aspuru-Guzik, *arXiv:2012.09712*, 2020. https://arxiv.org/abs/2012.09712.

32. A. Zunger, *Nat. Rev. Chem.*, 2018, **2**, 1-16.

33. J. Noh, J. Kim, H. S. Stein, B. Sanchez-Lengeling, J. M. Gregoire, A. Aspuru-Guzik and Y. Jung, *Matter*, 2019, **1**, 1370-1384.

34. Y. Dong, D. Li, C. Zhang, C. Wu, H. Wang, M. Xin, J. Cheng and J. Lin, *Carbon*, 2020.

35. B. Sanchez-Lengeling, C. Outeiral, G. L. Guimaraes and A. Aspuru-Guzik, *ChemRxiv*, 20170. https://chemrxiv.org/engage/chemrxiv/article-details/60c73d91702a9beea7189bc2.

36. D. C. Elton, Z. Boukouvalas, M. D. Fuge and P. W. Chung, *Mol. Syst. Des. Eng.*, 2019, **4**, 828-849.

37. R. Vasudevan, G. Pilania and P. V. Balachandran, *J. Appl. Phys.*, 2021, **129**, 070401. https://aip.scitation.org/doi/abs/10.1063/5.0043300.

38. B. Kim, S. Lee and J. Kim, *Sci. Adv.*, 2020, **6**, eaax9324.

39. Z. M. Sherman, M. P. Howard, B. A. Lindquist, R. B. Jadrich and T. M. Truskett, *J. Chem. Phys.*, 2020, **152**, 140902.

40. L. Chen, G. Pilania, R. Batra, T. D. Huan, C. Kim, C. Kuenneth and R. Ramprasad, *Mater. Sci. Eng. R*, 2021, **144**, 100595.

41. M. A. Webb, N. E. Jackson, P. S. Gil and J. J. de Pablo, *Sci. Adv.*, 2020, **6**, eabc6216.

42. S. Otsuka, I. Kuwajima, J. Hosoya, Y. Xu and M. Yamazaki, *2011 Inter. Conf. on Emer. Intell. Data and Web Tech., IEEE*, 2011, 22-29.

43. C. W. Coley, N. S. Eyke and K. F. Jensen, *Angew. Chem. Int. Ed.*, 2020, **59**, 22858-22893.

44. D. Klahr, A. L. Fay and K. Dunbar, *Cogn. Psychol.*, 1993, **25**, 111-146.

45. T.-S. Lin and B. Olsen, *Bull. Am. Phys. Soc.*, 2020, **65**.

46. N. M. O'Boyle, C. Morley and G. R. Hutchison, *Chem. Cent. J.*, 2008, **2**, 1-7.

47. C. Steinbeck, Y. Han, S. Kuhn, O. Horlacher, E. Luttmann and E. Willighagen, *J. Chem. Inform. Comput. Sci.*, 2003, **43**, 493-500.

48. G. Landrum, RDKit: Open-source cheminformatics, http://www.rdkit.org, (accessed 01-June-2021).

49. G. Hinselmann, BlueDesc - Molecular Descriptor Calculator, http://www.ra.cs.uni-tuebingen.de/software/bluedesc/welcome_e.html, (accessed 01-April-2021).

50. D.-S. Cao, Q.-S. Xu, Q.-N. Hu and Y.-Z. Liang, *Bioinformatics*, 2013, **29**, 1092-1094.

51. C. W. Yap, *J. Comput. Chem.*, 2011, **32**, 1466-1474.

52. G. Hinselmann, L. Rosenbaum, A. Jahn, N. Fechner and A. Zell, *J. Cheminformatics*, 2011, **3**, 1-14.

53. J. Dong, D.-S. Cao, H.-Y. Miao, S. Liu, B.-C. Deng, Y.-H. Yun, N.-N. Wang, A.-P. Lu, W.-B. Zeng and A. F. Chen, *J. Cheminformatics*, 2015, **7**, 1-10.

54. J. J. Stewart, MOPAC2012, http://openmopac.net/, (accessed 01-June-2021).

55. D. Weininger, *J. Chem. Inform. Comput. Sci.*, 1988, **28**, 31-36.

56. E. J. Bjerrum, *arXiv:1703.07076*, 2017. https://arxiv.org/abs/1703.07076.

57. J. Arús-Pous, A. Patronov, E. J. Bjerrum, C. Tyrchan, J.-L. Reymond, H. Chen and O. Engkvist, *J. Cheminformatics*, 2020, **12**, 1-18.

58. T.-S. Lin, C. W. Coley, H. Mochigase, H. K. Beech, W. Wang, Z. Wang, E. Woods, S. L. Craig, J. A. Johnson and J. A. Kalow, *ACS Cent. Sci.*, 2019, **5**, 1523-1531.

59. D. J. Klein, *J. Chem. Inform. Comput. Sci.*, 2002, **42**, 1507-1507.

60. R. Ma and T. Luo, *J. Chem. Inf. Model.*, 2020, **60**, 4684-4690.

61. M. Krenn, F. Häse, A. Nigam, P. Friederich and A. Aspuru-Guzik, *Mach. Learn.: Sci. Tech.*, 2020, **1**, 045024.

62. G. L. Guimaraes, B. Sanchez-Lengeling, C. Outeiral, P. L. C. Farias and A. Aspuru-Guzik, *arXiv:1705.10843*, 2017. https://arxiv.org/abs/1705.10843.

63. L. A. Thiede, M. Krenn, A. Nigam and A. Aspuru-Guzik, *arXiv:2012.11293*, 2020. https://arxiv.org/abs/2012.11293.

64. A. Dalke, *ChemRxiv*, 2018.

65. M. Guo, W. Shou, L. Makatura, T. Erps, M. Foshey and W. Matusik, *arXiv:2105.05278*, 2021. https://arxiv.org/abs/2105.05278.

66. H. Wang, J. Wang, J. Wang, M. Zhao, W. Zhang, F. Zhang, X. Xie and M. Guo, *Proceedings of the AAAI conference on artificial intelligence*, 2018, **32**. https://ojs.aaai.org/index.php/AAAI/article/view/11872.

67. C. Berge, *Hypergraphs: combinatorics of finite sets*, Elsevier, 1984.

68. S. Wu, Y. Kondo, M.-a. Kakimoto, B. Yang, H. Yamada, I. Kuwajima, G. Lambard, K. Hongo, Y. Xu and J. Shiomi, *npj Comput. Mater.*, 2019, **5**, 1-11.

69. P. Hohenberg and W. Kohn, *Physical review*, 1964, **136**, B864.

70. J. W. Barnett, C. R. Bilchak, Y. Wang, B. C. Benicewicz, L. A. Murdock, T. Bereau and S. K. Kumar, *Sci. Adv.*, 2020, **6**, eaaz4301.

71. J. A. Mohr, B. J. Jain and K. Obermayer, *J. Chem. Inf. Model.*, 2008, **48**, 1868-1881.

72. P. Labute, *J. Mol. Graphics Modell.*, 2000, **18**, 464-477.

73. S. Prasanna and R. Doerksen, *Curr. Med. Chem.*, 2009, **16**, 21-41.

74. C. Kuenneth, W. Schertzer and R. Ramprasad, *arXiv:2103.14174*, 2021. https://arxiv.org/abs/2103.14174.

75. R. Batra, H. D. Tran, C. Kim, J. Chapman, L. Chen, A. Chandrasekaran and R. Ramprasad, *J. Phys. Chem. C*, 2019, **123**, 15859-15866.

76. T. D. Huan, R. Batra, J. Chapman, S. Krishnan, L. Chen and R. Ramprasad, *npj Comput. Mater.*, 2017, **3**, 1-8.

77. A. Mannodi-Kanakkithodi, G. Pilania and R. Ramprasad, *Comput. Mater. Sci.*, 2016, **125**, 123-135.

78. K. P. Murphy, *Machine learning: a probabilistic perspective*, MIT press, 2012.

79. T. Hofmann, B. Schölkopf and A. J. Smola, *The annals of statistics*, 2008, 1171-1220.

80. D. W. Van Krevelen and K. Te Nijenhuis, *Properties of polymers: their correlation with chemical structure; their numerical estimation and prediction from additive group contributions*, Elsevier, 2009.

81. A. Patra, R. Batra, A. Chandrasekaran, C. Kim, T. D. Huan and R. Ramprasad, *Comput. Mater. Sci.*, 2020, **172**, 109286.

82. M. A. F. Afzal, M. Haghighatlari, S. P. Ganesh, C. Cheng and J. Hachmann, *J. Phys. Chem. C*, 2019, **123**, 14610-14618.

83. G. M. Treich, M. Tefferi, S. Nasreen, A. Mannodi-Kanakkithodi, Z. Li, R. Ramprasad, G. A. Sotzing and Y. Cao, *IEEE Trans. Dielectr. Electr. Insul.*, 2017, **24**, 732-743.

84. J. Noh, G. H. Gu, S. Kim and Y. Jung, *Chem. Sci.*, 2020, **11**, 4871-4881.

85. C. Kim, R. Batra, L. Chen, H. Tran and R. Ramprasad, *Comput. Mater. Sci.*, 2021, **186**, 110067.

86. J. Mockus, *Bayesian approach to global optimization: theory and applications*, Springer Science & Business Media, 2012.

87. C. K. Williams and C. E. Rasmussen, *Gaussian processes for machine learning*, MIT press Cambridge, MA, 2006.

88. Y. Wang, T. Xie, A. France-Lanord, A. Berkley, J. A. Johnson, Y. Shao-Horn and J. C. Grossman, *Chem. Mater.*, 2020, **32**, 4144-4151.

89. F. Häse, L. M. Roch, C. Kreisbeck and A. Aspuru-Guzik, *ACS Cent. Sci.*, 2018, **4**, 1134-1145.

90. B. Burger, P. M. Maffettone, V. V. Gusev, C. M. Aitchison, Y. Bai, X. Wang, X. Li, B. M. Alston, B. Li and R. Clowes, *Nature*, 2020, **583**, 237-241.

91. K. Kim, S. Kang, J. Yoo, Y. Kwon, Y. Nam, D. Lee, I. Kim, Y.-S. Choi, Y. Jung and S. Kim, *npj Comput. Mater.*, 2018, **4**, 1-7.

92. M. R. Khadilkar, S. Paradiso, K. T. Delaney and G. H. Fredrickson, *Macromolecules*, 2017, **50**, 6702-6709.

93. J. N. Kumar, Q. Li, K. Y. Tang, T. Buonassisi, A. L. Gonzalez-Oyarce and J. Ye, *npj Comput. Mater.*, 2019, **5**, 1-6.

94. V. Meenakshisundaram, J.-H. Hung, T. K. Patra and D. S. Simmons, *Macromolecules*, 2017, **50**, 1155-1166.

95. B. Srinivasan, T. Vo, Y. Zhang, O. Gang, S. Kumar and V. Venkatasubramanian, *PNAS*, 2013, **110**, 18431-18435.

96. T. Vo, V. Venkatasubramanian, S. Kumar, B. Srinivasan, S. Pal, Y. Zhang and O. Gang, *PNAS*, 2015, **112**, 4982-4987.

97. A. Y. Ng and M. I. Jordan, *NeurIPS*, 2002, 841-848.

98. I. Goodfellow, Y. Bengio, A. Courville and Y. Bengio, *Deep learning*, MIT press Cambridge, 2016.

99. F. Gers, Doctoral dissertation, Verlag nicht ermittelbar, 2001.

100. A. Gupta, A. T. Müller, B. J. Huisman, J. A. Fuchs, P. Schneider and G. Schneider, *Molecular informatics*, 2018, **37**, 1700111.

101. A. Lusci, G. Pollastri and P. Baldi, *J. Chem. Inf. Model.*, 2013, **53**, 1563-1575.

102. M. H. Segler, T. Kogej, C. Tyrchan and M. P. Waller, *ACS Cent. Sci.*, 2018, **4**, 120-131.

103. J. Chung, C. Gulcehre, K. Cho and Y. Bengio, *arXiv:1412.3555*, 2014. https://arxiv.org/abs/1412.3555.

104. P.-C. Kotsias, J. Arús-Pous, H. Chen, O. Engkvist, C. Tyrchan and E. J. Bjerrum, *Nat. Mach. Intell.*, 2020, **2**, 254-265.

105. A. L. Nazarova, L. Yang, K. Liu, A. Mishra, R. K. Kalia, K.-i. Nomura, A. Nakano, P. Vashishta and P. Rajak, *J. Chem. Inf. Model.*, 2021, **61**, 2175-2186.

106. B. Rezaeianjouybari and Y. Shang, *Measurement*, 2020, **163**, 107929.

107. S. Hochreiter and J. Schmidhuber, *Neural computation*, 1997, **9**, 1735-1780.

108. M. Popova, O. Isayev and A. Tropsha, *Sci. Adv.*, 2018, **4**, eaap7885.

109. D. P. Kingma and M. Welling, *arXiv:1312.6114*, 2013. https://arxiv.org/abs/1312.6114.

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

111. R. Batra, H. Dai, T. D. Huan, L. Chen, C. Kim, W. R. Gutekunst, L. Song and R. Ramprasad, *Chem. Mater.*, 2020, **32**, 10489-10500.

112. M. Krakovsky, *Commun. ACM*, 2016, **59**, 12-14.

113. D. Silver, A. Huang, C. J. Maddison, A. Guez, L. Sifre, G. Van Den Driessche, J. Schrittwieser, I. Antonoglou, V. Panneershelvam and M. Lanctot, *Nature*, 2016, **529**, 484-489.

114. H. J. Van Den Herik, J. W. Uiterwijk and J. Van Rijswijck, *Artificial Intelligence*, 2002, **134**, 277-311.

115. M. Sarmad, H. J. Lee and Y. M. Kim, *In Proceedings of the IEEE/CVF, CVPR*, 2019, 5898-5907.

116. Z. Zhou, X. Li and R. N. Zare, *ACS Cent. Sci.*, 2017, **3**, 1337-1344.

117. I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville and Y. Bengio, 2014.

118. A. Kadurin, S. Nikolenko, K. Khrabrov, A. Aliper and A. Zhavoronkov, *Mol. Pharm.*, 2017, **14**, 3098-3104.

119. T. Salimans, I. Goodfellow, W. Zaremba, V. Cheung, A. Radford and X. Chen, *arXiv:1606.03498*, 2016. https://arxiv.org/abs/1606.03498.

120. M. Mirza and S. Osindero, *arXiv:1411.1784*, 2014. https://arxiv.org/abs/1411.1784.

121. A. Odena, C. Olah and J. Shlens, 2017.

122. A. Gupta and J. Zou, *Nat. Mach. Intell.*, 2019, **1**, 105-111.

123. M. Rubinstein and R. H. Colby, *Polymer physics*, Oxford university press New York, 2003.

124. 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.

125. A. Chandrasekaran, C. Kim, S. Venkatram and R. Ramprasad, *Macromolecules*, 2020, **53**, 4764-4769.

126. M.-X. Zhu, H.-G. Song, Q.-C. Yu, J.-M. Chen and H.-Y. Zhang, *Int. J. Heat Mass Transfer*, 2020, **162**, 120381.

127. C. Kim, A. Chandrasekaran, A. Jha and R. Ramprasad, *MRS Commun.*, 2019, **9**, 860-866.

128. D. Reker and G. Schneider, *Drug Discovery Today*, 2015, **20**, 458-465.