CHEMISTRY OF MATERIALS

pubs.acs.org/cm

Article

Assessing Deep Generative Models in Chemical Composition Space

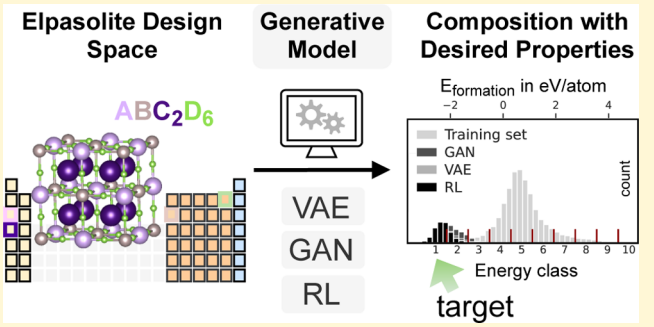
Hanna Türk,[§] Elisabetta Landini,[§] Christian Kunkel,[§] Johannes T. Margraf,[*] and Karsten Reuter

Cite This: *Chem. Mater.* 2022, 34, 9455−9467

Read Online

ACCESS | Metrics & More | Article Recommendations | Supporting Information

ABSTRACT: 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 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, variational autoencoders, and generative adversarial networks 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 are rigorously assessed. Additionally, a hyperparameter selection scheme for generative models in chemical composition space is developed.



1. INTRODUCTION

Generative machine learning (ML) models are increasingly used for the targeted generation of images,[1] video sequences,[2] text,[3] 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 the 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 for overcoming the limitations of the initially presented molecular library[10−14] by accessing molecules with a wide variety of chemical functionalities that are still consistent with the training examples.

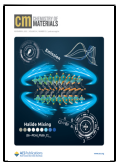
Beyond merely proposing realistic molecules, molecular design typically additionally demands the focused generation of candidates with desired properties. This can be achieved[15] by enhancing generative approaches via transfer learning,[16−20] semisupervised learning,[21,22] conditional generation,[21−26] reinforcement learning,[27−30] 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. This is because, on the one hand, a suitable (i.e., invertible) material representation that respects the symmetry and the periodicity of a three-dimensional crystal and is (ideally) independent of the number of atoms in the unit cell is not trivial to define.[34] On the other hand, a much larger variety of chemical compositions are possible for inorganic systems. The current paper will focus on the latter aspect.

In the prevailing data scarcity, inverse design of inorganic materials has 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]

Deep generative frameworks employing neural networks (NNs) have proven to be an invaluable tool in this context.[44] Notably, these comprise a large zoo of approaches, including variational autoencoders (VAEs),[34−36,39,42,45−47] generative

Received: June 22, 2022
Revised: October 5, 2022
Published: October 19, 2022



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 is whether the proposed candidates reliably display the desired properties.

Due to these questions, comparative studies which have been performed for organic molecule generation[11−14,52,53] and (less frequently) inorganic materials,[54] 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.,[55] which has previously served as a benchmark for regression models.[56−60] This set comprises the fully enumerated space of nearly 2 million systems, which can be derived by the main-group elemental exchange on the pristine quaternary Elpasolite mineral $AlNaK_2F_6$. Notably, this mineral is part of the quaternary double-perovskite prototype ABC_2D_6, 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 the 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.

2. METHODS

2.1. Dataset and Material Representation. The prototypical Elpasolite mineral ABC_2D_6 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, and D), leading to 1,974,024 possible structures. In ref 55, formation energies $E^{\mathrm{DFT}}_{\mathrm{formation}}$ computed at the PBE level are reported for a subset of 10,590 structures (assuming perfect cubic symmetry). Additionally, that work also provides accurate estimates of $E^{\mathrm{regression}}_{\mathrm{formation}}$ for the remaining structures, predicted by a kernel ridge regression (KRR) model with a mean absolute error 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[42] representation, closely following Faber et al.[55]

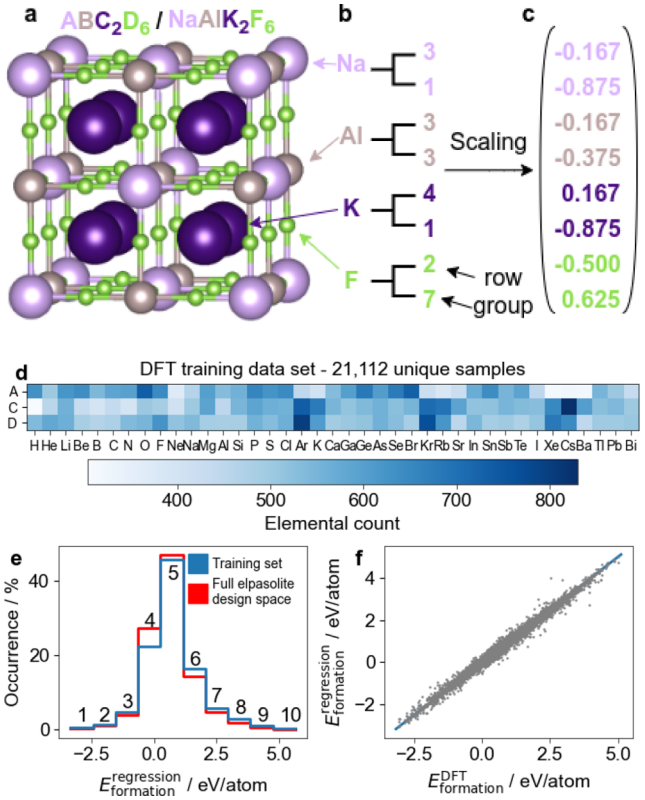


Figure 1. (a) Prototypical Elpasolite structure ABC_2D_6. (b) 8D representation of the Elpasolite elemental composition by row and main group number in the periodic table exemplified for $AlNaK_2F_6$. (c) Scaling of the 8D representation by eq 1. (d) Elemental distribution of the training dataset. (e) Formation energy distribution of DFT training and the full reference data set from ref 55, split into 10 equal-range energy classes, see text. (f) Formation energy of the DFT data vs the predicted one from the regression model.

In this way, each composition is represented by an eight-dimensional vector x_{8D}. As illustrated in Figure 1b, its entries correspond to the row number n_{row} (ranging from 1 to 6) and the main-group number n_{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

$$x_{8D,\mathrm{scaled}} = \frac{2x_{8D} - 1}{n_{\mathrm{row/group}}} - 1 \tag{1}$$

Note that this representation also encodes structures with nonexistent elements (e.g., first-row elements of group number 2−7) 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 ABC_2D_6 should have the same formation energy as BAC_2D_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 augmented the DFT training set by consistently adding all A/B permutations, obtaining a 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 to

https://doi.org/10.1021/acs.chemmater.2c01860
Chem. Mater. 2022, 34, 9455−9467

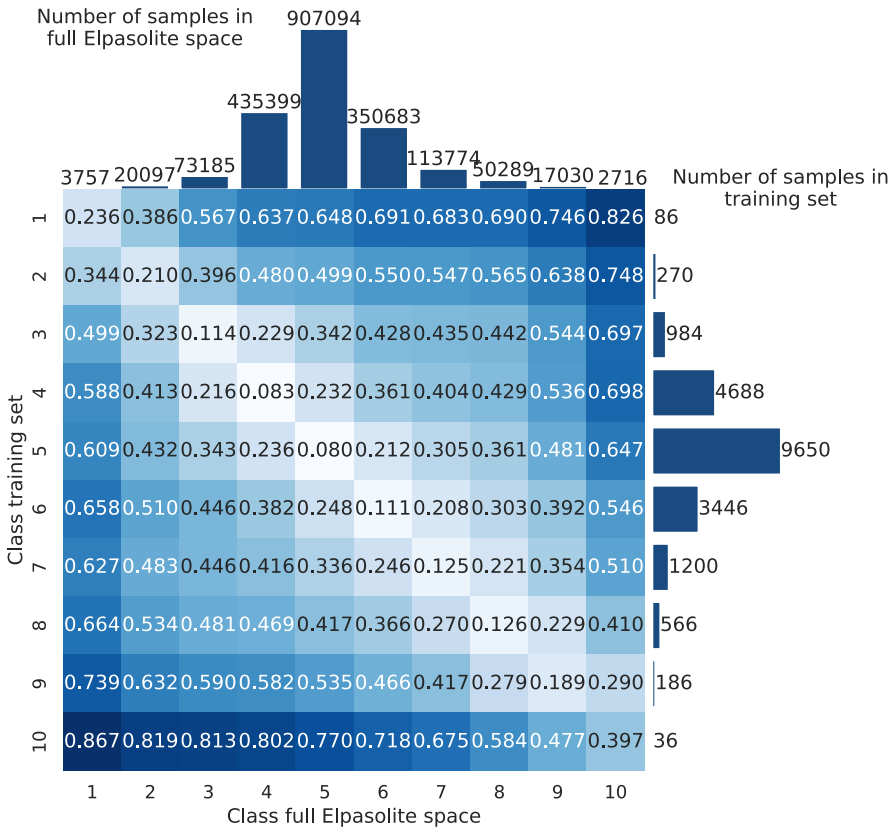


Figure 2. Reference JS distances between the training set and the full Elpasolite space.

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 $E^{\mathrm{DFT}}_{\mathrm{formation}}$ scale. Class 1 spans the lowest formation energies and thus the most stable Elpasolite compositions, while class 10 spans the highest formation energies and thus the 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. Note that this division into classes is mainly performed in order to simplify the subsequent analysis of the models. It would also be possible to directly condition the models on the formation energy.

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, that is, the fact that, for example, much fewer samples are in class 1 than in class 5 as apparent in 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 data loader. This code implements the described preprocessing 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.

2.2. 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 GAN[61] or VAE models,[62,63] 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.[55] We thus know the total number of N_{class} of compositions in each class (see Figure 1e) and the elemental distributions on the four sites (A, B, C, and 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

$$\text{precision (right class)} = \frac{N^{\mathrm{gen}}_{\mathrm{class}}}{N^{\mathrm{gen}}} \tag{2}$$

where N^{gen} is the total number of generated samples and $N^{\mathrm{gen}}_{\mathrm{class}}$ 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

$$\text{precision (neigh. class)} = \frac{N^{\mathrm{gen}}_{\mathrm{class}-1} + N^{\mathrm{gen}}_{\mathrm{class}+1}}{N^{\mathrm{gen}}} \tag{3}$$

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

https://doi.org/10.1021/acs.chemmater.2c01860
Chem. Mater. 2022, 34, 9455−9467

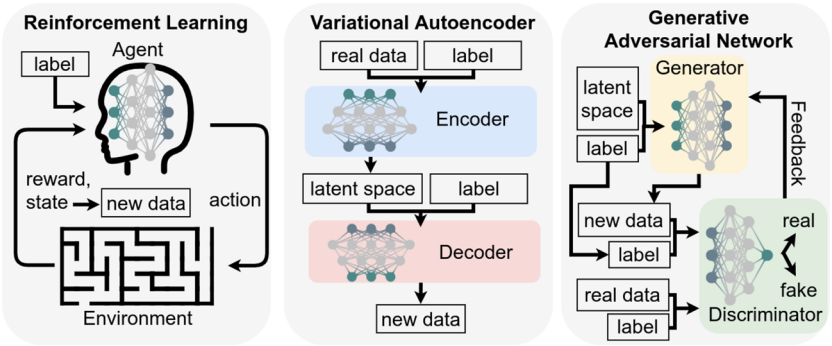


Figure 3. Schematic overview of the generative frameworks compared in this study.

$$\text{coverage (right class)} = \frac{N^{\text{gen}}_{\text{class,un.}}}{N_{\text{class}}} \tag{4}$$

where $N^{\text{gen}}_{\text{class,un.}}$ 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

$$\text{coverage (neigh. class)} = \frac{N^{\text{gen}}_{\text{class}-1,\text{un.}} + N^{\text{gen}}_{\text{class}+1,\text{un.}}}{N_{\text{class}-1,\text{un.}} + N_{\text{class}+1,\text{un.}}} \tag{5}$$

(5) The Jensen−Shannon (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 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

$$d_{\text{JS}}(p||q) = \frac{1}{4}\sum_S \sum_Z \left[\frac{1}{2}p(Z|S)\log_2\left(\frac{p(Z|S)}{m(Z|S)}\right) + \frac{1}{2}q(Z|S)\log_2\left(\frac{q(Z|S)}{m(Z|S)}\right)\right]^{1/2} \tag{6}$$

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

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 Figure 2. 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 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 a good agreement between the distributions.

2.3. Generative Frameworks. Our comparative study specifically covers the three deep generative frameworks shown in Figure 3. 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 data similar to the training examples when sampling from a low-dimensional latent space. Both models therefore effectively learn a (conditional) probability distribution of material compositions. However, the way this is achieved differs markedly. On the 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 the "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 multilayered 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 may display even better performance. All models are implemented using PyTorch (v1.9.0) and executed with Python (v3.8.8).

As a nonlinearity after every hidden layer, the commonly used LeakyReLU[64] 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[65] 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, which 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[12]) 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[66] was performed. See Table S4 for a listing of all searched parameters and their ranges. Figure S4 and Tables S5−S7 fully detail the procedure.

2.3.1. Variational Autoencoder. The generation of high-dimensional data with VAEs[67,68] proceeds as follows: a low-dimensional latent space vector $\mathbf{z}$ is drawn from a predefined probability distribution and mapped to a realistic data point $\tilde{\mathbf{x}}$ by a NN P_ϕ called the *decoder* (with trainable parameters ϕ).

$$\tilde{\mathbf{x}} = P_\phi(\mathbf{z}) \tag{7}$$

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

https://doi.org/10.1021/acs.chemmater.2c01860
Chem. Mater. 2022, 34, 9455−9467

$$q_\theta(\mathbf{z}|\mathbf{x}_i) = \mathcal{N}(\boldsymbol{\mu}(\mathbf{x}_i), \boldsymbol{\sigma}^2(\mathbf{x}_i)) \tag{8}$$

Here, the subscript θ indicates that this probability distribution depends on the trainable parameters of the encoder Q_θ.

Ideally, the decoder should be able to reconstruct $\mathbf{x}_i$ as accurately as possible, given a random sample $\mathbf{z}_\epsilon$ from the distribution $q_\theta(\mathbf{z}|\mathbf{x}_i)$. 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

$$J(\mathbf{x}_i|\phi, \theta) = \underbrace{(\mathbf{x}_i - P_\phi(\mathbf{z}_\epsilon))^2}_{\text{reconstruction}} + \underbrace{\lambda D_{\mathrm{KL}}(q_\theta(\mathbf{z}|\mathbf{x}_i)\|\mathcal{N}(0,1))}_{\text{regularization}} \tag{9}$$

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[69] D_{KL}, which quantifies the statistical distance between the learned distribution $q_\theta(\mathbf{z}|\mathbf{x}_i)$ 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 toward 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 $\mathbf{x}_i$ would be mapped to the same latent space distribution). This latter problem is known as posterior collapse.[63,70] To balance between these extremes, the regularization parameter λ 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.[71,72] 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_{18D} is encoded to an eight-dimensional latent space, and decoded into the scaled eight-dimensional representation of the composition. Both the encoder and decoder NNs have 2 hidden layers with 256 nodes. Layer normalization[73] 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 λ in the loss is fixed at 0.1.[62]

2.3.2. Generative Adversarial Network. Similar to the VAE, the GAN[74] framework uses a generator model G_θ (with trainable parameters θ), which is trained to generate realistic synthetic samples $\tilde{\mathbf{x}} = G_\theta(\mathbf{z})$ from a random latent space vector $\mathbf{z}_\epsilon$. Training of G_θ thereby follows an adversarial approach, which can be understood as a contest between G_θ and a discriminator model D_ϕ (with trainable parameters ϕ). The role of D_ϕ is to discriminate between real samples from the training set and the synthetic samples created by G_θ. The feedback from D_ϕ is in turn used to maximize the ability of G_θ to generate increasingly realistic samples, essentially trying to fool D_ϕ 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[61] is the so-called mode collapse—where G_θ resorts to fooling D_ϕ 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_ϕ 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_ϕ and G_θ 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

$$J_{\mathrm{D}}(\mathbf{x}_i|\phi, \theta) = \underbrace{-D_\phi(\mathbf{x}_i)}_{\text{real}} + \underbrace{D_\phi(G_\theta(\mathbf{z}_\epsilon))}_{\text{fake}} + \underbrace{\lambda\mathcal{P}(\mathbf{x}_i, G_\theta(\mathbf{z}_\epsilon))}_{\text{gradient penalty}} \tag{10}$$

Here, the first and second terms reward high scores for real and low scores for fake samples, respectively. Sparing the details,[78] 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

$$J_{\mathrm{G}}(\phi, \theta) = -D_\phi(G_\theta(\mathbf{z}_\epsilon)) \tag{11}$$

that is, it tries to maximize the score that D_ϕ 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_θ and D_ϕ 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_ϕ are updated five times as frequently as θ_{G}.[78] Based on the hyperparameter optimization, the final models use one hidden layer with 512 nodes each for G_θ and D_ϕ. An eight-dimensional latent space vector is passed to G_θ and decoded to the structural representation $x_{8\mathrm{D,\ scaled}}$, 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.

2.3.3. 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 to 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 the 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 the 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 $KMgXe_2O_6$ (where the scaling of eq 1 is omitted for clarity)

https://doi.org/10.1021/acs.chemmater.2c01860
Chem. Mater. 2022, 34, 9455−9467

$$\underbrace{(\_\ \_\ \_\ \_\ \_\ \_\ \_\ \_)}_{\text{state 0}} \xrightarrow{\text{action 0}} \underbrace{(4\ \_\ \_\ \_\ \_\ \_\ \_\ \_)}_{\text{state 1}}$$

$$\underbrace{(4\ \_\ \_\ \_\ \_\ \_\ \_\ \_)}_{\text{state 1}} \xrightarrow{\text{action 1}} \underbrace{(4\ 1\ \_\ \_\ \_\ \_\ \_\ \_)}_{\text{state 2}}$$

$$\vdots$$

$$\underbrace{(4\ 1\ 3\ 2\ 5\ 8\ 2\ \_)}_{\text{state 7}} \xrightarrow{\text{action 7}} \underbrace{(4\ 1\ 3\ 2\ 5\ 8\ 2\ 6)}_{\text{state 8}}$$

The agent thus starts from an empty composition vector and chooses the row of the element at site A of the Elpasolite structure (see Figure 1). 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 s'. The generation process can thus be described as a sequence of state-action pairs $(s_0, a_0, s_1, a_1, ..., s_7, a_7, s_8)$, with a final state s_8. 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

$$r(s, a) = \begin{cases} 0 & \text{if } s' \text{ non-final} \\ 1 & \text{if } s' \text{final and in desired class} \\ -1 & \text{if } s' \text{final and not in desired class} \end{cases} \quad (12)$$

The key component of any RL model is the policy π, 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 optimal long-term reward Q_* for a given state-action pair (s, a) is formalized by the Bellman equation

$$Q_*(s, a) = r(s, a) + \gamma \max_{a'} Q_*(s', a') \quad (13)$$

The optimal long-term reward $Q_*(s, a)$ (referred to as the *state-action value*) is thus recursively defined as a sum of the immediate reward $r(s, a)$ and the state-action value $Q_*(s', a')$, where a' is the action that maximizes $Q_*(s', a')$. The discount factor γ takes the potentially diminishing relevance of recursively included future actions into account. Specifically, by choosing $\gamma = 1$, all future rewards are weighted equally, whereas $\gamma \leq 1$ leads to a reduced impact of rewards that are acquired much after the action a.

Clearly, the state-action value Q_* in principle provides a sound basis for guiding the actions of an agent. However, it is generally only computable by exhaustively exploring all possible actions into the future, which would defeat the purpose of RL. The Q-learning approach[85] provides a way out of this problem as it allows the iterative approximation of Q_*. In the original Q-learning approach, this is achieved via a table of Q-values for all possible state-action combinations. This Q-table is iteratively updated as the agent explores the state space, according to

$$Q(s, a) = (1 - \alpha)Q(s, a) + \alpha(r(s, a) + \gamma \max_{a'} Q(s', a')) \quad (14)$$

where α is a learning rate. Although the Q-table is initialized with arbitrary values, this algorithm eventually converges to the true state-action values.

Relying on a Q-table is impractical for high-dimensional settings (such as materials design), however, because enumerating all possible states is in general not possible in this case. To overcome this limitation, the deep-Q network (DQN) approach was developed.[86] Here, the discrete Q-table is replaced by a NN Q_θ (with trainable parameters θ). This function takes any state s as an input and predicts the corresponding Q-values for all possible actions a.

To train this network, the cost function $J(\theta)$ defines a least-squares regression with eq 13 as the target value[85]

$$J(e_i, \theta) = \sum_{s,a \in e_i} [\underbrace{Q_\theta(s, a)}_{\text{prediction}} - \underbrace{r(s, a) + \gamma \max_{a'} Q_\theta(s', a')}_{\text{target}}]^2 \quad (15)$$

This function sums over the state-action pairs (s, a) of a given training episode e_i. 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_θ. 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.

It should be noted that the above description refers to the "batch" or "offline" RL setting,[87] 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.[88] To mitigate this, we use the double deep Q-learning[89] approach, in which two different versions of Q_θ 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_θ 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 π, that is, 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

$$p(s, a) = \frac{\exp(\beta Q_\theta(s, a))}{\sum_{a'} \exp(\beta Q_\theta(s, a'))} \quad (16)$$

Here, the sum in the denominator is over all possible actions (including a) and the hyperparameter β 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[73]). 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 10 steps.

3. RESULTS AND DISCUSSION

3.1. 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 composi-

https://doi.org/10.1021/acs.chemmater.2c01860
Chem. Mater. 2022, 34, 9455−9467

tions in the lowest energy class 1. As shown in Figure 1e, this class is one of the least populated and contains only 86 samples, corresponding to 0.4% of the total training set. This imbalance is equally present in the full Elpasolite space, with only 3757 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 4 illustrates the performance of a representative GAN model conditioned on the minority class 1 for the first 250,000

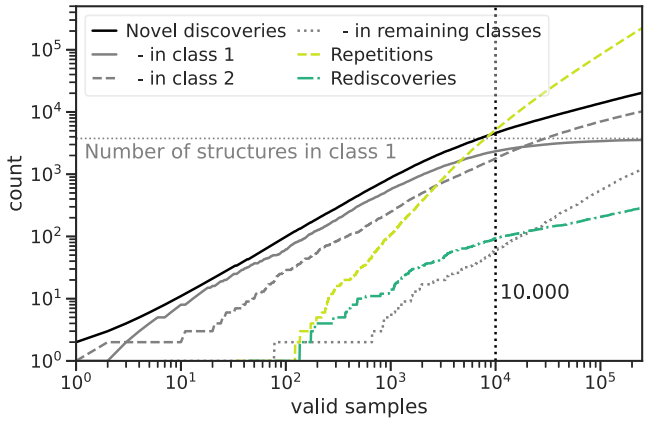


Figure 4. Performance of a representative GAN model for the first 250,000 generated valid samples when conditioned on the minority class 1. Repetitions are previously generated samples. Rediscoveries correspond to known training compositions. Novel discoveries are further discerned by the classes they belong to (in gray).

generated samples. Here, we distinguish between "novel discoveries" (unknown compositions proposed for the first time), "repetitions" (compositions which have previously been generated by this model), and "rediscoveries" (compositions which are part of the training set). The novel discoveries are further discerned according to their class.

This reveals that the model initially displays a high degree of novel discoveries in the target class, which saturates after ca. 10,000 generated samples. At this point, the number of repetitions begins to dominate the generation process, although some novel materials from class 2 are still discovered. Meanwhile, the numbers of rediscoveries and novel discoveries from other classes remain very low. Similar plots can be obtained for the VAE and RL models; see the Supporting Information.

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 generated or that high-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. Importantly, very few compositions from class 3 and beyond are generated, however, even though these make up the bulk of the Elpasolite composition space. The model thus correctly assigns very low probabilities to these compositions.

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 the model clearly does more than mere training-set memorization. Since repetitions begin to dominate the generation procedure after 10,000 samples, we terminate the generation procedure at this point for the following analyses of minority class generation.

Figure 5 shows the formation energy distributions of the unique compositions produced by representative VAE, GAN,

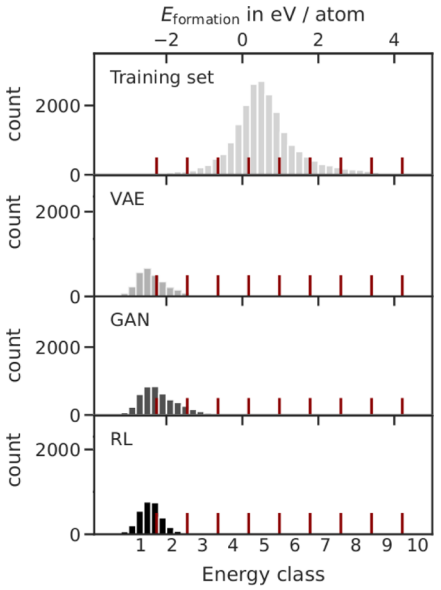


Figure 5. Formation energy distribution (in 0.2 eV/atom bins) of the unique class 1 conditioned compositions proposed by the three generative models (VAE, GAN, and RL) over 10,000 valid samples. Additionally shown is the corresponding distribution over the entire training set of 21,112 compositions.

and RL models. All three models obviously achieve conditional generation in the minority class 1 with great success. Specifically, they produce sample distributions with low formation energies, completely unlike the formation energy distribution of the original training set.

This plot also illustrates from how little class 1 data the models are able to learn. On this scale, the 87 class 1 samples in the training set are not even visible. Nonetheless, all three models generate over 1000 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.

For a more quantitative comparison of the generative frameworks considered herein, average performance metrics obtained from 50 model initializations are summarized in Table 1. 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

https://doi.org/10.1021/acs.chemmater.2c01860
Chem. Mater. 2022, 34, 9455−9467

Table 1. Performance Metrics for the Three Generative Models Conditioned to Generate Minority Class 1 Compositions after 10,000 Valid Samples[a]

	RL	VAE	GAN
precision (right class)	69 ± 8%	83 ± 1%	66 ± 2%
Precision (neigh. class)	30 ± 7%	16 ± 1%	28 ± 1%
coverage (right class)	53 ± 3%	54 ± 2%	62 ± 1%
coverage (neigh. class)	4 ± 1%	5 ± 0%	10 ± 1%
JS distance	0.20 ± 0.03	0.16 ± 0.01	0.16 ± 0.01

[a]Shown are averages over 50 model fits together with the standard deviation.

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 Figure 2). A more detailed illustration of the elemental distributions is given in Figure 6. 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, for example, alkali metals and alkaline earth on site C, although this tendency is less pronounced. The generative models thus develop a useful chemical intuition for how to construct stable Elpasolites.

To put these metrics into perspective, simple baseline models based on the prevalence of element combinations in class 1 were also developed (see the Supporting Information). The deep generative models discussed herein display higher coverage and precision by at least a factor of 2, as well as significantly lower JS distances. Since the choice of terminating the generation after 10,000 samples is rather arbitrary, the same analysis was also performed for 3500 samples (see Table S2). This reveals that precision and JS distance are unaffected by the number of samples, whereas (unsurprisingly) the coverage is somewhat lower when fewer samples are generated.

3.2. 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

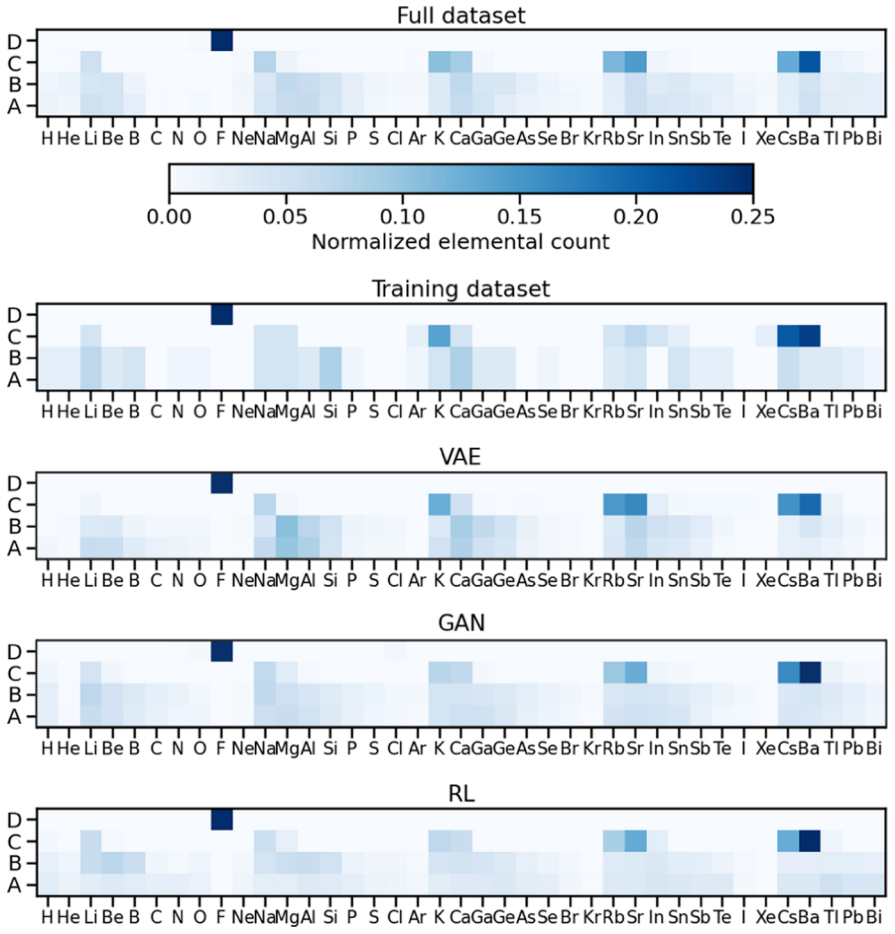


Figure 6. Comparison of the normalized elemental distributions $p(Z|S)$ in class 1 over the four sites (A, B, C, and D) for the full Elpasolite space, the training set, and datasets conditionally generated by the three models (VAE, GAN, and RL). The color scale is chosen to depict the details of the overall distribution. The value of fluorine on lattice site D surpasses its limits, exceeding 0.997 in all cases.

https://doi.org/10.1021/acs.chemmater.2c01860
Chem. Mater. 2022, 34, 9455−9467

46% of the entire Elpasolite composition space and is thus by 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 9650 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).

Analyzing the generation run of a representative GAN model for the majority class (Figure 7) reveals largely analogous

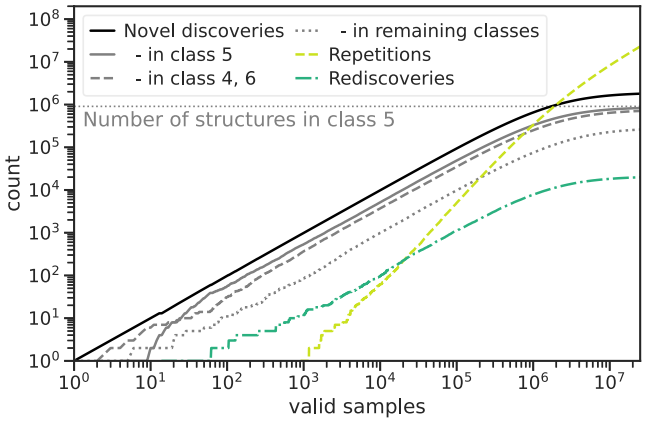


Figure 7. Same as Figure 4, but for a GAN model conditioned on the majority class 5. Note the much larger number of samples required to saturate the number of unique novel discoveries generated, reflecting the much larger composition space covered by class 5.

behavior to the minority class case shown in Figure 4. 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 the Supporting Information) so that we evaluate the performance metrics for the majority class after 25 million valid samples in the following.

Table 2 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

Table 2. Performance Metrics for the Three Generative Models Conditioned to Generate Majority Class 5 Compositions after 25 Million Valid Samples[a]

	RL	VAE	GAN
precision (right class)	87 ± 7%	58 ± 1%	54 ± 1%
Precision (neigh. classes)	13 ± 7%	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

[a]Shown are averages over 50 model fits together with the standard deviation.

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 β 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). As above, a similar analysis was performed for a smaller number of generated samples (2.5 million, see Table S3). Again, the precision and JS distance are unaffected by this choice, whereas the coverage decreases.

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 Figure S3), we find that VAE and GAN sample elements are 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 its own characteristic elemental fingerprints. This is also reflected by the fact that RL displays a much higher number of repetitions in the generation process. Overall, 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.

3.3. 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 ∼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 require 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 $\frac{N_{\text{train}}}{2}$. These results are summarized in Figure 8. For comparison, we also include the results for

https://doi.org/10.1021/acs.chemmater.2c01860
Chem. Mater. 2022, 34, 9455−9467

models trained with the original training set reported by Faber et al. (*without* permutational symmetry).

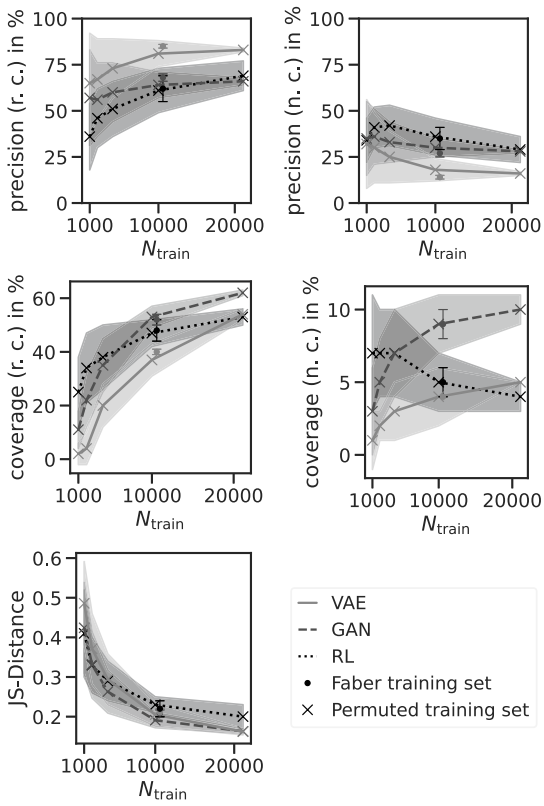


Figure 8. Influence of training set size on the performance metrics of VAE, GAN, and RL models for minority class generation after 10,000 generated samples. Shown are averages over 50 model fits employing 50 different random training subsets. Shaded areas represent the corresponding standard deviations.

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 of 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.

4. CONCLUSIONS

We herein assessed the performance of three deep generative ML 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_2D_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 NN architectures, all studied models are capable of reliably generating candidates within the desired formation energy classes. This shows that reasonable model hyperparameters could be determined by an automated procedure while accommodating a reasonable trade-off between coverage and precision in a conditional generation.

Nonetheless, there are a number of notable differences between the approaches. The RL models showed greater robustness toward small datasets but also greater variability 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 the majority class generation. 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, although 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.

It should be noted that ML-based regression models remain a powerful alternative to generative models for materials discovery, particularly in relatively small design spaces such as the one considered herein. The question of whether a generative or regression model is more appropriate for a given task thus remains interesting. In previous work on organic semiconductors, we found that regression models are not well suited for discovering exceptional materials since these are by definition strongly underrepresented in the training set.[90] The excellent performance of the generative models for minority class generation is highly promising in this context.

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.

https://doi.org/10.1021/acs.chemmater.2c01860
Chem. Mater. 2022, 34, 9455−9467

■ ASSOCIATED CONTENT

Supporting Information

The Supporting Information is available free of charge at https://pubs.acs.org/doi/10.1021/acs.chemmater.2c01860.

Additional generation plots for VAE and RL, elemental distributions for majority class generation, additional details for the hyperparameter search procedure, and GAN generation plots with smaller training sets (PDF)

■ AUTHOR INFORMATION

Corresponding Author

Johannes T. Margraf − *Fritz-Haber-Institut der Max-Planck-Gesellschaft, D-14195 Berlin, Germany;* orcid.org/0000-0002-0862-5289; Email: margraf@fhi.mpg.de

Authors

Hanna Türk − *Fritz-Haber-Institut der Max-Planck-Gesellschaft, D-14195 Berlin, Germany; Chair for Theoretical Chemistry and Catalysis Research Center, Department of Chemistry, Technische Universität München, 85748 Garching, Germany;* orcid.org/0000-0002-9858-7019

Elisabetta Landini − *Fritz-Haber-Institut der Max-Planck-Gesellschaft, D-14195 Berlin, Germany; Chair for Theoretical Chemistry and Catalysis Research Center, Department of Chemistry, Technische Universität München, 85748 Garching, Germany*

Christian Kunkel − *Fritz-Haber-Institut der Max-Planck-Gesellschaft, D-14195 Berlin, Germany*

Karsten Reuter − *Fritz-Haber-Institut der Max-Planck-Gesellschaft, D-14195 Berlin, Germany*

Complete contact information is available at: https://pubs.acs.org/10.1021/acs.chemmater.2c01860

Author Contributions

§H.T., E.L., and C.K. contributed equally.

Funding

Open access funded by Max Planck Society.

Notes

The authors declare no competing financial interest.

■ ACKNOWLEDGMENTS

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

■ REFERENCES

(1) Brock, A.; Donahue, J.; Simonyan, K.Large Scale GAN Training for High Fidelity Natural Image Synthesis. **2019**. arXiv preprint arXiv:1809.11096

(2) Clark, A.; Donahue, J.; Simonyan, K.Adversarial Video Generation on Complex Datasets. **2019**. arXiv preprint arXiv:1907.06571

(3) Nie, W.; Narodytska, N.; Patel, A.RelGAN: Relational Generative Adversarial Networks for Text Generation. *International Conference on Learning Representations*, 2019.

(4) Dong, 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 Annual AAAI Conference on Artificial Intelligence*, 2018.

(5) Merk, D.; Friedrich, L.; Grisoni, F.; Schneider, G. De Novo Design of Bioactive Small Molecules by Artificial Intelligence. *Mol. Inform.* **2018**, *37*, 1700153.

(6) Zhavoronkov, A.; et al. Deep learning e rapid identification of potent DDR1 kinase inhibitors. *Nat. Biotechnol.* **2019**, *37*, 1038−1040.

(7) Sanchez-Lengeling, B.; Aspuru-Guzik, A. Inverse molecular design using machine learning: Generative models for matter engineering. *Science* **2018**, *361*, 360−365.

(8) Elton, 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.

(9) Merz, K. M.; De Fabritiis, G.; Wei, G.-W. Generative Models for Molecular Design. *J. Chem. Inf. Model.* **2020**, *60*, 5635−5636.

(10) Arús-Pous, J.; Blaschke, T.; Ulander, S.; Reymond, J.-L.; Chen, H.; Engkvist, O. Exploring the GDB-13 chemical space using deep generative models. *J. Cheminf.* **2019**, *11*, 20.

(11) Brown, N.; Fiscato, M.; Segler, M. H.; Vaucher, A. C. GuacaMol: Benchmarking Models for de Novo Molecular Design. *J. Chem. Inf. Model.* **2019**, *59*, 1096−1108.

(12) Polykovskiy, D.; et al. Molecular Sets (MOSES): A Benchmarking Platform for Molecular Generation Models. *Front. Pharmacol.* **2020**, *11*, 565644.

(13) Zhang, 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.

(14) Flam-Shepherd, D.; Zhu, K.; Aspuru-Guzik, A.Keeping it Simple: Language Models can learn Complex Molecular Distributions. **2021**. arXiv preprint arXiv:2112.03041

(15) Sousa, T.; Correia, J.; Pereira, V.; Rocha, M. Generative Deep Learning for Targeted Compound Design. *J. Chem. Inf. Model.* **2021**, *61*, 5343−5361.

(16) Gupta, A.; Müller, 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.

(17) Segler, M. H. S.; Kogej, T.; Tyrchan, C.; Waller, M. P. Generating Focused Molecule Libraries for Drug Discovery with Recurrent Neural Networks. *ACS Cent. Sci.* **2018**, *4*, 120−131.

(18) Amabilino, 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*, 5699−5713.

(19) Cai, 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.

(20) Moret, M.; Friedrich, L.; Grisoni, F.; Merk, D.; Schneider, G. Generative molecular design in low data regimes. *Nat. Mach. Intell.* **2020**, *2*, 171−180.

(21) Kang, S.; Cho, K. Conditional Molecular Design with Deep Generative Models. *J. Chem. Inf. Model.* **2019**, *59*, 43−52.

(22) Lim, J.; Hwang, S.-Y.; Moon, S.; Kim, S.; Kim, W. Y. Scaffold-based molecular design with a graph generative model. *Chem. Sci.* **2020**, *11*, 1153−1164.

(23) Li, Y.; Zhang, L.; Liu, Z. Multi-objective de novo drug design with conditional graph generative model. *J. Cheminf.* **2018**, *10*, 33.

(24) Kotsias, P.-C.; Arús-Pous, J.; Chen, H.; Engkvist, O.; Tyrchan, C.; Bjerrum, E. J. Direct steering of de novo molecular generation with descriptor conditional recurrent neural networks. *Nat. Mach. Intell.* **2020**, *2*, 254−265.

(25) 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.

(26) Lim, J.; Ryu, S.; Kim, J. W.; Kim, W. Y. Molecular generative model based on conditional variational autoencoder for de novo molecular design. *J. Cheminf.* **2018**, *10*, 31.

(27) 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. **2017**. arXiv preprint arXiv:1705.10843

(28) Popova, M.; Isayev, O.; Tropsha, A. Deep reinforcement learning for de novo drug design. *Sci. Adv.* **2018**, *4*, No. eaap7885.
(29) Olivecrona, M.; Blaschke, T.; Engkvist, O.; Chen, H. Molecular de-novo design through deep reinforcement learning. *J. Cheminf.* **2017**, *9*, 48.
(30) 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.
(31) 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.
(32) Jin, W.; Barzilay, R.; Jaakkola, T.Junction Tree Variational Autoencoder for Molecular Graph Generation. **2019**. arXiv preprint arXiv:1802.04364
(33) Griffiths, R.-R.; Hernández-Lobato, J. M. Constrained Bayesian optimization for automatic chemical design using variational autoencoders. *Chem. Sci.* **2020**, *11*, 577−586.
(34) 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.
(35) 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.
(36) Lee, I.-H.; Chang, K. J. Crystal structure prediction in a continuous representative space. *Comput. Mater. Sci.* **2021**, *194*, 110436.
(37) Long, T.; Fortunato, N. M.; Opahle, I.; Zhang, Y.; Samathrakis, I.; Shen, C.; Gutfleisch, O.; Zhang, H. Constrained crystals deep convolutional generative adversarial network for the inverse design of crystal structures. *npj Comput. Mater.* **2021**, *7*, 66.
(38) 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.
(39) 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.
(40) Dong, Y.; Li, D.; Zhang, C.; Wu, C.; Wang, H.; Xin, M.; Cheng, J.; Lin, J. Inverse design of two-dimensional graphene/h-BN hybrids by a regressional and conditional GAN. *Carbon* **2020**, *169*, 9−16.
(41) 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.
(42) Pathak, Y.; Juneja, K. S.; Varma, G.; Ehara, M.; Priyakumar, U. D. Deep learning enabled inorganic material generator. *Phys. Chem. Chem. Phys.* **2020**, *22*, 26935−26943.
(43) 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.
(44) 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.
(45) Hoffmann, J.; Maestrati, L.; Sawada, Y.; Tang, J.; Sellier, J. M.; Bengio, Y.Data-driven approach to encoding and decoding 3-d crystal structures. **2019**. arXiv preprint arXiv:1909.00949
(46) Ren, Z.; et al. An invertible crystallographic representation for general inverse design of inorganic crystals with targeted properties. *Matter* **2022**, *5*, 314−335.
(47) 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.
(48) Nouira, A.; Sokolovska, N.; Crivello, J.-C.CrystalGAN: Learning to Discover Crystallographic Structures with Generative Adversarial Networks. **2019**. arXiv preprint arXiv:1810.11203
(49) Debnath, A.; Krajewski, 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. **2021**. arXiv preprint arXiv:2108.12019
(50) Sajedian, I.; Badloe, T.; Rho, J. Optimisation of colour generation from dielectric nanostructures using reinforcement learning. *Opt. Express* **2019**, *27*, 5874.
(51) 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.
(52) Renz, P.; Van Rompaey, D.; Wegner, J. K.; Hochreiter, S.; Klambauer, G. On failure modes in molecule generation and optimization. *Drug Discovery Today: Technol.* **2019**, *32-33*, 55−63.
(53) 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.
(54) Sawada, Y.; Morikawa, K.; Fujii, M.Study of deep generative models for inorganic chemical compositions. **2019**. arXiv preprint arXiv:1910.11499
(55) 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.
(56) 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.
(57) Tshitoyan, V.; Dagdelen, J.; Weston, L.; Dunn, A.; Rong, Z.; Kononova, O.; Persson, K. A.; Ceder, G.; Jain, A. Unsupervised word embeddings capture latent knowledge from materials science literature. *Nature* **2019**, *571*, 95−98.
(58) Huo, H.; Rupp, M.Unified Representation of Molecules and Crystals for Machine Learning. **2017**. arXiv preprint arXiv:1704.06439
(59) 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*, 29661−29668.
(60) 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.
(61) Metz, L.; Poole, B.; Pfau, D.; Sohl-Dickstein, J.Unrolled Generative Adversarial Networks. **2017**. arXiv preprint arXiv:1611.02163
(62) 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.
(63) Lucas, J.; Tucker, G.; Grosse, R. B.; Norouzi, M.*Understanding Posterior Collapse in Generative Latent Variable Models*; DGS@ICLR, 2019.
(64) Maas, A. L.; Hannun, A.; Ng, A.Rectifier Nonlinearities Improve Neural Network Acoustic Models. *Proceedings of the 30th International Conference on Machine Learning*, 2013.
(65) Kingma, D. P.; Ba, J.Adam: A Method for Stochastic Optimization. **2017**. arXiv preprint arXiv:1412.6980
(66) Bergstra, J.; Bengio, Y. Random Search for Hyper-Parameter Optimization. *J. Mach. Learn. Res.* **2012**, *13*, 281−305.
(67) Kingma, D. P.; Welling, M.Auto-encoding variational bayes. **2013**. arXiv preprint arXiv:1312.6114
(68) Rezende, D. J.; Mohamed, S.; Wierstra, D.Stochastic back-propagation and approximate inference in deep generative models. *Proceedings of the 31st International Conference on Machine Learning*, 2014; pp 1278−1286.
(69) Kullback, S.; Leibler, R. A. On information and sufficiency. *Ann. Math. Stat.* **1951**, *22*, 79−86.
(70) Kusner, M. J.; Paige, B.; Hernández-Lobato, J. M.*Grammar Variational Autoencoder. International Conference on Machine Learning*, 2017; pp 1945−1954.

(71) Sohn, K.; Lee, H.; Yan, X.Learning Structured Output Representation using Deep Conditional Generative Models. *Advances in Neural Information Processing Systems*, 2015.
(72) Karanasou, P.; Karlapati, S.; Moinet, A.; Joly, A.; Abbas, A.; Slangen, S.; Trueba, J. L.; Drugman, T.A learned conditional prior for the VAE acoustic space of a TTS system. **2021**. arXiv preprint arXiv:2106.10229
(73) Ba, J. L.; Kiros, J. R.; Hinton, G. E.Layer Normalization. **2016**. arXiv preprint arXiv:1607.06450
(74) Goodfellow, I. J.; Pouget-Abadie, J.; Mirza, M.; Xu, B.; Warde-Farley, D.; Ozair, S.; Courville, A.; Bengio, Y.Generative Adversarial Networks. **2014**. arXiv preprint arXiv:1406.2661
(75) Salimans, T.; Goodfellow, I.; Zaremba, W.; Cheung, V.; Radford, A.; Chen, X.Improved Techniques for Training GANs. **2016**. arXiv preprint arXiv:1606.03498
(76) Gui, J.; Sun, Z.; Wen, Y.; Tao, D.; Ye, J.A Review on Generative Adversarial Networks: Algorithms, Theory, and Applications. **2020**. arXiv preprint arXiv:2001.06937
(77) Arjovsky, M.; Chintala, S.; Bottou, L.; Wasserstein GAN. **2017**. arXiv preprint arXiv:1701.07875
(78) Gulrajani, I.; Ahmed, F.; Arjovsky, M.; Dumoulin, V.; Courville, A.Improved Training of Wasserstein GANs. **2017**. arXiv preprint arXiv:1704.00028
(79) Mirza, M.; Osindero, S.Conditional Generative Adversarial Nets. **2014**. arXiv preprint arXiv:1411.1784
(80) Gauthier, J.Conditional generative adversarial nets for convolutional face generation. *Class Project for Stanford CS231N: Convolutional Neural Networks for Visual Recognition, Winter semester*2014; Vol. *2014*, p 2.
(81) Simm, G. N.; Pinsler, R.; Hernández-Lobato, J. M.Reinforcement learning for molecular design guided by quantum mechanics. *37th International Conference on Machine Learning. ICML 2020*, 2020; PartF168147-12, pp 8906−8916.
(82) Blaschke, T.; Engkvist, O.; Bajorath, J.; Chen, H. Memory-assisted reinforcement learning for diverse molecular de novo design. *J. Cheminf.* **2020**, *12*, 68.
(83) Meldgaard, S. A.; Köhler, J.; Mortensen, H. L.; Christiansen, M.-P. V.; Noé, F.; Hammer, B.Generating stable molecules using imitation and reinforcement learning. **2021**. arXiv preprint arXiv:2107.05007
(84) Thiede, L. A.; Krenn, M.Curiosity in exploring chemical space : Intrinsic rewards for deep molecular reinforcement learning. **2020**. arXiv preprint arXiv:2012.11293
(85) Sutton, R.; Barto, A.*Reinforcement Learning: An Introduction*; The MIT Press, 2014.
(86) Mnih, V.; Kavukcuoglu, K.; Silver, D.; Graves, A.; Antonoglou, I.; Wierstra, D.; Riedmiller, M.Playing Atari with Deep Reinforcement Learning. **2013**. arXiv preprint arXiv:1312.5602
(87) Mousavi, S. S.; Schukat, M.; Howley, E. Deep Reinforcement Learning: An Overview. *Lect. Notes Networks Syst.* **2018**, *16*, 426−440.
(88) Hessel, M.; Modayil, J.; Van Hasselt, H.; Schaul, T.; Ostrovski, G.; Dabney, W.; Horgan, D.; Piot, B.; Azar, M.; Silver, D.Rainbow: Combining improvements in deep reinforcement learning. *32nd AAAI Conference on Artificial Intelligence AAAI 2018*, 2018; pp 3215−3222.
(89) Van Hasselt, H.; Guez, A.; Silver, D.Deep reinforcement learning with double Q-Learning. *30th AAAI Conference on Artificial Intelligence AAAI 2016*, 2016; pp 2094−2100.
(90) Chen, K.; Kunkel, C.; Reuter, K.; Margraf, J. T. Reorganization energies of flexible organic molecules as a challenging target for machine learning enhanced virtual screening. *Digi. Discovery* **2022**, *1*, 147−157.

https://doi.org/10.1021/acs.chemmater.2c01860
Chem. Mater. 2022, 34, 9455−9467