Assessing Deep Generative Models in Chemical Composition Space

Hanna Türk*,1, 2 Elisabetta Landini*,1, 2 Christian Kunkel*,1 Johannes T. Margraf,1, a) and Karsten Reuter1

1) Fritz-Haber-Institut der Max-Planck-Gesellschaft, Faradayweg 4-6, D-14195 Berlin, Germany
2) Chair for Theoretical Chemistry and Catalysis Research Center, Department of Chemistry, Technische Universität München, Lichtenbergstraße 4, 85748 Garching, Germany

The computational discovery of novel materials has been one of the main motivations behind research in theoretical chemistry for several decades. Despite much effort, this is far from a solved problem, however. Among other reasons, this is due to the enormous space of possible structures and compositions that could potentially be of interest. In the case of inorganic materials, this is exacerbated by the combinatorics of the periodic table, since even a single crystal structure can in principle display millions of compositions. Consequently, there is a need for tools that enable a more guided exploration of the materials design space. Here, generative machine learning (ML) models have recently emerged as a promising technology. In this work, we assess the performance of a range of deep generative models based on Reinforcement Learning (RL), Variational Autoencoders (VAE) and Generative Adversarial Networks (GAN) for the prototypical case of designing Elpasolite compositions with low formation energies. By relying on the fully enumerated space of 2 million main group Elpasolites, the precision, coverage and diversity of the generated materials is rigorously assessed. Additionally, a hyperparameter selection scheme for generative models in chemical composition space is developed.

I. INTRODUCTION

Generative machine learning (ML) models are increasingly used for the targeted generation of images,1 video sequences,2 text3 or music.4 This can be achieved by approximately representing the underlying data distribution as inferred from a provided data set and sampling from this distribution. Alternatively, the underlying "construction rules" of a dataset can be learned from examples without taking the underlying distribution into account explicitly. Ultimately, both approaches provide access to novel but realistic examples, which incorporate the decisive features of the training data.

In the context of organic chemistry such ML models have already been successfully applied for the inverse design of novel molecular candidates.5,6 Rather than using an explicit set of rules that link molecular fragments to new molecules, the underlying wealth of construction principles is captured from data in the training phase,7–9 often using existing (public) libraries of organic molecules presented to the model. The generation process then allows overcoming the limitations of the initially presented molecular library10–14 by accessing molecules with a wide variety of chemical functionalities that are still consistent with the training examples.

Beyond merely proposing realistic molecules, molecular design typically additionally demands the focused generation of candidates with desired properties. This can be achieved15 by enhancing generative approaches, i.e. via transfer learning,16–20 semi-supervised learning,21,22 conditional generation,21–26 reinforcement learning27–30 or by carrying out optimization in a well-structured (latent) representation space.31–33

In essence, this molecular design toolbox can equally be applied to the proposition of novel and useful inorganic materials for catalysis, energy conversion or other applications. However, inorganic materials present notable challenges in this regard. Foremost, a suitable (i.e. invertible) material representation that respects the symmetry and the periodicity of a 3-dimensional crystal and is (ideally) independent of the number of atoms in the unit cell, is not trivial to define.34 Moreover, a much larger variety of chemical compositions are possible for inorganic systems. In the prevailing data scarcity, inverse design of inorganic materials has therefore usually focused more on structural prediction in limited composition spaces,35–40 or compositional optimization with fixed structural prototypes.41–43 However, even if trained on a limited subset of structure types, it has been shown that these models are able to generalize to new structure types that were not included in the training process.34

In the above examples, deep generative frameworks employing neural networks (NNs) have proven to be an invaluable tool.44 Notably, these comprise a large zoo of approaches, including variational autoencoders (VAEs),34–36,39,42,45–47 generative adversarial networks (GANs)37,38,41–43,48,49 and reinforcement learning (RL).50,51 Given this wide range of approaches it is a priori difficult to decide which method should be used for a new inverse design task.

While different ML models can be compared objectively by checking their predictive accuracy on an unseen test set for regression or classification tasks, this is much less straightforward for generative models, where each model generates new data independently. Here, one must instead evaluate to what extent the new data covers the underlying distribution of the training set and how well it generalizes to new samples that are unlike the training examples.11–13,52 Furthermore, in the context of targeted materials design, an important question

a) Electronic mail: margraf@fhi.mpg.de
is whether the proposed candidates reliably display the desired properties.

Due to these questions, comparative studies which have been performed for organic molecule generation\cite{Quesada2020,Adolph2020,Zhao2020} and (less frequently) inorganic materials,\cite{Kumamaru2020} are particularly valuable. Aiming to establish deeper insight into the advantages and disadvantages of the variety of generative frameworks for inorganic materials design, we here embark on such a comparative study. Specifically, we use the targeted search for novel, stable compositions within a fixed structural prototype as a suitable benchmark for comparing the performance of three prevalent generative ML approaches (VAE, GAN and RL).

To this end, we rely on a well-established computational data set of Elpasolite structures reported by Faber et al.\cite{Faber2018}, which has previously served as a benchmark for regression models.\cite{Faber2018,Faber2019,Faber2020,Faber2021} This set comprises the fully enumerated space of nearly two million systems, which can be derived by main group elemental exchange on the pristine quarternary Elpasolite mineral AlNaK2F6. Notably, this mineral is part of the quarternary double perovskite prototype ABC2D6, which is of significant technological interest.

Relying on the distribution provided by this fully enumerated chemical composition space, we demonstrate that straightforward realizations of different generative frameworks using simple NN architectures can reliably generate promising material candidates. Noting the plethora of algorithmic ramifications and subtleties in hyperparameter settings in each generative model class, we define general performance metrics with regard to diversity, coverage and fitness of the proposed materials. With this, we hope to establish a well-defined and representative benchmark problem for future assessment and tailoring of generative models.

II. METHODS

A. Dataset and material representation

The prototypical Elpasolite mineral ABC2D6 is reproduced in Figure 1a. The complete Elpasolite material space considered herein then emerges from all possible combinations of main group elements (from H to Bi, 39 overall) on the four lattice sites (A,B,C,D), leading to 1,974,024 possible structures. In ref.\cite{Faber2018} formation energies \( E_{\text{DFT}} \) computed at the PBE level are reported for a subset of 10,590 structures. Additionally, that work also provides accurate estimates \( E_{\text{regression}} \) for the remaining structures, predicted by a kernel ridge regression (KRR) model with a mean absolute error (MAE) of 0.1 eV, see Figure 1e-f. In the following, we use DFT values to train the conditional generative models, while estimated values from the KRR model are used to gauge the quality of generated samples. All formation energies are given per atom.

To represent different Elpasolites, we employ a simplified version of the bag-of-atoms\cite{Schütt2017} representation, closely following Faber et al.\cite{Faber2018}. In this way, each composition is represented by an 8-dimensional vector \( x_{\text{SD}} \). As illustrated in Figure 1b, its entries correspond to the row number \( n_{\text{row}} \) (ranging from 1 to 6) and the main group number \( n_{\text{group}} \) (ranging from 1 to 8) of each of the four sites in the structure. The different absolute ranges of the row and main group entries in the vector are thereby normalized to the common interval (-1,1) by rescaling to

\[
x_{\text{8D,scaled}} = \frac{2 \times x_{\text{SD}} - 1}{n_{\text{row}}/n_{\text{group}}} - 1.
\]

Note that this representation also encodes structures with non-existent elements (e.g. first-row elements of group number 2 to 7) or structures with repeated elements. The generative models can therefore in principle also generate such invalid structures. However, as structure generation is not a time-limiting factor and the
proportion of invalid compositions is generally low, we simply disregard invalid samples, rather than fixing this shortcoming of the compositional representation.

It should further be noted that the A and B sites are equivalent in the Elpasolite structure, so that structure $$\text{ABC}_2\text{D}_6$$ should have the same formation energy as $$\text{BAC}_2\text{D}_6$$. This permutational symmetry is not exploited in the DFT training set of ref. [55]. Indeed, there are only 34 occurrences where both structures are contained (with slightly different DFT formation energies deviating on average by 2.3 meV/atom). As part of our data curation we enlarged the DFT training set by consistently adding all A/B permutations, obtaining an augmented final DFT training set size of 21,112 structures.

Figure 1d depicts the elemental distribution of the final training set. The corresponding DFT formation energies range from $$-3.07$$ eV/atom to $$5.02$$ eV/atom, with a negative value indicating stability with respect to elemental decomposition. In order to conditionally generate samples in given stability ranges, the training data was split into 10 classes, each containing an equal span of 0.81 eV/atom on the $$E_{\text{formation}}$$ scale. Class 1 spans the lowest formation energies and thus most stable Elpasolite compositions, while class 10 spans the highest formation energies and thus least stable compositions. When necessary for conditional training, these classes are encoded into a 10-dimensional one-hot vector, which is concatenated to the structural representation discussed above.

During training, batches are formed by randomly sampling training compositions with a probability that is the inverse of the frequency with which the corresponding energetic class appears in the DFT training set. This weighted sampling allows us to mitigate the class imbalance in the DFT training set, i.e. the fact that e.g. much fewer samples are in class 1 than in class 5 as apparent from Figure 1e. Note that all cost functions below are defined for a single training example, for notational convenience. Extension to batch-based training is straightforward. For practical implementation, a common interface to the data set is provided by a custom dataloader. This code implements the described pre-processing steps and handles the balanced sampling of batches during the training of the generative models. For reference and further benchmarks, the python code is freely available at https://gitlab.mpcdf.mpg.de/fhi-theory/elpasolite_generative_model_assessment.

B. Performance metrics

Using the DFT training set, we build deep generative models for the inverse design of Elpasolite compositions. Ideally, these models should: (i) propose compositions in a targeted energy class with high precision, (ii) yield a high diversity among the proposed compositions, and (iii) display high coverage of the chemical composition space. While the first of these is directly enforced by the conditional generation (and in the RL reward), diversity and coverage are more subtle factors. Indeed, so called mode- or posterior collapse is commonly observed in GANs or VAE models, ultimately resulting in a limited number of samples that the models can generate.

To assess these factors, we exploit the fact that the full Elpasolite composition space is known from the prior work of Faber et al. We thus know the total number $$N_{\text{class}}$$ of compositions in each class (see Figure 1e) and the elemental distributions on the four sites (A, B, C, D) in each class. Based on this, we define the following quantitative performance metrics:

1) **Precision (right class)** measures the model’s ability to generate samples in the desired class. To this end, the fraction of generated compositions that actually fall into the desired class is computed,

$$\text{Precision (right class)} = \frac{N_{\text{class}}^\text{gen}}{N_{\text{gen}}}$$

where $$N_{\text{class}}^\text{gen}$$ is the total number of generated samples and $$N_{\text{gen}}$$ is the number of generated samples which belong to the desired class. This means that high precision is achieved if the model reliably produces samples in the desired class. Note however that this can also be achieved from only a few repetitively produced samples (or even by memorizing the training set).

2) **Precision (neighboring class)** analogously measures the fraction of samples falling into the classes just above or below the requested class:

$$\text{Precision (neigh. class)} = \frac{N_{\text{class}-1}^\text{gen} + N_{\text{class}+1}^\text{gen}}{N_{\text{gen}}}$$

This allows capturing the tails of the predicted formation energy distributions. Since the models are trained on DFT data but evaluated using the approximate KRR energetics for the full Elpasolite space, this metric captures the residual uncertainty in the definition of the energy classes.

3) **Coverage (right class)** measures the fraction of unique compositions in the desired class that could be generated by the model:

$$\text{Coverage (right class)} = \frac{N_{\text{class,un}}^\text{gen}}{N_{\text{class}}}$$

where $$N_{\text{class,un}}^\text{gen}$$ is the number of unique compositions generated in the desired class.

4) **Coverage (neighboring class)** analogously measures the fraction of unique compositions in the classes just above or below the requested class,

$$\text{Coverage (neigh. class)} = \frac{N_{\text{class}-1,un}^\text{gen} + N_{\text{class}+1,un}^\text{gen}}{N_{\text{class}-1,un} + N_{\text{class}+1,un}}$$

5) **JS-Distance** measures how strongly the elemental distribution in the generated examples differs from the corresponding distribution in the full Elpasolite composition space (for a given class). Specifically, we consider probabilities $$p(Z|S)$$ of finding element Z on site S in a
given dataset. To compare two such probability distributions, we use the Jensen-Shannon (JS) distance $d_{JS}$, which provides intuitively interpretable values between 0 and 1 (with 0 indicating that the distributions are identical and 1 indicating no overlap at all). For two elemental distributions $p$ and $q$, this is calculated as

$$d_{JS}(p||q) = \frac{1}{4} \sum_s \sum_Z \left[ \frac{1}{2} p(Z|S) \log_2 \left( \frac{p(Z|S)}{m(Z|S)} \right) + \frac{1}{2} q(Z|S) \log_2 \left( \frac{q(Z|S)}{m(Z|S)} \right) \right]$$

where the first sum runs over all sites $A, B, C, D$, the second sum runs over all elements in the Elpasolite dataset and $m(Z|S) = \frac{1}{2} [p(Z|S) + q(Z|S)]$.

To provide some reference values for the JS-Distance, we compare the elemental distributions corresponding to different classes in the training set and the full Elpasolite space in Table I. This shows that identical classes have low JS-Distances between training and full sets, while different classes have high distances. Unsurprisingly, the distance for identical classes is lower when many examples are included in the training set (i.e. for class 5). Furthermore, the distances change smoothly across the classes, so that the distance between class 1 and class 2 is lower than the distance between class 1 and class 5. Overall, values of 0.25 or lower can be considered as good agreement between the distributions.

| Training set (size) | Full Elpasolite space (size) | JS-Distance |
|---------------------|------------------------------|-------------|
| class 1 (86)        | class 1 (3,757)               | 0.236       |
| class 1 (86)        | class 2 (20,097)              | 0.386       |
| class 1 (86)        | class 5 (907,094)             | 0.648       |
| class 2 (270)       | class 2 (20,097)              | 0.210       |
| class 5 (9,650)     | class 5 (907,094)             | 0.080       |

C. Generative frameworks

Our comparative study specifically covers the three deep generative frameworks shown in Figure 2. The ideas behind the three models differ fundamentally. In RL, an agent constructs the materials in a step-wise procedure. During training, the agent receives feedback about the quality of the samples (i.e., whether they fall into the desired class) and thus improves its decision making policy. In contrast, VAEs and GANs are both trained to generate a low-dimensional latent space into which the training examples are mapped. New materials can then be generated by sampling in the latent space. Both models therefore effectively learn a (conditional) probability distribution of material compositions. However, the way this is achieved differs markedly. On one hand, the VAE consists of an encoder network, which maps training samples to latent space and a decoder network, which reconstructs them back. On the other hand, the GAN uses an adversarial principle. A generator network is trained to translate random samples from latent space to realistic material representations. In parallel, a discriminator is trained to distinguish the ‘fake’ samples from the generator and ‘real’ samples from the training set. Through a feedback process, the generator ultimately learns to fool the discriminator.

To implement the three models we employ fully connected multi-layered artificial NNs. This is the simplest and most general NN architecture, and it is not specifically tailored to the problems at hand. The corresponding generative models therefore serve to establish simple baselines, while specifically tailored architectures (such as graph neural networks) may display even better performance. All models are implemented using pytorch (v1.9.0) and executed with python (v3.8.8).

As a non-linearity after every hidden layer, the commonly used LeakyReLU function is used with a negative slope of 0.2. Based on the model-dependent cost function, the free parameters (weights and biases) of these networks are updated by gradient descent using the Adam optimizer and gradients obtained through backpropagation. While some of the illustrations below are based on one representative model fit, all reported performance metrics are obtained from 50 separate model initializations, unless otherwise stated.

As is always the case for deep learning models, a variety of additional hyperparameters must be set to determine the NN architecture (e.g. the number of hidden layers and neurons per layer) and training procedure (e.g. the learning rate and batch size). We optimize these model- and training hyperparameters with respect to two objective functions, that can be cheaply computed against the training set: 1) The model’s ability to reproduce the elemental distribution occurring in the targeted lowest formation energy class of the training set (practically measured by the JS-Distance) and 2) the model’s ability to generate an overall large number of unique new samples, not contained in the training set. To identify suitable combinations of hyperparameters, a simple random search was performed. See Table S1 for a listing of all searched parameters and their ranges. Figure S4 and Tables S2, S3 and S4 fully detail the procedure.

1. Variational Autoencoder

The generation of high-dimensional data with VAEs proceeds as follows: A low-dimensional latent space vector $z$ is drawn from a predefined probability distribution and mapped to a realistic data point $\tilde{x}$ by a NN $P_\phi$ called the decoder (with trainable parameters $\phi$).

$$\tilde{x} = P_\phi(z)$$
During training, the decoder is paired with a second trainable NN — the encoder \( Q_{\phi} \) (with trainable parameters \( \theta \)). The encoder takes a sample \( x_i \) from the training set and produces a representation of this datapoint in the latent space. Importantly, VAEs use a probabilistic mapping for this purpose: Each input sample \( x_i \) is mapped to a multivariate normal distribution \( N \) in latent space, so that the output of the encoder is a vector of means \( \mu(x_i) \) and variances \( \sigma^2(x_i) \). The corresponding probability distribution \( q_{\phi}(z|x_i) \) in latent space is thus defined as:

\[
q_{\phi}(z|x_i) = N(\mu(x_i), \sigma^2(x_i)) \quad .
\]

Here, the subscript \( \theta \) indicates that this probability distribution depends on the trainable parameters of the encoder \( Q_{\phi} \).

Ideally, the decoder should be able to reconstruct \( x_i \) as accurately as possible, given a random sample \( z_i \) from the distribution \( q_{\phi}(z|x_i) \). Furthermore, the probabilistic mapping of the training data should be smooth and continuous in latent space, to obtain an effective generative model. This is achieved by training the VAE with a combined cost function \( J(\phi, \theta) \), defined as:

\[
J(x_i|\phi, \theta) = \underbrace{(x_i - P_{\phi}(z_i))_2}_\text{reconstruction} + \underbrace{\lambda D_{\text{KL}}(q_{\phi}(z|x_i) \parallel N(0,1))}_\text{regularization} .
\]

Here the reconstruction term is simply the squared deviation between the original and reconstructed input. The regularization term is given by the Kullback-Leibler divergence\(^6\) \( D_{\text{KL}} \), which quantifies the statistical distance between the learned distribution \( q_{\phi}(z|x_i) \) and a multivariate normal distribution with zero means and unit variances.

Minimizing this combined cost function during training requires finding a trade-off between reconstruction and regularization. Intuition about this process can be gained from looking at the two terms separately. If only the reconstruction term was included, the model would be very accurate in reproducing the training set, but its generalization capability towards novel compositions would suffer. Similarly, if the regularization prevails and only the Kullback-Leibler divergence is minimized, the decoder would resort to generating very few unique samples, and instead output an average representation of the training set (since all inputs \( x_i \) would be mapped to the same latent space distribution). This latter problem is known as posterior collapse.\(^6\)\(^,\)\(^7\) To balance between these extremes, the regularization parameter \( \lambda \) in the loss can be adjusted.

As described so far, the VAE would simply propose materials from the full chemical composition space. For a targeted generation of materials with specific properties, we operate the VAE as a conditional model.\(^7\)\(^1\)\(^7\)\(^2\) Such class-conditional learning can easily be incorporated while keeping a similar training procedure. To this end, both the encoder and the decoder receive the class-label of each training sample, which is concatenated with their respective inputs. After training, the decoder can then generate new samples from a specific class when provided with a random latent space vector and the desired class-label.

Following the hyperparameter search described above, the final VAE models used below are constructed as follows: The class-conditional input \( x_{\text{cond}} \) is encoded to a 8-dimensional latent space, and decoded into the scaled 8-dimensional representation of the composition. Both the encoder and decoder NNs have 2 hidden layers with 256 nodes. Layer normalization\(^7\)\(^3\) is employed. The final output function of the decoder is a hyperbolic tangent that returns values between -1 and 1. All models were trained for 10,000 network updates on batches of 500 samples each, using the Adam optimizer with a learning rate of 0.001. The regularization parameter \( \lambda \) in the loss is fixed at 0.1.\(^6\)\(^2\)
2. Generative Adversarial Network

Similar to the VAE, the GAN\(^74\) framework uses a generator model \(G_\theta\) (with trainable parameters \(\theta\)), which is trained to generate realistic synthetic samples \(\hat{x} = G_\theta(z)\) from a random latent space vector \(z\). Training of \(G_\theta\) thereby follows an adversarial approach, which can be understood as a contest between \(G_\theta\) and a discriminator model \(D_\phi\) (with trainable parameters \(\phi\)). The role of \(D_\phi\) is to discriminate between real samples from the training set and the synthetic samples created by \(G_\theta\). The feedback from \(D_\phi\) is in turn used to improve the ability of \(G_\theta\) to generate increasingly realistic samples, essentially trying to fool \(D_\phi\) into misclassification. Over the course of this competition, both models successively improve.

While this idea is highly intuitive, successfully training a GAN can be challenging. For example, a commonly observed problem\(^41\) is so-called mode collapse – where \(G_\theta\) resorts to fooling \(D_\phi\) by repetitively generating the same highly convincing samples. Intensive research has therefore been devoted to improving the training procedure of GANs. This has led to a series of different proposed cost functions and modified NN architectures that improve training stability\(^75\) or generative performance.\(^76\) We here rely on the Wasserstein GAN\(^77\) with a gradient-penalty\(^78\) term (WGAN-GP). In this framework, \(D_\phi\) is usually referred to as a “critic”, and returns a scalar value that represents the sample quality, instead of a mere binary classification of data as real or fake.

As for the VAE, \(D_\phi\) and \(G_\theta\) are represented by fully connected NNs herein, whose parameters are optimized during the adversarial training by gradient descent. Due to the adversarial nature of the GAN, two separate cost functions \(J_D\) and \(J_G\) are used for the discriminator and generator, respectively. The former is defined as

\[
J_D(x_i|\phi, \theta) = -D_\phi(x_i) + D_\phi(G_\theta(z_i)) + \lambda P(x_i, G_\theta(z_i)) \tag{10}
\]

Here, the first and second terms reward high scores for real and low scores for fake samples, respectively. Sparing the detail,\(^77\) the gradient-penalty term represents a practical solution to enforce a well-behaved critic function. Its evaluation depends on linearly interpolated data points between the real and fake data points. The recommended penalty coefficient \(\lambda = 10\) is used throughout.\(^78\)

Meanwhile, the cost function of the generator is simply

\[
J_G(\phi, \theta) = -D_\phi(G_\theta(z)) \tag{11}
\]

\(i.e.\) it tries to maximize the score that \(D_\phi\) assigns to the generated samples.

In analogy to the VAE, our model operates as a conditional GAN,\(^79\)\(^80\) with sample class-conditional information provided to \(G_\theta\) and \(D_\phi\) alongside the input during training. After training, this again allows us to query the generator network for samples from the desired class.

Note that to achieve stable training, the parameters of \(D_\phi\) are updated five times as frequently as \(\theta_G.\)^\(^78\) Based on the hyperparameter optimization, the final models use one hidden layer with 512 nodes each for \(G_\theta\) and \(D_\phi\). A 8-dimensional latent space vector is passed to \(G_\theta\) and decoded to the structural representation \(\phi_{SD, scaled}\), again employing a hyperbolic tangent output layer. The networks are trained for 200,000 update steps on batches of 100 samples with a learning rate of 0.00001.

3. Reinforcement Learning

In RL an agent makes goal-oriented decisions in a successively evolving environment. To this end, a reward function is defined, which provides feedback about whether the actions of the agent lead to the desired outcomes. Because of this focus on acting in a complex environment, the classic fields of application of RL are the control of robots or other autonomous agents (e.g. in games). Nonetheless, RL has also been applied for the design of molecules and materials.\(^28\)\(^30\)\(^81\)\(^84\)

In this case, the agent sequentially adds atoms or functional groups to a system. The main challenge associated with this is that the reward (\(e.g.\) as a measure of stability or a desired property) can only be determined for the complete system. It is therefore not straightforward to judge the quality of actions taken early on in the generation process (the so-called ‘credit assignment’ or ‘sparse reward’ problem). In this sense, designing a material is similar to a game of chess, where the reward (win/loose/draw) can only be determined at the end of the game.

In context of this paper, a RL agent makes sequential decisions about the elemental composition of an Elpasolite, with the goal of the final formation energy being in the desired range. This step-wise process is illustrated for the composition of KMgXe\(_2\)O\(_6\) (where the scaling of eq. (1) is omitted for clarity):

\[
\begin{align*}
\text{state 0} & \quad \text{action 0} \quad \begin{pmatrix} \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \end{pmatrix} \\
\text{state 1} & \quad \text{action 1} \quad \begin{pmatrix} 4 & \cdot & \cdot & \cdot & \cdot \end{pmatrix} \\
\vdots & \quad \vdots \\
\text{state 7} & \quad \text{action 7} \quad \begin{pmatrix} 4 & 1 & 3 & 2 & 5 & 8 & 2 & 6 \end{pmatrix}
\end{align*}
\]

The agent thus starts from an empty composition vector and chooses the row of the element in position \(A\). This leads to an updated composition vector and a new decision to make, namely what the group of element \(A\) should be. In RL terminology, the composition vector defines the state \(s\) of the environment and each decision the agent makes is an action \(a\), which leads to a new state.
The generation process can thus be described as a sequence of state-action pairs \((s_0, a_0, s_1, a_1, \ldots, s_T, a_T, s_s)\), with a final state \(s_s\). Such a structure generation sequence is called an episode \((e)\) in the following.

During training, the agent receives a reward \(r\) after each action, which is based on the formation energy of the new state \(s'\), according to the classes introduced above. Specifically,

\[
r(s, a) = \begin{cases} 
0 & \text{if } s' \text{ non-final} \\
1 & \text{if } s' \text{ final and in desired class} \\
-1 & \text{if } s' \text{ final and not in desired class}
\end{cases}
\]

The key component of any RL model is the policy \(\pi\), which is the algorithm used to decide on the next action of the agent based on the current state of the environment. Importantly, to overcome the sparse reward problem the policy cannot simply maximize the immediate reward for the current action. Indeed, this would be completely ineffective for the task at hand, since the reward for most actions is zero. Instead, the ideal policy maximizes the sum of the current and all future rewards. This is directly analogous to conventional Q-learning, where the Q-values are updated based on the immediate reward \(r(s, a)\) and other (randomly initialized) Q-values in the table. This works because the information about the reward \(r(s, a)\) obtained at the end of each episode propagates to earlier state-action pairs as the training progresses.

To train this network, the cost function \(J(\theta)\) defines a least-squares regression with eq. 13 as the target value:

\[
J(e, \theta) = \sum_{s,a \in e} \left[ Q_{\theta}(s,a) - r(s,a) + \gamma \max_{a'} Q_{\theta}(s',a') \right]^2.
\]

This function sums over the state-action pairs \((s, a)\) of a given training episode \(e\). Here, a peculiarity of DQNs compared to other deep learning methods is that both the prediction and the target value in the loss depend on \(Q_{\theta}\).

It should be noted that the above description refers to the ‘batch’ or ‘offline’ RL setting, which assumes a fixed training set of episodes which are collected independently from the model itself. In principle, ‘active’ or ‘online’ learning training schemes are also possible, which use the model itself to explore the action space and thus construct the training set. In the present context this would require performing DFT calculations on-the-fly during training.

The interdependence of predictions and targets in eq. 15 causes some stability problems when training DQN models. To mitigate this we use the Double Deep Q-Learning approach, in which two different versions of \(Q_{\theta}\) are used during training. The first of these is the policy network which is used to make the predictions in the first term of eq. 15 and whose weights are updated at every training step. The second is the target network, which is used to obtain the Q-values in the target term of Eq. 15 and whose parameters are synchronized with the policy network in regular intervals. This breaks the direct dependence of prediction and target in the loss function and thus stabilizes training.

Once the training is completed, \(Q_{\theta}\) can be used to estimate the long-term reward of any action \(a\) given a state \(s\). To construct a generative model based on this, we must finally specify the policy \(\pi\), i.e. the algorithm according to which the next action \(a\) is selected. Here, a greedy algorithm which simply selects the action with the highest \(Q_{\theta}(s,a)\) would be possible, but this would obviously not lead to a diverse sample of materials. For a better balance between exploration and exploitation, we therefore sample actions according to probabilities \(p(s,a)\), which
are obtained from $Q_\theta(s,a)$ with the softmax function

$$p(s,a) = \frac{\exp (\beta Q_\theta(s,a))}{\sum_{a'} \exp (\beta Q_\theta(s,a'))}.$$  \hspace{1cm} (16)

Here, the sum in the denominator is over all possible actions (including $a$) and the hyperparameter $\beta$ behaves like an inverse temperature. These state-action probabilities can thus be tuned from uniform random sampling (for $\beta = 0$) to a fully greedy policy (for $\beta \to \infty$). In practice, we found a value of $\beta = 5$ to be optimal (see Figure S7 in the Supporting Information).

From the hyperparameter search, we obtained a network architecture with five hidden layers with 512 nodes each (with layer normalization \cite{Glorot10}). The RL model was trained for 800,000 network updates with a batch size of 250, and a learning rate of 0.00001. Following common practice in the DQN literature, a discount factor $\gamma = 0.999$ is used and the target network is synchronized every ten steps.

III. RESULTS AND DISCUSSION

A. Targeted generation in a minority class

As discussed above, the goal of inverse materials design is to generate promising sample candidates from a large design space. Importantly, these promising candidates are typically exceedingly rare. In the Elpasolite dataset, this challenge can be emulated by attempting the targeted generation of compositions in the lowest energy class 1. As shown in Figure 1c, this class is one of the least populated and contains only 86 samples, corresponding to 0.04% of the total training set. This imbalance is equally present in the full Elpasolite space, with only 3,757 compositions amounting to 0.2% of all possible ones falling into this minority class. For comparison, the neighboring class 2 already amounts to 20,097 compositions (1.0% of the full Elpasolite space).

Figure 3 illustrates the performance of a representative GAN model for the first 250,000 generated valid samples, when conditioned on the minority class 1. Novel discoveries are further discerned by the classes they belong to (in gray).

![Figure 3](image)

This behavior can be understood from the fact that the generative models essentially learn a conditional probability distribution. When sampling from this distribution, initially mostly unique, high-probability samples are drawn. However, upon continued sampling there is an increased likelihood that lower probability samples are repeated. This explains the fact that the number of unique class 2 samples eventually overtakes the number of unique class 1 samples. While they are considered to be less probable candidates by the model, there are simply many more unique class 2 compositions to discover in the dataset.

For all models generated in this work, the curve for the number of novel discoveries eventually flattens. In other words, the models display a limited capacity for generating unique samples, which is unsurprising since the number of possible Elpasolites is also limited. Notably, the rediscoveries constitute a low share throughout, so that high-probability samples are repeated.

For all models generated in this work, the curve for the number of novel discoveries eventually flattens. In other words, the models display a limited capacity for generating unique samples, which is unsurprising since the number of possible Elpasolites is also limited. Notably, the rediscoveries constitute a low share throughout, so that high-probability samples are repeated.
FIG. 4. Formation energy distribution (in 0.2 eV/atom bins) of the unique class 1 compositions proposed by the three generative models (VAE, GAN, RL) over 10,000 valid samples. Additionally shown is the corresponding distribution over the entire training set of 21,112 compositions.

This plot also illustrates from how little class 1 data the models are able learn. On this scale, the 87 class 1 samples in the training set are not even visible. Nonetheless, all three models generate over one thousand unique new compositions in this class. It can also be seen how the formation energy distributions decay into the neighboring class, generating a negligible amount of samples in higher energy classes.

TABLE II. Performance metrics for the three generative models conditioned to generate minority class 1 compositions after 10,000 valid samples. Shown are averages over 50 model fits together with the standard deviation.

|                  | RL    | VAE   | GAN   |
|------------------|-------|-------|-------|
| Precision (right class) | $69 \pm 8\%$ | $83 \pm 1\%$ | $66 \pm 2\%$ |
| Precision (neigh. class) | $29 \pm 7\%$ | $16 \pm 1\%$ | $28 \pm 1\%$ |
| Coverage (right class)  | $53 \pm 3\%$ | $54 \pm 2\%$ | $62 \pm 1\%$ |
| Coverage (neigh. class) | $4 \pm 1\%$ | $5 \pm 0\%$ | $10 \pm 1\%$ |
| JS-Distance        | $0.20 \pm 0.03$ | $0.16 \pm 0.01$ | $0.16 \pm 0.01$ |

For a more quantitative comparison of the generative frameworks considered herein, average performance metrics obtained from 50 model initializations are summarized in Table II. These metrics confirm the relatively high precision with which all three models generate compositions in the target minority class 1. Given the discrete class boundaries and the larger size of the neighboring class 2, it is not surprising that the generated distributions tail into it. Essentially all unique samples (>94%) produced by the models therefore jointly fall into these two classes. All three models can thus capture the underlying building principles that lead to stable Elpasolite compositions, while covering at least half of the full class 1 composition space.

Notably, both VAE and GAN models display a very low standard deviation for these metrics, indicating that essentially every model fit robustly yields this high precision. There also seems to be a slight trade-off between precision and coverage between VAE and GAN, with the GAN yielding higher coverage but lower precision. In contrast, the RL models display somewhat higher variance.

As indicated by the JS-Distance, all models accurately reproduce the elemental distribution of the targeted class, with comparatively small values of 0.16 for VAE and GAN and a slightly larger distance of 0.2 for RL. Notably, these distances are actually closer to the target distribution than the training set (0.236, see Table I). A more detailed illustration of the elemental distributions is given in Figure 5. A prominent feature of all elemental distributions is the overwhelmingly dominant occupation of site D by fluorine, likely due to its high electronegativity and reactivity. Other sites also exhibit...
preferences for certain element types e.g. alkali metals and alkaline earths on site C, although this tendency is less pronounced. The generative models thus develop a useful chemical intuition for how to construct stable Elpasolites.

B. Targeted generation in the majority class

Complementary to the generation in class 1, we can also consider conditional generation in the majority class 5. This is particularly interesting with respect to the capacity of the generative models. With 907,094 compositions, class 5 spans 46% of the entire Elpasolite composition space and is thus far the dominant majority class. Taken together, its neighboring classes 4 and 6 account for another 39.9% of the entire Elpasolite design space. The dominance of class 5 is also reflected in the DFT training set, which contains 9,650 such compositions, a much larger number than the 86 compositions of class 1. However, compared to the total size of the class 5 composition space, this only amounts to 1%. In relative terms, the coverage of training data in the class 5 composition space is thus even worse than for the minority class 1 (2%, see above).

![Figure 6](image)

**FIG. 6.** Same as Fig. 3, but for a GAN model conditioned on the majority class 5. Note the much larger number of samples required to saturate the number of unique novel discoveries generated, reflecting the much larger composition space covered by class 5.

Analyzing the generation run of a representative GAN model for the majority class (Figure 6) reveals largely analogous behavior to the minority class case shown in Figure 3. The rediscovery rate of training examples is again low throughout the run. With continued sampling, the number of repetitions rises steeply, overtaking the novel generations after around 2.5 million samples. The number of novel discoveries is fully converged after 25 million valid samples. Comparable findings are obtained for VAE and RL (see SI) so that we evaluate the performance metrics for the majority class after 25 million valid samples in the following.

|        | RL | VAE | GAN |
|--------|----|-----|-----|
| Precision (right class) | 87 ± 7% | 58 ± 1% | 54 ± 1% |
| Coverage (neigh. classes) | 13 ± 1% | 34 ± 1% | 36 ± 1% |
| Coverage (right class) | 44 ± 15% | 86 ± 9% | 89 ± 2% |
| Coverage (neigh. class) | 9 ± 4% | 82 ± 11% | 85 ± 3% |
| JS-Distance | 0.49 ± 0.14 | 0.21 ± 0.01 | 0.16 ± 0.01 |

Table III collects the respective metrics. Both GAN and VAE achieve average class coverages of 86% or higher, with precisions between 50 and 60%. In contrast, the RL models achieve significantly higher precision (87% on average), but lower coverages (typically <50%). We observe such trade-offs between precision and coverage frequently, and they should not be strictly attributed to a fundamental difference between the RL and VAE/GAN frameworks. For example, the RL agents could in principle achieve higher coverage at the expense of lower precision by decreasing the inverse temperature $\beta$ in Eq. 16. Similarly, different network architectures in the random hyperparameter search form a Pareto front with respect to the number of unique discoveries (coverage) and the JS-Distance to the training set (precision) (see Figure S4).

We do see fundamental differences between RL and VAE/GAN based models with respect to the elemental distributions, however. This can be seen from the substantially larger JS-Distance of the RL generated data with respect to the target distribution (on average 0.49 for RL vs. 0.21/0.16 for VAE/GAN). By analyzing the corresponding elemental distributions in detail (see Fig. S3), we find that VAE and GAN sample elements relatively evenly across the periodic table, in fairly good agreement with the reference data. In contrast, RL models display a much more selective elemental distribution. Indeed, we find that differently initialized RL models converge to different local minima, each with their own characteristic elemental fingerprints. This indicates that the distribution learners (GAN and VAE) are better suited for representing large composition spaces than agent-based RL models. However, the latter may also be improved in this respect by using more sophisticated reward functions.

C. Influence of training data

The generative frameworks discussed herein thus display promising performance for inverse materials design, in particular for minority classes. However, this capability does not come for free, as it relies on an extensive training set of labeled compositions (for which DFT
calculations are required). The size of this training set amounts to \( \sim 1.0\% \) of the full Elpasolite design space that spans nearly 2 million compositions. At first sight, this seems like impressively little data.

However, it should be noted that in a materials discovery setting additional DFT calculations would be required to verify the formation energies of all unique generated samples. Taking the example of the GAN, this would mean around 4,500 additional calculations, leading to the discovery of 2329 class 1 materials (and 2009 class 2 materials). For comparison, to discover a similar number of class 1 materials through brute-force random screening would required a factor of 50 more DFT calculations.

Since the bulk of the DFT calculations for the generative models is spent on training data generation, decreasing the size of the training sets would thus be an appealing route to further boost the computational efficiency of this approach.

To explore this possibility, we assessed the performance metrics for minority class generation when training with randomly selected subsets of the full training set, keeping all other settings fixed. As in the original training set, we again account for the permutational symmetry of the A and B sites in each subset, so that the total number of DFT calculations required for each subset is \( N_{train} \). These results are summarized in Figure 7. For comparison, we also include the results for models trained with the original training set reported by Faber et al. (without permutational symmetry).

This reveals that our data augmentation approach works very well. Models trained on 10,000 data points (including permutations) reach the same performance as models trained on the original set, while reducing the number of required DFT calculations by a factor of two. In general, we find that both precision and (particularly) coverage for the target class decrease when the training set decreases. Similarly, the JS-Distance between generated and target distribution increases. Here, the RL models are somewhat less sensitive, retaining higher coverages for the smallest training sets.

Nonetheless, even reduced training sets yield models capable of discovering a significant number of new materials. Again taking the GAN as an example, even at the smallest training set size of 1000 compositions, on average 11\% (> 410 samples) of all possible class 1 compositions are discovered, easily surpassing the 86 examples found in the original training set. Given a precision of around 50\% for this class, this means that roughly 1000 further DFT calculations would be required to obtain the formation energies of all generated samples. With random sampling, finding the same number of class 1 materials would require 220,000 DFT calculations (i.e. a factor 100 more).

Importantly, these estimates are somewhat conservative. For one, all model hyperparameters were kept fixed, although small datasets often lead to different optimal network architectures. Furthermore, random sampling is a fairly strong baseline in this example due to the limited size of the Elpasolite composition space. In reality, chemical space is practically unlimited and brute-force random search is not a viable strategy at all.

**IV. CONCLUSIONS**

We herein assessed the performance of three deep generative machine learning frameworks (VAE, GAN and RL) for the exploration of a large chemical composition space. To this end, we relied on the fully enumerated space of Elpasolite minerals (ABC\(_2\)D\(_6\)) as a target, which allowed us to quantitatively assess the generative models in terms of precision, coverage and elemental distributions. In our view, this is highly valuable as evaluating and comparing generative ML models is notoriously difficult.

Despite being built from simple neural network architectures, all studied models are capable of reliably generating candidates within the desired formation energy classes. This shows that reasonable model hyperparam-
eters could be determined by an automated procedure, while accommodating a reasonable trade-off between coverage and precision in conditional generation.

Nonetheless, there are a number of notable differences between the approaches. The RL models showed greater robustness towards small datasets, but also greater variance between differently initialized models (i.e., a tendency to converge to distinct local minima). In contrast, the VAE and GAN approaches usually produced models that more faithfully reproduced the target elemental distributions, as quantified by the JS-Distance.

From a technical perspective, the RL models were trained to generate samples from a given class. We therefore had to train completely new models (with a different reward function) for majority class generation. Additional details for hybrid models can be found in the paper.

In contrast, the VAE and GAN models directly capture class conditional probability distributions, so that a single model can generate all classes. Among these, the VAE showed slightly higher precision and was (subjectively) somewhat easier to train than the GAN. From this perspective, the VAE appears to us to be overall best suited for large scale materials discovery, though this certainly depends on the application.

Indeed, high coverage and faithful reproduction of the target distribution are not always necessary. If the objective is simply to generate a limited number of high-quality samples, a goal-oriented RL model may be the better choice. RL furthermore has the advantage that the reward function can easily be modified to accommodate more complex design targets.

We hope that the present work provides a sound basis for the development of deep generative models for materials discovery in chemical composition space. To this end, the establishment of quantitative performance metrics is of paramount importance. In future work, we aim to generalize the current models beyond a single crystal prototype.

Acknowledgements: E.L. and K.R. gratefully acknowledge funding by the Deutsche Forschungsgemeinschaft (DFG, German Research Foundation) under the priority programme SPP 2196. H.T. was supported by DFG under the priority programme SPP 2080 DynaKat.

Additional Information: * These authors contributed equally.

Supporting Information Available: Additional generation plots for VAE and RL. Elemental distributions for majority class generation. Additional details for hyperparameter search procedure. GAN generation plots with smaller training sets.

1Brock, A.; Donahue, J.; Simonyan, K. Large Scale GAN Training for High Fidelity Natural Image Synthesis. arXiv preprint arXiv:1809.11096 2019.
2Clark, A.; Donahue, J.; Simonyan, K. Adversarial Video Generation on Complex Datasets. arXiv preprint arXiv:1907.06571 2019.
3Nie, W.; Narodytska, N.; Patel, A. RelGAN: Relational Generative Adversarial Networks for Text Generation. International Conference on Learning Representations. 2019.
4Dong, H.-W.; Hsiao, W.-Y.; Yang, L.-C.; Yang, Y.-H. MuseGAN: Multi-track Sequential Generative Adversarial Networks for Symbolic Music Generation and Accompaniment. Proceedings of the AAAI Conf. on Artif. Intell. 2018.
5Merk, D.; Friedrich, L.; Grisoni, F.; Schneider, G. De Novo Design of Bioactive Small Molecules by Artificial Intelligence. Mol. Inform. 2018, 37, 1700153.
6Zhavoronkov, A. et al. Deep learning enables rapid identification of potent DTRI kinase inhibitors. Nat. Biotechnol. 2019, 37, 1038–1040.
7Sanchez-Lengeling, B.; Aspuru-Guzik, A. Inverse molecular design using machine learning: Generative models for matter engineering. Science 2018, 361, 360–365.
8Elton, D. C.; Boukouvalas, Z.; Fuge, M. D.; Chung, P. W. Deep learning for molecular design—a review of the state of the art. Mol. Syst. Des. Eng. 2019, 4, 828–849.
9Merz, K. M.; De Fabritiis, G.; Wei, G.-W. Generative Models for Molecular Design. J. Chem. Inf. Model. 2020, 60, 5635–5636.
10Aris-Pous, J.; Blaschke, T.; Ulander, S.; Reymond, J.-L.; Chen, H.; Engkvist, O. Exploring the GDB-13 chemical space using deep generative models. J. Cheminform. 2019, 11, 20.
11Brown, N.; Fiscato, M.; Segler, M. H.; Vacher, A. C. Guacamol: Benchmarking Models for de Novo Molecular Design. J. Chem. Inf. Model. 2019, 59, 1096–1108.
12Polykovskiy, D. et al. Molecular Sets (MOSES): A Benchmarking Platform for Molecular Generation Models. Front. Pharmacol. 2020, 11, 1931.
13Zhang, J.; Mercado, R.; Engkvist, O.; Chen, H. Comparative Study of Deep Generative Models on Chemical Space Coverage. J. Chem. Inf. Model. 2021, 61, 2572–2581.
14Flam-Shepherd, D.; Zhu, K.; Aspuru-Guzik, A. Keeping it Simple: Language Models can learn Complex Molecular Distributions. arXiv preprint arXiv:2112.03041 2021.
15Sousa, T.; Correia, J.; Pereira, V.; Rocha, M. Generative Deep Learning for Targeted Compound Design. J. Chem. Inf. Model. 2021, 61, 5343–5361.
16Gupta, A.; Muller, A. T.; Huisman, B. J. H.; Fuchs, J. A.; Schneider, P.; Schneider, G. Generative Recurrent Networks for De Novo Drug Design. Mol. Inform. 2018, 37, 1700111.
17Segler, M. H. S.; Kogej, T.; Tyrczyn, C.; Waller, M. P. Generating Focused Molecule Libraries for Drug Discovery with Recurrent Neural Networks. ACS Cent. Sci. 2018, 4, 120–131.
18Amabilino, S.; Pogány, P.; Pickett, S. D.; Green, D. V. S. Guidelines for Recurrent Neural Network Transfer Learning-Based Molecular Generation of Focused Libraries. J. Chem. Inf. Model. 2020, 60, 5069–5713.
19Cai, C.; Wang, S.; Xu, Y.; Zhang, W.; Tang, K.; Ouyang, Q.; Lai, L.; Pei, J. Transfer Learning for Drug Discovery. J. Med. Chem. 2020, 63, 8683–8694, PMID: 32672961.
20Moret, M.; Friedlich, L.; Grisoni, F.; Merk, D.; Schneider, G. Generative molecular design in low data regimes. Nat. Mach. Intell. 2020, 2, 171–180.
21Kang, S.; Cho, K. Conditional Molecular Design with Deep Generative Models. J. Chem. Inf. Model. 2019, 59, 43–52.
22Lim, J.; Hwang, S.-Y.; Moon, S.; Kim, S.; Kim, W. Y. Scaffold-based molecular design with a graphic generative model. Chem. Sci. 2020, 11, 1153–1164.
23Li, Y.; Zhang, L.; Liu, Z. Multi-objective de novo drug design with conditional graph generative model. J. Cheminform. 2018, 10, 33.
24Kotsias, P.-C.; Aris-Pous, J.; Chen, H.; Engkvist, O.; Tyr-
chan, C.; Bjerrum, E. J. Direct steering of de novo molecular generation with descriptor conditional recurrent neuronal networks. *Nat. Mach. Intell.* 2020, 2, 254–265.

Gebauer, N. W. A.; Gastegger, M.; Hessmann, S. S. P.; Müller, K.-R.; Schütt, K. T. Inverse design of 3d molecular structures with conditional generative neural networks. *Nat. Commun.* 2022, 13, 973.

Lim, J.; Ryu, S.; Kim, J. W.; Kim, W. Y. Molecular generative model based on conditional variational autoencoder for de novo molecular design. *J. Cheminform.* 2018, 10, 1–9.

Guimaraes, G. L.; Sanchez-Lengeling, B.; Outeiral, C.; Farias, P. L. C.; Aspuru-Guzik, A. Objective-Reinforced Generative Artificial Networks (ORGAN) for Sequence Generation Models. *arXiv preprint arXiv:1705.10843* 2017.

Popova, M.; Isayev, O.; Troppa, A. Deep reinforcement learning for de novo drug design. *Sci. Adv.* 2018, 4, 1–15.

Oliveverona, M.; Blaschke, T.; Engkvist, O.; Chen, H. Molecular de novo design through deep reinforcement learning. *J. Cheminform.* 2017, 9, 48.

Blaschke, T.; Arús-Pous, J.; Chen, H.; Margreitter, C.; Tyrchan, C.; Engkvist, O.; Papadopoulos, K.; Patronov, A. REINVENT 2.0: An AI Tool for de Novo Drug Design. *J. Chem. Inf. Model.* 2020, 60, 5918–5922.

Gómez-Bombarelli, R.; Wei, J. N.; Duvenaud, D.; Hernández-Lobato, J. M.; Sánchez-Lengeling, B.; Sheberla, D.; Aguilera-Iparraguirre, J.; Hirzel, T. D.; Adams, R. P.; Aspuru-Guzik, A. Automatic chemical design using a data-driven continuous representation of molecules. *ACS Cent. Sci.* 2018, 4, 268–276.

Jin, W.; Barzilay, R.; Jaakkola, T. Junction Tree Variational Autoencoder for Molecular Graph Generation. *arXiv preprint arXiv:1802.04364* 2019.

Griffiths, R.-R.; Hernández-Lobato, J. M. Constrained Bayesian optimization for automatic chemical design using variational autoencoders. *Chem. Sci.* 2020, 11, 577–586.

Court, C. J.; Yildirim, B.; Jain, A.; Cole, J. M. 3-D inorganic crystal structure generation and property prediction via representation learning. *J. Chem. Inf. Model.* 2020, 60, 4518–4535.

Noh, J.; Kim, J.; Stein, H. S.; Sanchez-Lengeling, B.; Gregoire, J. M.; Aspuru-Guzik, A.; Jung, Y. Inverse Design of Solid-State Materials via a Continuous Representation. *Matter* 2019, 1, 1370–1384.

Lee, I.-H.; Chang, K. J. Crystal structure prediction in a continuous representational space. *Comput. Mater. Sci.* 2021, 194, 110436.

Liang, T.; Fortunato, N. M.; Opahle, I.; Zhang, Y.; Samathrakis, I.; Shen, C.; Gutleisch, O.; Zhang, H. Constrained crystals deep convolutional generative adversarial network for the inverse design of crystal structures. *NPJ Comput. Mater.* 2021, 7, 1–7.

Kim, S.; Noh, J.; Gu, G. H.; Aspuru-Guzik, A.; Jung, Y. Generative Adversarial Networks for Crystal Structure Prediction. *ACS Cent. Sci.* 2020, 6, 1412–1420.

Yao, Z.; Sánchez-Lengeling, B.; Bobbitt, N. S.; Bucior, B. J.; Kumar, S. G. H.; Collins, S. P.; Burns, T.; Woo, T. K.; Farha, O. K.; Snurr, R. Q.; Aspuru-Guzik, A. Inverse design of nanoporous crystalline reticular materials with deep generative models. *Nat. Mach. Intell.* 2021, 3, 76–86.

Dong, Y.; Li, D.; Zhang, C.; Wu, C.; Wang, H.; Xin, M.; Cheng, J.; Lin, J. Inverse design of two-dimensional graphene/I-BN hybrids by a regressional and conditional GAN. *Carbon* 2020, 169, 9–16.

Dan, Y.; Zhao, Y.; Li, X.; Li, S.; Hu, M.; Hu, J. Generative adversarial networks (GAN) based efficient sampling of chemical composition space for inverse design of inorganic materials. *NPJ Comput. Mater.* 2020, 6, 84.

Patnayak, Y.; Juneja, K. S.; Varma, G.; Ehar, M.; Priyakumar, U. D. Deep learning enabled inorganic material generator. *Phys. Chem. Chem. Phys.* 2020, 22, 26935–26943.

Zhao, Y.; Al-Fahdi, M.; Hu, M.; Siriwardane, E. M.; Song, Y.; Nasiri, A.; Hu, J. High-throughput discovery of novel cubic crystal materials using deep generative neural networks. *Adv. Sci.* 2021, 8, 2100566.

Chen, L.; Zhang, W.; Nie, Z.; Li, S.; Pan, F. Generative models for inverse design of inorganic solid materials. *J. Mater. Inf.* 2021, 1, 4.

Hoffmann, J.; Maestri, L.; Sawada, Y.; Tang, J.; Sellier, J. M.; Bengo, Y. Data-driven approach to encoding and decoding 3d crystal structures. *arXiv preprint arXiv:1909.09849* 2019.

Ren, Z. et al. An invertible crystallographic representation for general inverse design of inorganic crystals with targeted properties. *Matter* 2022, 5, 314–335.

Korolev, V.; Mitrofanov, A.; Eliseev, A.; Tkachenko, V. Machine learning-assisted search for functional materials over extended chemical space. *Mater. Horiz.* 2020, 7, 2710–2718.

Noura, A.; Sokolovska, N.; Crivello, J.-C. CrystalGAN: Learning to Discover Crystallographic Structures with Generative Adversarial Networks. *arXiv preprint arXiv:1810.11203* 2019.

Dehmetha, A.; Krajeski, A. M.; Sun, H.; Lin, S.; Ahn, M.; Li, W.; Priya, S.; Singh, J.; Shang, S.; Beese, A. M.; et al. Generative deep learning as a tool for inverse design of high-entropy refractory alloys. *arXiv preprint arXiv:2108.12019* 2021.

Sajedian, I.; Badloe, T.; Rho, J. Optimisation of colour generation from dielectric nanostructures using reinforcement learning. *Opt. Express* 2020, 28, 5874–5884.

Luo, C.; Ning, S.; Liu, Z.; Zhuang, Z. Interactive inverse design of layered phononic crystals based on reinforcement learning. *Extrem. Mech. Lett.* 2020, 36, 100651.

Ren, Z.; Van Rompaey, D.; Wegner, J. K.; Hochreiter, S.; Klambauer, G. On failure modes in molecule generation and optimization. *Drug Discov. Today: Technol.* 2019, 32-33, 55–63.

Skinnider, M. A.; Stacey, R. G.; Wishart, D. S.; Foster, L. J. Chemical language models enable navigation in sparsely populated chemical space. *Nat. Mach. Intell.* 2021, 3, 759–770.

Sawada, Y.; Morikawa, K.; Fujii, M. Study of deep generative models for inorganic chemical compositions. *arXiv preprint arXiv:1910.11499* 2019.

Faber, F. A.; Lindmaa, A.; Von Lilienfeld, O. A.; Armiento, R. Machine Learning Energies of 2 Million Elpasolite (ABC2D6) Crystals. *Phys. Rev. Lett.* 2016, 117, 135502.

Zheng, X.; Zheng, P.; Zheng, L.; Zhang, Y.; Zhang, R. Z. Multi-channel convolutional neural networks for materials properties prediction. *Comput. Mater. Sci.* 2020, 173, 109436.

Tshitoyan, V.; Dagdelen, J.; Weston, L.; Dunn, A.; Rong, Z.; Konovona, O.; Persson, K. A.; Ceder, G.; Jain, A. Unsupervised word embeddings capture latent knowledge from materials science literature. *Nature* 2019, 572, 95–98.

Huo, H.; Rupp, M. Unified Representation of Molecules and Crystals for Machine Learning. *arXiv preprint arXiv:1704.06439* 2017.

Willatt, M. J.; Musil, F.; Ceriotti, M. Feature optimization for atomistic machine learning yields a data-driven construction of the periodic table of the elements. *Phys. Chem. Chem. Phys.* 2018, 20, 29661–29666.

Zhou, Q.; Tang, P.; Liu, S.; Pan, J.; Yan, Q.; Zhang, S. C. Learning atoms for materials discovery. *Proc. Natl. Acad. Sci. U. S. A.* 2018, 115, E6411–E6417.

Mets, L.; Poole, B.; Pfau, D.; Sohl-Dickstein, J. Unrolled Generative Adversarial Networks. *arXiv preprint arXiv:1611.02163* 2017.

Yan, C.; Wang, S.; Yang, J.; Xu, T.; Huang, J. Re-balancing variational autoencoder loss for molecule sequence generation. Proceedings of the 11th ACM International Conference on Bioinformatics, Computational Biology and Health Informatics. 2020; pp 1–7.

Lucas, J.; Tucker, G.; Grosse, R. B.; Norouzi, M. Understanding Posterior Collapse in Generative Latent Variable Models. *DGS@ICLR* 2019.

Maas, A. L.; Hannun, A.; Ng, A. Rectifier Nonlinearities Improve Neural Network Acoustic Models. Proceedings of the 30th International Conference on Machine Learning. 2013.

Kingma, D. P.; Ba, J. Adam: A Method for Stochastic Optimiza-
14

14

tion. arXiv preprint arXiv:1412.6980 2017.

66Bergstra, J.; Bengio, Y. Randomized Search for Hyper-Parameter
Optimization. J. Mach. Learn. Res. 2012, 13, 281–305.

67Kingma, D. P.; Welling, M. Auto-encoding variational bayes.
arXiv preprint arXiv:1312.6114 2013.

68Kullback, S.; Leibler, R. A. On information and sufficiency. Ann.
Math. Stat. 1951, 22, 79–86.

70Kusner, M. J.; Paige, B.; Hernández-Lobato, J. M. Grammar
variational autoencoder. International conference on machine
learning. 2017; pp 1945–1954.

73Ba, J. L.; Kiros, J. R.; Hinton, G. E. Layer Normalization. arXiv
preprint arXiv:1607.06450 2016.

76Gulrajani, I.; Ahmed, F.; Arjovsky, M.; Dumoulin, V.; Rud-
ford, A.; Chintala, S.; Courville, A. Improved Techniques for Training GANs.
arXiv preprint arXiv:1696.04678 2016.

79Mirza, M.; Osindero, S. Conditional Generative Adversarial Nets.
arXiv preprint arXiv:1411.1784 2014.

80Gauthier, J. Conditional generative adversarial nets for convolu-
tional face generation. Class project for Stanford CS231N: convo-
lutional neural networks for visual recognition, Winter semester
2014. 2014.

82Sutton, R.; Barto, A. Reinforcement Learning: An Introduction;
The MIT Press, 2014.

85Mnih, V.; Kavukcuoglu, K.; Silver, D.; Graves, A.; Antonoglou,
I.; Wierstra, D.; Riedmiller, M. Playing Atari with Deep Reinforcement Learning.
arXiv preprint arXiv:1312.5502 2013.

88Mousavi, S. S.; Schukat, M.; Howley, E. Deep Reinforcement
Learning: An Overview. Lect. Notes Networks Syst. 2018, 16, 426–440.

91Van Hasselt, H.; Guez, A.; Silver, D. Deep reinforcement learning
with double Q-Learning. 30th AAAI Conf. Artif. Intell. AAAI 2016 2016, 2094–2100.
Table of Contents Graphic