

Using GNN property predictors as molecule generators

Received: 2 July 2024

Accepted: 16 April 2025

Published online: 08 May 2025

Félix Therrien , Edward H. Sargent  & Oleksandr Voznyy  

Graph neural networks (GNNs) have emerged as powerful tools to accurately predict materials and molecular properties in computational and automated discovery pipelines. In this article, we exploit the invertible nature of these neural networks to directly generate molecular structures with desired electronic properties. Starting from a random graph or an existing molecule, we perform a gradient ascent while holding the GNN weights fixed in order to optimize its input, the molecular graph, towards the target property. Valence rules are enforced strictly through a judicious graph construction. The method relies entirely on the property predictor; no additional training is required on molecular structures. We demonstrate the application of this method by generating molecules with specific energy gaps verified with density functional theory (DFT) and with specific octanol-water partition coefficients (logP). Our approach hits target properties with rates comparable to or better than state-of-the-art generative models while consistently generating more diverse molecules. Moreover, while validating our framework we created a dataset of 1617 new molecules and their corresponding DFT-calculated properties that could serve as an out-of-distribution test set for QM9-trained models.

One of the ultimate goals of computational materials science is to rapidly identify promising material structures and compositions with specific properties to guide experimental scientists and automated laboratories¹. This is particularly true in the time-critical fields of pharmacy and materials for energy and sustainability where a number of large scale autonomous experimentation initiatives are underway^{2,3}. In the past decades, computational materials discovery has been achieved by going through large databases of existing materials and computing properties from first principles using methods such as density functional theory (DFT) and molecular dynamics^{4,5}. These methods have proven to be successful in many cases^{6–9}, but suffer from two main limitations: (1) they are computationally expensive and (2) they cover a small subspace of all possible materials.

In an effort to alleviate the first problem, machine learning (ML) based property prediction methods have become an integral part of materials science^{10,11}. Various models exploit different materials representations (e.g., graphs, fingerprints)^{12,13}, model architectures (e.g., neural networks, random forests)¹⁴ and datasets (experimental or computed) and their accuracy has been steadily increasing, often

competing with that of DFT¹⁰. The success and adoption of these models is due largely to the powerful tools developed by the ML and data science communities (such as Pytorch¹⁵, Tensorflow¹⁶, Pandas¹⁷, etc.). However, despite their promising performance on benchmark datasets, ML property predictors still suffer from poor generalizability¹⁸, exhibiting much lower performance on out-of-distribution data, i.e., materials that are different from what they have been trained on.

Materials and molecule generation can alleviate the second limitation: it can theoretically explore the full space of all possible materials. Traditionally this has been done using minima hopping¹⁹, metadynamics²⁰ and evolutionary approaches^{21–23}, but machines can also learn to generate realistic materials⁴. The goal is no longer to predict properties but to predict realistic structures²⁴. Materials and molecular generation using ML is a rapidly evolving field with a large body of recent methods including variational autoencoders^{25–27}, flow-based models^{28,29}, diffusion models^{30,31}, models based on reinforcement learning (RL)^{32,33} and many others^{34,35}.

In this paper, we exploit one of the most important and fundamental features of neural networks, their differentiability, to directly

optimize a target property with respect to the graph representation itself starting from a pre-trained predictive model. This concept sometimes termed *gradient ascent*, or *input optimization*, has been used extensively in other fields^{36,37} and a similar idea has been applied to molecular generation with SELFIES³⁸. Here we apply input optimization to molecular GNNs. We describe how carefully constraining the molecular representation makes this “naive” approach possible and show that it can generate molecules with requested properties as verified with density functional theory and empirical models. It does so with comparable or better performance than existing methods while consistently generating the most diverse set of molecules.

Results

Rationale and workflow overview

Our method can technically be applied to any GNN architecture that uses molecular graphs. To train our GNN we use (1) an explicit representation of the adjacency matrix, A , where non-zero elements are the bond orders and (2) a feature matrix, F , often referred to as a *feature vector*, that contains a one-hot representation of the atoms. These two matrices fully describe the graph and contain exactly the same information as a SMILES string. Since all functions in the GNN have well-defined gradients (allowing it to be trained in the first place), the adjacency matrix and the feature vector can be optimized through a gradient descent with respect to a target property as illustrated in Fig. 1. This is termed *gradient ascent*—although it does not change the direction of the gradients but rather, the variable with respect to which they are taken. This approach could be seen as a “naive” way to tackle the problem of conditional molecular and materials generation; unconstrained, it would lead to meaningless results that do not follow the basic structures of the adjacency matrix (e.g., its symmetry) and the feature matrix (e.g., one non-zero element per line). The major contribution of this paper is to enforce structural and chemical rules such that optimized inputs can only be valid molecules allowing for direct optimization into graph space.

The adjacency matrix is constructed from a weight vector \mathbf{w}_{adj} containing $\frac{N^2-N}{2}$ elements. These elements are squared and populated in an upper triangular matrix with zeros on the main diagonal. The resulting matrix is then added to its transpose to obtain a positive symmetric matrix with zero trace. Elements of the matrix are then rounded to the nearest integer to obtain the adjacency matrix. The key element here is that the adjacency matrix needs to have non-zero gradients with respect to \mathbf{w}_{adj} , which is not the case when using a conventional rounding half-up function. To alleviate this problem we used a sloped rounding function,

$$[x]_{\text{sloped}} = [x] + \alpha(x - [x]) \quad (1)$$

where $[x]$ is the conventional rounding half-up function and α is an adjustable hyper-parameter. These steps guarantee that only valid near-integer filled adjacency matrices are constructed. However, it does not take into account any chemistry: atoms are allowed to form as many bonds as there are rows in the adjacency matrix. To avoid that, we use two strategies: (1) we penalize valence (sum of bond orders) of more than 4 through the loss function and (2) we do not allow gradients in the direction of higher number of bonds when the valence is already 4.

The feature vector, on the other hand, is constructed directly from the adjacency matrix. The idea is to define the atoms from their valence, i.e., the sum of their bond orders. For example, a node with four edges of value one, or, in other words, an atom forming four single bonds, would be defined as a carbon atom, a node forming one double bond would be defined as an oxygen atom, a node forming a double bond and a single bond would be defined as a nitrogen atom, etc. In terms of matrices, this means that the sum of a row (or column) of the adjacency matrix defines the element associated with that row (or column).

To deal with an arbitrary number of elements that may have the same valence, an additional weight matrix \mathbf{w}_{fea} is used to differentiate between these elements. If the valence of an atom is 1, the weights of the corresponding row in \mathbf{w}_{fea} specify if this atom is H, F or Cl for example.

More details on how constraints on the adjacency matrix and feature vectors are implemented can be found in the supplementary information.

Energy gap targeting

The first task is to generate molecules with a specific energy gap μ between their highest occupied molecular orbital (HOMO) and their lowest unoccupied molecular orbital (LUMO). The HOMO-LUMO gap is a particularly interesting property because it is relevant to a number of applications, it is relatively expensive to compute and it is available in many databases^{39–42}. Furthermore, the ability to generate molecules with a specific emission wavelength is of special interest to our own research on the discovery of efficient blue organic light emitting diodes (OLED) materials.

We trained a simple GNN proxy model on the QM9 dataset³⁹ and used our direct inverse design procedure to generate 100 molecules with HOMO-LUMO gaps within 10 meV of 3 different target values: the first percentile of QM9 energy gaps, 4.1 eV, the median, 6.8 eV, and the 99th percentile, 9.3 eV. As an additional soft target, we aimed for atomic fractions (the relative number of each elements in the molecule) close to the average atomic fractions in QM9. This helps guide the generation towards molecules that the proxy can predict better.

The results are illustrated in Fig. 2. Note that all generated molecules are within the requested range according to the proxy model by

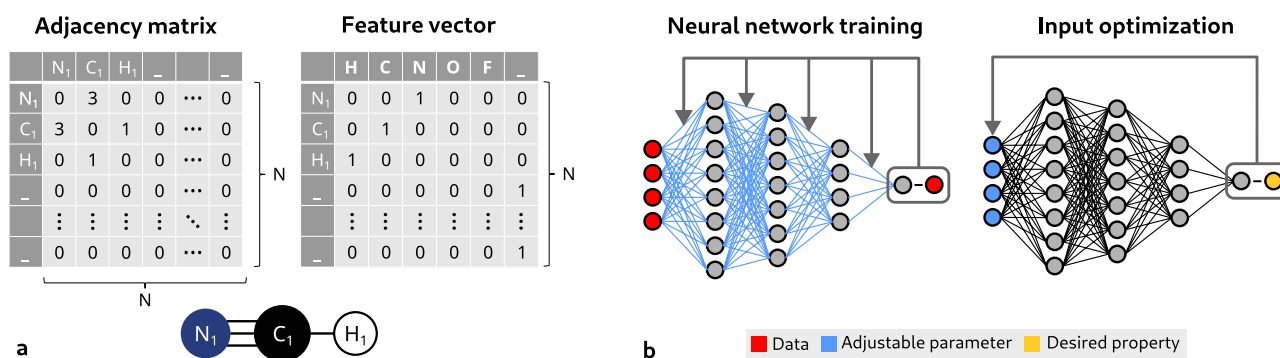


Fig. 1 | Key concepts of the generative framework. **a** Molecular representation for this work using an HCN molecule as an example. **b** A visual representation of a typical training process for a neural network in comparison to an input optimization scheme.

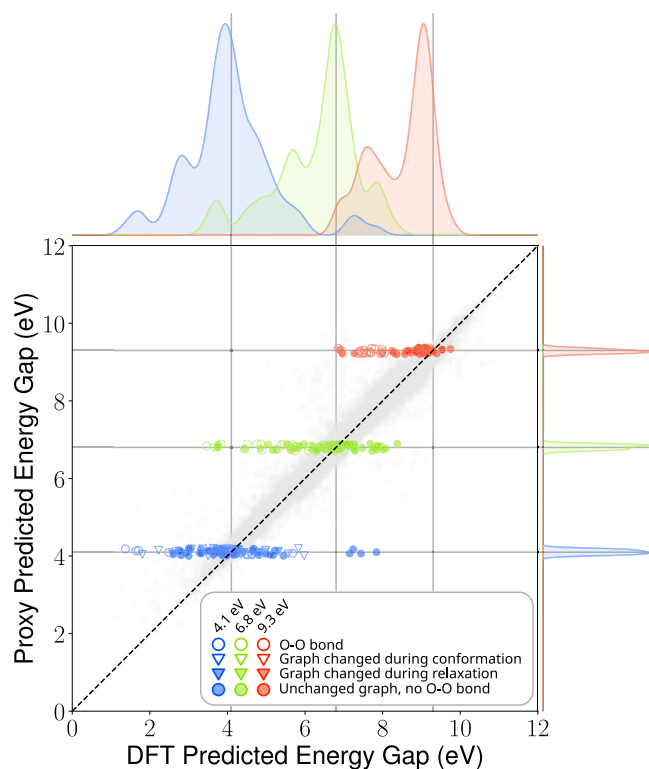


Fig. 2 | Generated molecules energy gap, density functional theory (DFT) and proxy predictions. Generated molecules are overlaid on the proxy model performance on the QM9 dataset (test + train) in grey. The shape and filling of the dots highlight different aspects of the generated molecule that may affect predictive performance. Open dots indicate a trait that can be detected before the calculation. Source data are provided as a Source Data file.

construction: generation stops when that criterion is met. The DFT-calculated energy gap, on the other hand, is distributed around the requested property with relatively small overlap between different targets.

The generated molecules predictions are overlaid on the predictions for the QM9 dataset. It is apparent that the proxy model's performance is significantly worse on generated molecules than on the test set. If the model was generalizing perfectly, we should expect the performance to be similar to that of the test set, MAE=0.12eV, rather than the observed performance of about 0.8 eV. This highlights the importance of benchmarking generating schemes on DFT-confirmed properties, not solely on ML predicted properties.

We compared our method to JANUS³⁵ an ML enhanced state-of-the-art genetic algorithm that was recently tested against several materials generations schemes on various benchmarks⁴³. We chose to compare our method to a genetic algorithm, because of their prevalence, their performance⁴⁴ and because, like our method they do not require any training other than that of the proxy model. We ran the algorithm directly with DFT as an evaluation function and with our proxy model; the results are presented in Table 1 and in Figure S6. Details of the calculations and JANUS model parameters can be found in the SI.

As a measure of performance, for each target, we counted the number of molecules within 0.5 eV of the target, the mean absolute distance from the target value and the average Tanimoto distance between Morgan fingerprints of each pair molecules within 0.5 eV of the target. In Table 1 we refer to our method as DIDgen, direct inverse design generator. DIDgen and JANUS are both able to significantly increase the proportion of molecules within the target range compared to a random draw of QM9 molecules. Our approach nearly

matches or outperforms the genetic algorithm for all 9 metrics in Table 1.

DIDgen takes on average 12.0 s to generate an in-target molecule for 4.1 eV, 2.1 s for 6.8 eV and 10.4 s for 9.3 eV whereas JANUS takes about 100 s to generate 100 in-target molecules (1 s per molecule) for all targets using the same computer (4-CPU, 3.40 GHz). Timing varies significantly depending on the task and parameters used for both methods. DIDgen generates all molecules completely independently; all 100 molecules could be generated simultaneously. We did not implement batch generation with the use of a GPU, but it might significantly enhance performance in future versions.

logP targeting

The second task is to target a specific range of octanol-water partition coefficient (logP) values. It is relevant for drug discovery where logP can be used as a measure of cell permeability^{45,46}. Most commercial drugs have a value between 0 and 5. In more recent studies in the field of ML, logP and penalized logP have been used extensively as a benchmark for generative models due to the existence of a cheap empirical model for logP developed by Wildman and Crippen⁴⁵ sometimes called "Crippen logP" that is readily available in the RDKit⁴⁷. Here we will use the same target range as³³ which were used in several recent papers on molecular generation.

We trained "CrippenNet" a GNN developed specifically for this task on a subset of the ZINC dataset^{26,48} and QM9³⁹. More details about CrippenNet can be found in the Methods and in the supplementary information. For each of the two target ranges ([−2.5, −2], [5, 5.5]) we generated 1000 molecules and evaluated their diversity using the average pairwise Tanimoto distance—we evaluated the diversity of all generated molecules, not only the ones in the target range to be consistent with ref. 33. We limited the molecule size to 85 atoms (including hydrogens), again to be consistent with ref. 33. For this task generated molecules can contain the same element types as in the ZINC250 dataset: C, O, N, F, H, S, Cl, Br, I, and P. We initialized the generation with random molecules from QM9 because CrippenNet performed significantly better on these molecules.

The results are presented in Table 2 in comparison with refs. 25,27,32–34. DIDgen generates the most diverse molecules of all methods for both target ranges. When compared to other methods that use a trained proxy model as a predictor for logP, as opposed to the ground truth empirical model which we termed "oracle" in Table 2, DIDgen shows the highest performance by a factor of x4 and x2 respectively. However, it does not have a higher success rate than GCPN and SGDS which both use the oracle directly. This is not surprising since the success rate relies heavily on the proxy model performance and generalizability. All methods except JT-VAE could technically use CrippenNet which would offer a way to compare the generation schemes themselves separately of the proxy model's performance (like we did for the energy gap task).

DIDgen takes on average 5.6 s to generate a molecule in the −2.5 to −2 range and it takes on average 3.4 s to generate a molecule in the 5 to 5.5 range on a local machine (4-CPU, 3.40GHz). The authors of LIMO report generating 33 molecules per second within the target range on 2 GTX 1080 Ti GPUs. Other authors did not report their compute times. Again, the performance of DIDgen could be improved with trivial parallelization and batch generation on GPUs.

Discussion

Generative methods like ours that use a learned proxy rely heavily on its performance, especially its ability to generalize in order to hit target properties. Models that can directly use an oracle do not always suffer from this limitation. For example, in the logP targeting task, the oracle was not computationally limiting and thus the number of evaluations was not taken into account. In that case, oracle-based methods had superior success rates. But since oracle evaluations are not costly, as

Table 1 | Comparison of generated molecules from our procedure (DIDgen) with those generated with a state-of-the-art genetic algorithm (JANUS)³⁵

Method	4.1 eV				6.8 eV				9.3 eV				ρ
	n_{calcs}	$n_{\pm 0.5}$	MAE	Div.	n_{calcs}	$n_{\pm 0.5}$	MAE	Div.	n_{calcs}	$n_{\pm 0.5}$	MAE	Div.	
QM9		3.4%		0.87		24%		0.89		6.0%		0.84	–
JANUS (DFT)	197	24 (12.2%)	0.96	0.79	392	42 (10.7%)	0.92	0.80	484	26 (5.4%)	1.28	0.81	–
JANUS (Proxy)	100	36	1.05	0.86	100	46	0.80	0.82	100	37	1.24	0.81	0.78
DIDgen	100	46	0.81	0.91	100	50	0.83	0.90	100	34	0.83	0.83	0.86

The comparison metrics are the number of density functional theory (DFT) calculations (n_{calcs}), the number of molecules that are within 0.5 eV of the target ($n_{\pm 0.5}$), the mean absolute error (MAE) to the target property and the diversity (Div.) of the generated molecules that are within 0.5 eV of the property. Pearson correlation coefficients (ρ) between all 300 ML and DFT predictions are reported in the last column. The highest performance for each metric is in bold.

Table 2 | LogP targeting to a certain range. Data from Ref. 34

Method	Evaluation	$-2.5 \leq \log P \leq -2$		$5 \leq \log P \leq 5.5$	
		Success	Diversity	Success	Diversity
ZINC		0.4%	0.919	1.3%	0.901
ORGAN ³²	Oracle	0	–	0.2%	0.909
GCPN ³³	Oracle	85.5%	0.392	54.7%	0.855
SGDS ³⁴	Oracle	86.0%	0.874	62.2%	0.858
JT-VAE ²⁵	Proxy	11.3%	0.846	7.6%	0.907
LIMO ²⁷	Proxy	10.4%	0.914	–	–
DIDgen	Proxy	43.5%	0.932	14.4%	0.917

The highest performance for each metric in each category (Oracle or Proxy) is in bold.

pointed out by other groups of authors²⁷, it would be easy to generate more molecules and filter out molecules that do not meet the criteria.

For instance, when targeting logP between -2.5 and -2 , since the rate of success of DIDgen is about half of that of SGDS, we could simply generate twice as many molecules to obtain the same number of in-target molecules. Since the oracle is inexpensive, this would be easy to verify. In fact, the success rate metric makes little sense outside the context of budgeted (expensive) evaluations: in principle one can always make an algorithm that “generates” a molecule only if it reaches the target. For example, a genetic algorithm that would iterate until the entire population is inside the target range would achieve exactly that. The success rate becomes useful when one has a budget of x oracle evaluations and wants to obtain the highest number of true positives within that budget.

Furthermore, application-relevant properties are often computationally expensive and directly optimizing them is not an option, this is actually the case with the water-octanol partition (logP) which is difficult to evaluate precisely beyond Wildman and Crippen’s model⁴⁹. The energy gap task is another example of that. In that case, we have seen that JANUS was able to find significantly more molecules in the target range when using a proxy than when using DFT directly because it could not go through enough generations within a reasonable time limit. So, even though JANUS does not require the use of a proxy, it clearly benefits from its use. This means that for a lot of applications, methods that can use an oracle will also end up being dependant on the performance and generalizability of an ML proxy.

In that same vein, inverting a predictive ML model with our method can be a good way to identify its weaknesses. For example, in the energy gap task, many of the generated molecules for which the proxy model was least accurate contained an O-O bond. The presence of that bond is particularly important for the large gaps (9.3 eV). As illustrated in Fig. 2, there is a group of generated molecules containing O-O bonds around a DFT gap of 8 eV which are all predicted to have a

gap of around 9.3 eV. This explains, in part, the poor performance of the model on generated data. In the future, our method could be used as part of an active learning loop to iteratively improve the model generalizability, in this case, by retraining on a set of molecules containing O-O bonds, for example.

This drop in model performance is observed across all generated molecules including molecules generated with JANUS using DFT and with DIDgen using the Graph Isomorphism Network (GIN)⁵⁰. To verify if this drop in performance would also affect more complex models, we trained, PaiNN⁵¹, an equivariant message passing network on QM9 and tested its performance on generated molecules. Despite having a much lower MAE on QM9 data (0.048 eV vs. 0.12 eV), PaiNN performed worse than our model on generated molecules with an MAE of 1.16 eV versus 0.96 eV for our model (see Figure S7 and Table S2). This suggests that generating molecules might be a good way to obtain out of distribution data that could be used for benchmarking and fine-tuning ML models. As a first step in that direction, we provide on our public repository a dataset of 1617 unique generated molecules with their corresponding DFT relaxed 3D coordinates as well as their dipole moments, HOMO energy, LUMO energy and energy gap. This data was generated to obtain Table 1 and Table S1.

Another way to evaluate the performance of generative models that is independent of the evaluation method is to measure their ability to generate diverse molecules. On that metric, our method performed better than all other methods we have tested it against. This is important because for any application, new materials must simultaneously meet multiple criteria that cannot all be modeled and optimized for. For example, molecules for display applications must emit light of a certain color, but they must also be synthesizable, bright, stable, cheap, safe, soluble, etc. Generating a more diverse set of molecules simply increases the chances that one of the generated molecules will meet multiple of these criteria.

The inversion procedure presented here can be used on any GNN architecture that uses molecular 2D graphs (like SMILES), but it requires modifying them so that they use an explicit representation of the adjacency matrix which may require retraining. This can be a limiting factor for the adoption of our procedure. To help ease this process, in addition to the implementations of our simple energy gap GNN and CrippenNet, we implemented and tested 5 commonly used GNNs: Graph Isomorphism Network (GIN)⁵⁰, Graph Attention Networks (GATs)⁵², GraphSAGE (SAmple and aggreGatE)⁵³, Graph Convolutional Network (GCN)⁵⁴ and Crystal Graph Convolutional Neural Networks (CGCNN)⁵⁵. For GIN, GAT, GraphSAGE and GCN, we started from the implementations found in ref. 56. These models can either be used directly to train and generate, or as a starting point to implement other models. For example, we performed the energy gap targeting task with GIN and compared the properties of the generated molecule with DFT. The model predictive performance matched the literature without any hyper-parameter tuning and the generation performance was slightly worse than when using our GNN. Detailed results are presented in the SI.

In conclusion, we demonstrated that by carefully constraining the representation of molecules, a property predicting GNN can be turned into a diverse conditional generator without any additional training. Our method is a literal implementation of the inverse design paradigm: properties are put in and materials are obtained going backwards through a pipeline that is built to do the opposite. Although it is a “naive” way to approach this problem, we showed that it rivals with complex state-of-the-art models. Ultimately, we hope that will help accelerate the design and discovery of functional materials.

Methods

Training of the property predictors

Since optimizing the inputs (generating molecules) will require an explicit representation of the adjacency matrix, sparse matrices and lists of edge indices cannot be used during training of the property predictor GNN. Mini-batching using a single adjacency matrix per batch thus uses a large amount of memory that scales quadratically with the total number of atoms in a batch. To avoid that issue, we use a fixed graph size (N) instead which allows us to store the adjacency matrices as a $B \times N \times N$ array where B is the batch size. Empty rows of the adjacency matrix are associated with atoms of type “no-atom” in the feature vector one-hot encoding. This allows for molecules of any size smaller than N .

Although sloped rounding and maximum functions guarantee that the adjacency matrix and the feature vector are populated with near-integer values, these values are not exactly integers and these variations around integer values can have a significant impact on the property predicated by the GNN. To make the GNN less sensitive to these variations, we add a small amount of random noise around integer values of A and F during training.

For the energy gap task, we used a simple graph convolutional network (GCN) with an adjustable number of convolutions, a pooling layer and an adjustable number of linear layers. We trained this model on the QM9 dataset containing approximately 130000 small molecules for which the energy gap was calculated with density functional theory (DFT). With manual hyperparameter tuning, we obtain a mean absolute error of about 0.12 eV on the test set which is on par with other models using the same amount of information (2D graphs, no coordinates)³⁹. More details on this model can be found in the SI.

For the LogP task, we use CrippenNet a graph neural network that we developed based on the structure of Crippen’s empirical model. CrippenNet classifies atoms inside molecules as one of 69 classes associated with a logP value and then sums the individual contributions of each atom to obtain the total logP of the molecules. In the empirical model, each of these classes is defined by certain properties

described by SMARTS string. For example, class #18 is an aromatic carbon atom and its associated logP is 0.1581⁴⁵. In CrippenNet the classes are learned from the graph of the molecule by training on a subset of the ZINC dataset⁴⁸ containing 250,000 molecules (the same subset used in Ref. 26) and QM9. More details on CrippenNet can be found in the supplemental information. Our goal was to get a model that resembles the ground truth model as much as possible in order to be as close as possible to directly inverting it. Note that it may be possible to obtain a perfect classification from the graph representation of the molecule in which case DIDgen’s success rate would be 100%. This would require a careful analysis of the SMARTS classes and their resulting convolutions.

DFT validation of the energy gap

For each generated molecule, we obtained the conformation (3D atomic positions) using the RDKit. In some cases no conformation could be obtained from the generated molecule. In those cases, the molecule was discarded and a new molecule was generated until a total of 100 molecules with conformation were obtained for each target. About 19% of initially generated molecules were discarded when targeting 4.1 eV, 2% when targeting 6.8 eV and 1% when targeting 9.3 eV.

For all 300 molecules, we performed density functional theory calculations to obtain the “true” HOMO-LUMO gap. For consistency, we used DFT parameters as close as possible to Ramakrishnan et al.³⁹. We fully relaxed the molecules and obtained the HOMO-LUMO gap with B3LYP/6-31G(2df,2p) as implemented in ORCA⁵⁷.

Loss function for the inversion procedure

In order to improve generative performance, we added a simple additional term (L_c) to the loss function: the average atomic fraction, C_i , in QM9 for each element i , i.e. the fraction of all atoms that are of type i . The total loss for this task was the following:

$$L = \frac{\lambda_l L_l^2 + \lambda_s L_s^2 + \lambda_c L_c^2}{\lambda_l + \lambda_s + \lambda_c}, \quad (2)$$

where

$$L_c = \sum_i \frac{\sum_j F_{ij}}{\sum_{ij} F_{ij}} - C_i, \quad (3)$$

L_l is the root mean square error with respect to the target p , L_s is the loss associated with the maximum valence defined in equation S1 of the supplementary information, F is the feature vector and $\lambda_{l,s,c}$ are adjustable hyperparameters. Generated molecules with balanced valence could include molecules with disproportional amounts of certain elements (by replacing all H with F, for instance). The idea of this loss is to steer the generation away from these cases for which the proxy model would underperform and where there would likely be no valid conformation. It is a very simple example of how the loss can be used to guide generation towards molecules with desirable properties. More complex co-objectives such as accessibility scores could be used in the future.

Reporting summary

Further information on research design is available in the Nature Portfolio Reporting Summary linked to this article.

Data availability

A dataset of 2410 DFT calculations performed for this article including calculations for 1617 generated molecules is available at github.com/ftherrien/inv-design/tree/master/dataset⁵⁸. Source data for all figures are provided with this paper.

Code availability

The code used for the experiments presented in this paper is available at github.com/ftherrien/inv-design^{58–60}.

References

1. Zunger, A. Inverse design in search of materials with target functionalities. *Nat. Rev. Chem.* **2**, 1–16 (2018).
2. Stach, E. et al. Autonomous experimentation systems for materials development: A community perspective. *Matter* **4**, 2702–2726 (2021).
3. Abolhasani, M. & Kumacheva, E. The rise of self-driving labs in chemical and materials sciences. *Nat. Synth.* **2**, 483–492 (2023).
4. Lu, Z. Computational discovery of energy materials in the era of big data and machine learning: a critical review. *Mater. Reports: Energy* **1**, 100047 (2021).
5. Therrien, F., Jones, E. B. & Stevanović, V. Metastable materials discovery in the age of large-scale computation. *Appl. Phys. Rev.* **8**, 031310 (2021).
6. Gorai, P., Stevanović, V. & Toberer, E. S. Computationally guided discovery of thermoelectric materials. *Nat. Rev. Mater.* **2**, 1–16 (2017).
7. Garrity, K. F. High-throughput first-principles search for new ferroelectrics. *Phys. Rev. B* **97**, 024115 (2018).
8. Curtarolo, S. et al. The high-throughput highway to computational materials design. *Nat. Mater.* **12**, 191–201 (2013).
9. Alberi, K. et al. The 2019 materials by design roadmap. *J. Phys. D: Appl. Phys.* **52**, 013001 (2018).
10. Schmidt, J., Marques, M. R., Botti, S. & Marques, M. A. Recent advances and applications of machine learning in solid-state materials science. *npj Computational Mater.* **5**, 1–36 (2019).
11. Butler, K. T., Davies, D. W., Cartwright, H., Isayev, O. & Walsh, A. Machine learning for molecular and materials science. *Nature* **559**, 547–555 (2018).
12. Reiser, P. et al. Graph neural networks for materials science and chemistry. *Commun. Mater.* **3**, 93 (2022).
13. Wigh, D. S., Goodman, J. M. & Lapkin, A. A review of molecular representation in the age of machine learning. *Wiley Interdiscip. Reviews: Computational Mol. Sci.* **12**, e1603 (2022).
14. Wei, J. et al. Machine learning in materials science. *InfoMat* **1**, 338–358 (2019).
15. Paszke, A. et al. Pytorch: An imperative style, high-performance deep learning library. *Advances in neural information processing systems* **32** (2019).
16. Abadi, M. et al. TensorFlow: Large-scale machine learning on heterogeneous systems (2015). <https://www.tensorflow.org/>. Software available from tensorflow.org
17. pandas development team, T. pandas-dev/pandas: Pandas. <https://doi.org/10.5281/zenodo.3509134> (2020).
18. Li, K., DeCost, B., Choudhary, K., Greenwood, M. & Hatrick-Simpers, J. A critical examination of robustness and generalizability of machine learning prediction of materials properties. *npj Computational Mater.* **9**, 55 (2023).
19. Goedecker, S. Minima hopping: An efficient search method for the global minimum of the potential energy surface of complex molecular systems. *J. Chem. Phys.* **120**, 9911–9917 (2004).
20. Martoňák, R., Laio, A. & Parrinello, M. Predicting crystal structures: the parrinello-rahman method revisited. *Phys. Rev. Lett.* **90**, 075503 (2003).
21. Glass, C. W., Oganov, A. R. & Hansen, N. Uspex—evolutionary crystal structure prediction. *Computer Phys. Commun.* **175**, 713–720 (2006).
22. Jensen, J. H. A graph-based genetic algorithm and generative model/monte carlo tree search for the exploration of chemical space. *Chem. Sci.* **10**, 3567–3572 (2019).
23. Oganov, A. R., Pickard, C. J., Zhu, Q. & Needs, R. J. Structure prediction drives materials discovery. *Nat. Rev. Mater.* **4**, 331–348 (2019).
24. Sanchez-Lengeling, B. & Aspuru-Guzik, A. Inverse molecular design using machine learning: Generative models for matter engineering. *Science* **361**, 360–365 (2018).
25. Jin, W., Barzilay, R. & Jaakkola, T. Junction tree variational auto-encoder for molecular graph generation. In *International conference on machine learning*, 2323–2332 (PMLR, 2018).
26. Gómez-Bombarelli, R. et al. Automatic chemical design using a data-driven continuous representation of molecules. *ACS Cent. Sci.* **4**, 268–276 (2018).
27. Eckmann, P. et al. Limo: Latent inceptionism for targeted molecule generation. In *International Conference on Machine Learning*, vol. 162, 5777–5792 (PMLR, 2022).
28. Bengio, E., Jain, M., Korablyov, M., Precup, D. & Bengio, Y. Flow network based generative models for non-iterative diverse candidate generation. *Adv. Neural Inf. Process. Syst.* **34**, 27381–27394 (2021).
29. Roy, J., Bacon, P.-L., Pal, C. & Bengio, E. Goal-conditioned gflownets for controllable multi-objective molecular design. *arXiv preprint arXiv:2306.04620* (2023).
30. Vignac, C. et al. Digress: Discrete denoising diffusion for graph generation. In *International Conference on Learning Representations* (2023).
31. Xu, M. et al. Geodiff: A geometric diffusion model for molecular conformation generation. In *International Conference on Learning Representations* (2022).
32. Guimaraes, G. L., Sanchez-Lengeling, B., Outeiral, C., Farias, P. L. C. & Aspuru-Guzik, A. Objective-reinforced generative adversarial networks (organ) for sequence generation models. *arXiv preprint arXiv:1705.10843* (2017).
33. You, J., Liu, B., Ying, Z., Pande, V. & Leskovec, J. Graph convolutional policy network for goal-directed molecular graph generation. *Advances in neural information processing systems* **31** (2018).
34. Kong, D., Pang, B., Han, T. & Wu, Y. Molecule design by latent space energy-based modeling and gradual distribution shifting. *Conference on Uncertainty in Artificial Intelligence*. <https://arxiv.org/abs/2306.14902v1> (2023).
35. Nigam, A., Pollice, R. & Aspuru-Guzik, A. Parallel tempered genetic algorithm guided by deep neural networks for inverse molecular design. *Digital Discovery* **1**, 390–404 (2022).
36. Trabucco, B., Geng, X., Kumar, A. & Levine, S. Design-bench: Benchmarks for data-driven offline model-based optimization. In *International Conference on Machine Learning*, vol. 162, 21658–21676 (PMLR, 2022).
37. Linder, J. & Seelig, G. Fast activation maximization for molecular sequence design. *BMC Bioinforma.* **22**, 1–20 (2021).
38. Shen, C., Krenn, M., Eppel, S. & Aspuru-Guzik, A. Deep molecular dreaming: Inverse machine learning for de-novo molecular design and interpretability with surjective representations. *Mach. Learning: Sci. Technol.* **2**, 03LT02 (2021).
39. Ramakrishnan, R., Dral, P. O., Rupp, M. & Von Lilienfeld, O. A. Quantum chemistry structures and properties of 134 kilo molecules. *Sci. data* **1**, 1–7 (2014).
40. Wu, Z. et al. Moleculenet: a benchmark for molecular machine learning. *Chem. Sci.* **9**, 513–530 (2018).
41. Nakata, M. & Shimazaki, T. Pubchemqc project: a large-scale first-principles electronic structure database for data-driven chemistry. *J. Chem. Inf. modeling* **57**, 1300–1308 (2017).
42. Hachmann, J. et al. The harvard clean energy project: large-scale computational screening and design of organic photovoltaics on the world community grid. *J. Phys. Chem. Lett.* **2**, 2241–2251 (2011).
43. Nigam, A. et al. Tartarus: A benchmarking platform for realistic and practical inverse molecular design. *Advances in Neural Information Processing Systems*, vol. 36, 3263–3306 (2023).

44. Tripp, A. & Hernández-Lobato, J. M. Genetic algorithms are strong baselines for molecule generation. *arXiv preprint arXiv:2310.09267* (2023).
45. Wildman, S. A. & Crippen, G. M. Prediction of physicochemical parameters by atomic contributions. *J. Chem. Inf. computer Sci.* **39**, 868–873 (1999).
46. Lipinski, C. A., Lombardo, F., Dominy, B. W. & Feeney, P. J. experimental and computational approaches to estimate solubility and permeability in drug discovery and development settings. *Adv. Drug. Deliv. Rev.* **46**, 00129–O (2001).
47. Landrum, G. et al. rdkit/rdkit: 2024_03_1 (q1 2024) release. <https://doi.org/10.5281/zenodo.10893044> (2024).
48. Irwin, J. J., Sterling, T., Mysinger, M. M., Bolstad, E. S. & Coleman, R. G. Zinc: a free tool to discover chemistry for biology. *J. Chem. Inf. modeling* **52**, 1757–1768 (2012).
49. Plante, J. & Werner, S. Jplogp: an improved logp predictor trained using predicted data. *J. Cheminform.* **10**. <https://www.ncbi.nlm.nih.gov/pmc/articles/PMC6755606>.
50. Xu, K., Hu, W., Leskovec, J. & Jegelka, S. How powerful are graph neural networks? *International Conference on Learning Representations* (2019).
51. Schütt, K., Unke, O. & Gastegger, M. Equivariant message passing for the prediction of tensorial properties and molecular spectra. In *International Conference on Machine Learning*, 9377–9388 (PMLR, 2021).
52. Veličković, P. et al. Graph attention networks. *International Conference on Learning Representations* (2018).
53. Hamilton, W., Ying, Z. & Leskovec, J. Inductive representation learning on large graphs. *Advances in neural information processing systems* **30** (2017).
54. Duvenaud, D. K. et al. Convolutional networks on graphs for learning molecular fingerprints. *Advances in neural information processing systems* **28** (2015).
55. Xie, T. & Grossman, J. C. Crystal graph convolutional neural networks for an accurate and interpretable prediction of material properties. *Phys. Rev. Lett.* **120**, 145301 (2018).
56. Liu, S. et al. Symmetry-informed geometric representation for molecules, proteins, and crystalline materials. *Advances in neural information processing systems* **36** (2024).
57. Neese, F., Wennmohs, F., Becker, U. & Riplinger, C. The orca quantum chemistry program package. *The Journal of chemical physics* **152** (2020).
58. Therrien, F. ftherrien/inv-design: Didgen (as published + license). <https://doi.org/10.5281/zenodo.15086610> (2025).
59. Kingma, D. P. & Ba, J. Adam: A method for stochastic optimization. *International Conference on Learning Representations*. <https://arxiv.org/abs/1412.6980v9> (2014).
60. Resheff, Y. S., Mandelbom, A. & Weinshall, D. Controlling imbalanced error in deep learning with the log bilinear loss. In *First International Workshop on Learning with Imbalanced Domains: Theory and Applications*, 141–151 (PMLR, 2017).

Acknowledgements

We thank Dr. Alexander Davis from the University of Toronto for insightful discussions, ideas and valuable feedback. The authors acknowledge support from The Alliance for AI-Accelerated Materials Discovery (A3MD), which includes funding from Total Energies SE., BP, Meta, and LG AI Research.

Author contributions

F.T. wrote the manuscript, designed the algorithm and performed the experiments, O.V. and E.H.S supervised the work.

Competing interests

The authors declare no competing interests.

Additional information

Supplementary information The online version contains supplementary material available at <https://doi.org/10.1038/s41467-025-59439-1>.

Correspondence and requests for materials should be addressed to Oleksandr Voznyy.

Peer review information *Nature Communications* thanks Camille Bilo-deau and the other anonymous reviewer(s) for their contribution to the peer review of this work. A peer review file is available.

Reprints and permissions information is available at <http://www.nature.com/reprints>

Publisher's note Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.

Open Access This article is licensed under a Creative Commons Attribution-NonCommercial-NoDerivatives 4.0 International License, which permits any non-commercial use, sharing, distribution and reproduction in any medium or format, as long as you give appropriate credit to the original author(s) and the source, provide a link to the Creative Commons licence, and indicate if you modified the licensed material. You do not have permission under this licence to share adapted material derived from this article or parts of it. The images or other third party material in this article are included in the article's Creative Commons licence, unless indicated otherwise in a credit line to the material. If material is not included in the article's Creative Commons licence and your intended use is not permitted by statutory regulation or exceeds the permitted use, you will need to obtain permission directly from the copyright holder. To view a copy of this licence, visit <http://creativecommons.org/licenses/by-nc-nd/4.0/>.

© The Author(s) 2025