

Generative Deep Learning for Advanced Battery Materials

Deepalaxmi Rajagopal,* Adrian Cierpka, Britta Nestler, and Arnd Koeppel

The development of battery materials presents a complex multi-scale challenge, where optimizing the properties of battery systems across various length scales is essential for achieving targeted performance, enhanced safety, lower costs, and resource availability. Traditional methods for solving this complex, multiscale problem rely on time-intensive trial-and-error approaches, which hinder progress. However, integrating advanced machine learning (ML) frameworks significantly changes the landscape of battery materials research by enabling faster discovery, predictive modeling, and optimization of

material properties. Among the ML frameworks, generative deep learning (DL) models stand out, as they capture the statistics of real-world scenarios by learning an underlying condensed representation of a higher-dimensional input space to generate information-rich outputs. By merging computational techniques with experimental research, generative DL provides a significant paradigm shift in analyzing battery materials. This review aims to provide valuable insights into generative models, highlighting their potential to accelerate the characterization, screening, and design of battery materials.

1. Introduction

Batteries are composed of a complex material system that includes two electrodes containing redox pairs, which are separated by an electronically insulating medium known as the electrolyte. The materials used in battery components must be carefully selected to meet requirements for energy density, safety, cost, and sustainability. Current battery research focuses on developing new battery chemistries to meet target functionalities that surpass those of traditional lithium-ion batteries.^[1]

Various factors need to be considered to understand the electrochemical functionality of new battery chemistries, including material synthesizability, manufacturing, cell assembly, and performance analysis. Even at the material level, the possibility of finding material compositions from naturally occurring elements in the periodic table that meet the given target performance and cell requirements is beyond human capability. Moreover, relying solely on conventional trial-and-error experimental approaches is insufficient for exhaustively exploring the vast compositional space to discover optimal material stoichiometries.^[2,3]


With the increase in computational power, multiscale simulations are used in the development of new materials.

By maintaining specific material parameters as constants, researchers can more efficiently and cost-effectively investigate the structural, chemical, and functional characteristics of innovative battery materials, thereby advancing the development of functional materials. Simulation tools that incorporate the principles of physics, chemistry, and mathematics are vital for a comprehensive understanding of material behavior on various scales.^[4] Quantum mechanics^[5] is critical in elucidating the electronic structure of materials, whereas molecular dynamics^[6,7] effectively models the interactions and movements of atoms and molecules. Continuum modeling^[8] focuses on predicting properties and behaviors ranging from microscopic to macroscopic scales. Together, these theoretical frameworks provide a thorough understanding of material dynamics, greatly enhancing the ability to analyze and predict their behavior. Using computational tools to simulate the behavior of new materials on different scales results in a large amount of information-rich data, which in turn fuels the development of material databases such as the Materials Project (MP) database,^[9] AFLOW,^[10] Cambridge Structural Database,^[11] and Inorganic Crystal Structure Database.^[12] The different material chemistries can be studied using machine learning (ML) methods to determine the influence of material structure on properties that describe the necessary material functionality and to explain how different material properties can be used to define the material performance in the functional space.

ML is a subset of the broader field of artificial intelligence that utilizes a data-driven modeling approach to make predictions about the response variable based on observations of the system being studied.^[13,14] In the context of ML, data-driven models are categorized into discriminative and generative models based on how the model learns to sample the space of presented data. Discriminative models focus on directly mapping the input to the output and understanding the decision boundary between classes. In contrast, generative models learn the underlying probability distribution of the data itself, which allows them to generate new, realistic samples and produce information-rich

D. Rajagopal, A. Cierpka, B. Nestler, A. Koeppel
Institute of Applied Materials (IAM-MMS)
Karlsruhe Institute of Technology (KIT)
Straße am Forum 7, 76131 Karlsruhe, Germany
E-mail: deepalaxmi.rajagopal@kit.edu

D. Rajagopal, A. Cierpka, B. Nestler, A. Koeppel
Institute of Nanotechnology (INT)
Karlsruhe Institute of Technology (KIT)
Hermann-von-Helmholtz-Platz 1, 76344 Eggenstein-Leopoldshafen,
Germany

 © 2025 The Author(s). Batteries & Supercaps published by Wiley-VCH GmbH. This is an open access article under the terms of the Creative Commons Attribution License, which permits use, distribution and reproduction in any medium, provided the original work is properly cited.

outputs.^[15,16] The application modes of generative learning vary depending on how it models the probability distribution of the data, the type of model architecture used, and the training objectives.^[17–20] In contrast to previous complementary studies^[21–23] that primarily focused on the application of generative models in battery research and development, this article aims to provide a deeper understanding of the underlying theory of general ML and generative learning, along with updated application examples in a rapidly developing field. The review investigates how different training objectives can be employed to model generative learning for battery material analysis. Furthermore, the review aims to assist battery researchers outside of the ML domain in utilizing these data-driven tools to expedite their research timelines, particularly in terms of characterization, screening, and material design.

2. Foundation of Generative Learning

2.1. Generative Modeling

The capacity to imagine and create ideas that extend beyond immediate reality is fundamental to human thinking.

By envisioning different scenarios, we can gain insights into how actions influence what lies ahead without needing to experience every possibility firsthand. In ML, the endeavor to equip machines with the ability to imagine and generate new ideas is called generative modeling. A generative model aims to capture the process by which the data are generated, and it can produce new instances of data similar to those in the training set. Formally, a generative model learns a probability distribution over its data. The key idea is to utilize this learned distribution to sample or generate new data and incorporate information.^[24]

In contrast, discriminative modeling estimates the likelihood that an observation falls within a specific category (Figure 1). Whereas generative modeling does not concentrate on evaluating observations, the approach estimates the probability of encountering observation x . A key concept in generative modeling is the sample space, which refers to the complete set of all values an observation x can take. To describe the likelihood of different outcomes in this space, the probability density function, denoted as $p_{\theta}(x)$, assigns probabilities to continuous variables based on model parameters θ . Since there are infinitely many density functions that can be used to estimate the real distribution of the data, parametric modeling structures the approach in



Deepalaxmi Rajagopal received her Master of Science degree in Computational Materials Science in 2020 from the TU Bergakademie Freiberg. She is now a doctoral researcher at the Karlsruhe Institute of Technology under the supervision of Professor Britta Nestler. Her research interests primarily focus on the application of generative deep learning frameworks for advanced battery material design.



Adrian Cierpka received his Master of Science degree in Computer Science in 2023 from the Karlsruhe Institute of Technology (KIT). Since 2024 he is a Ph.D. student at KIT under the supervision of Professor Britta Nestler. In his research, he focuses on machine learning for Battery Science and AI-assistants in research data management.



Britta Nestler is a full professor (W3) of Microstructure Simulation in Materials Engineering at the Department of Mechanical Engineering, Karlsruhe Institute of Technology (KIT). She serves as Director of the Institute of Applied Materials (IAM-MMS) and leads the Department of Microstructure Simulation at the Institute of Nanotechnology (INT), KIT. In addition, she has been Director of the Institute of Digital Materials Science (IDM) at Karlsruhe University of Applied Sciences. She studied physics and mathematics at RWTH Aachen University, where she also earned her doctorate. Her research focuses on phase-field modeling of multicomponent and multiphase materials, multiphysics and multiscale microstructure simulation, high-performance materials simulation, and advanced data processing and analysis.



Dr. Arnd Koepp received his Ph.D. from the Institute of General Mechanics at RWTH Aachen University. His doctoral research focuses on the application of artificial intelligence to computational mechanics, including deep learning models as surrogate models, physics-explaining material models, and intelligent meta-elements. He is currently the group leader for Data-driven Modeling and Artificial Intelligence at the Institute of Applied Materials (IAM-MMS) of the Karlsruhe Institute of Technology (KIT). His current research interests focus on integrating predictive, generative, and agentic AI and data management into materials sciences.

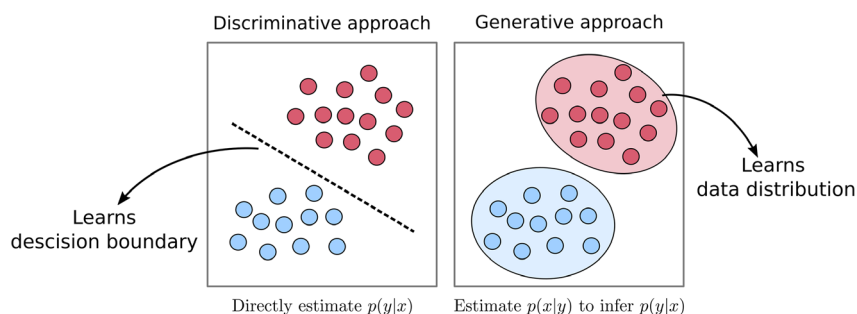


Figure 1. Visual representation of the discriminative and generative modeling approach: the discriminative modeling focuses on finding the decision boundary that separates the different kinds of data instances by directly estimating the $p(y|x)$, whereas the generative models capture the probability distribution of data instances by estimating $p(x|y)$ to infer $p(y|x)$ in the case of a classification task.

finding a suitable $p(x)$. Assuming a simple, learnable distribution, such as the Gaussian, the parameters that best describe the observed data can be learned. The likelihood measures how well a given set of parameters explains the observed data, expressed as $p_\theta(x)$, which represents the probability of data x under the model parameters θ . The likelihood function can be expressed as,

$$\mathcal{L}(\theta|x) = p_\theta(x) \quad (1)$$

The likelihood of θ given an observed point x is defined as the value of the density function, which is parameterized by θ , at the point x . For an entire dataset \mathbf{X} consisting of independent observations, the likelihood function is formulated as,

$$\mathcal{L}(\theta|\mathbf{X}) = \prod_{x \in \mathbf{X}} p_\theta(x) \quad (2)$$

In likelihood calculations of complex datasets, multiplying numerous small probabilities can lead to numerical underflow, where the values become so small that they are eventually treated as zero in computations. To mitigate this issue, taking the logarithm of the likelihood is a helpful strategy, as it transforms the products of probabilities into sums, which are easier to handle numerically. The log likelihood is given as

$$\ell(\theta|\mathbf{X}) = \sum_{x \in \mathbf{X}} \log p_\theta(x) \quad (3)$$

To optimize these parameters, the maximum likelihood estimate is used to find the values of $\hat{\theta}$, the set of parameters θ of a density function $p_\theta(x)$ that maximizes the likelihood function, ensuring the model best explains the observed data. Mathematically, the maximum likelihood estimate $\hat{\theta}$ is given as

$$\hat{\theta} = \arg \max_{\theta} \ell(\theta|\mathbf{X}) \quad (4)$$

Using maximum likelihood estimation in generative modeling necessitates analyzing feature interactions, which can become exceedingly complex and may grow exponentially with the increasing number of features. As a result, this approach tends to assign a constant probability to points not observed in the original dataset. In contrast, the Naive Bayes-based parametric

model assumes that the features are conditionally independent, which means it requires fewer parameters to estimate. This makes Naive Bayes much more efficient and scalable. However, this simplification also means that Naive Bayes struggles to capture the intricate relationships between features. For example, the Naive Bayes approach fails to capture the correlation between pixels in images, and sampling each pixel of the image independently results in a large sample space for high-dimensional features. To solve this challenge, deep learning (DL) uses generative modeling.^[17,25] DL can uncover complex, nonlinear relationships between features by processing information through multiple-layer abstractions without making prior assumptions.

2.2. Generative DL

DL is a branch of ML that uses neural networks with multiple layers to enhance the way machines learn from data. The neural networks consist of layers of interconnected nodes that learn specific features of the input data. DL can transform information from a basic level to more complex, abstract representations using these layered structures. This process allows the model to identify and understand patterns within raw data, making it particularly effective. Each neural network layer is defined by its own set of weights and biases. The process of identifying the optimal weights that lead to the most accurate predictions is known as training. Throughout this training process, the network's predictions are compared to the ground truth using a loss function selected according to the prediction type. The calculated loss is then back-propagated through the network to adjust the weights incrementally in a direction that improves prediction accuracy. Different neural network architectures are developed to handle different types of input data. Convolutional Neural Networks are used for image-based data due to their ability to capture local spatial patterns, whereas Recurrent Neural Networks are used to handle sequential data where temporal dependencies matter.^[26,27]

Integrating generative modeling techniques with DL enables the extraction of higher-level features through representation learning using a deep neural network, and learning the underlying distribution $p(x)$ in the case of unlabeled data or $p(x,y)$ in the case of labeled input data, using generative modeling. With the generative approach, the model not only learns to make

decisions but also quantifies these decisions in terms of the learning environment using probability theory. Generative models utilize Bayes' theorem to infer $p(y | x)$ from $p(x, y)$, which is then employed for the generation of information-rich data. Unlike discriminative models, which directly estimate $p(y | x)$, generative models learn the prior probability $p(y)$, representing the probability of a conditional variable before observing any data. The generative models also facilitate the estimation of the likelihood $p(x | y)$, which describes how the observed data x is generated given a class or latent variable y . To perform inference in generative tasks, the posterior probability $p(y | x)$ is required. The posterior probability reflects updated belief about y after observing x . Although the posterior probability cannot be computed directly, it can be calculated using Bayes' theorem.

$$p(y | x) = \frac{p(x, y)}{p(x)} = \frac{p(x | y)p(y)}{p(x)} \quad (5)$$

where $p(x)$ is the evidence that represents the marginal probability of observing x for all the possible values of y . The evidence $p(x)$ is often intractable in high-dimensional space because it requires integrating all the possible values of y , making it impossible when y has an infinite or large discrete space. To solve this, most generative models employ approximation techniques, such as variational inference (VI) or Markov chain Monte Carlo estimators, to effectively estimate the posterior distribution without computing the exact evidence. Understanding the data distribution $p(x)$ from a generative perspective unlocks a broad range of applications, including sampling and generation of new battery material properties with target performance. Generative models facilitate data compression by utilizing learned distributions to extract important latent features that support various tasks, including clustering and feature extraction. Additionally, generative models provide valuable assistance in data augmentation by creating synthetic samples, which can enhance model generalization, especially when data are scarce.

In the context of battery material analysis, generative models are used to simulate and optimize material properties, predict electrode degradation patterns, reconstruct high-resolution microscopy images from low-quality scans, accelerate the

discovery of new energy storage materials, and improve battery lifespan prediction. Next, this article will explore the fundamental principles underlying several prominent generative models used in battery research. Specifically, the review will focus on variational autoencoders (VAEs), generative adversarial networks (GANs), diffusion models (DMs), and autoregressive (AR) models.

2.3. VAEs

A VAE is a DL architecture designed to generate new data that is similar to the training dataset. To understand VAEs, it is essential to first understand autoencoders. Autoencoder learns efficient data representations by compressing and reconstructing input data. The fundamental architecture of an autoencoder comprises two primary components: the encoder and the decoder. The role of the encoder is to compress the input data into a low-dimensional embedding, which serves as a latent representation of the original input. In contrast, the decoder reconstructs the original input data from the learned lower-dimensional representation of the data. The training objective of autoencoders is to minimize the difference between the input data and its reconstructed output. Autoencoders produce distinct encodings for each type of image in the latent space, simplifying the decoding process. This characteristic makes them practical for image compression and regeneration. However, their limitations become evident when creating a generative model. In such cases, the goal is not merely to replicate the input data but to randomly sample from the latent space or generate variations of an input image within a continuous latent space.^[28–31] If the latent space exhibits discontinuities, creating different variations of the input data, the decoder may yield unrealistic outputs. This issue arises because the decoder lacks information about those areas within the latent space. In VAEs, the encoder maps the input data to a lower-dimensional latent space, but rather than producing a single point, it generates a probabilistic distribution from which a new data point can be sampled. This probabilistic approach enables VAEs to learn a more structured and continuous representation in the latent space, which is beneficial for generative modeling and data synthesis. As illustrated in Figure 2, an autoencoder compresses and

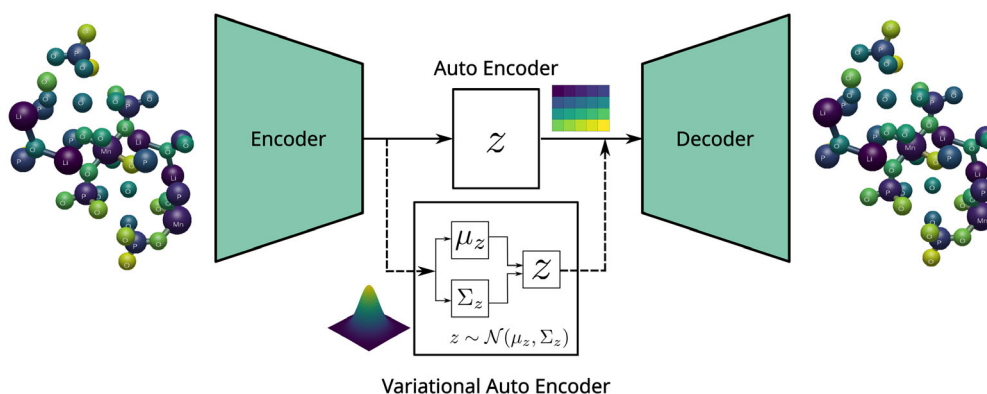


Figure 2. A schematic illustration of an autoencoder and a VAE. The autoencoder transforms input data into a compressed latent vector, which is then reconstructed back into the original data. In contrast, the VAE encodes the input into the parameters of a statistical distribution, resulting in a continuous latent representation where μ_z denotes the mean and Σ_z indicates the variance of the learned distribution.

reconstructs data through a latent vector, while a VAE encodes data into a continuous latent distribution defined by its mean and variance. Following the intuition of autoencoders, suppose that some latent variable \mathbf{z} generates an observation \mathbf{x} . To generate the output \mathbf{x} , the parameters of \mathbf{z} are inferred. In other words, $p(\mathbf{z} | \mathbf{x})$ is computed using Bayesian rule,

$$p(\mathbf{z} | \mathbf{x}) = \frac{p(\mathbf{x} | \mathbf{z}) p(\mathbf{z})}{p(\mathbf{x})} \quad (6)$$

For a continuous distribution, the integral is taken over all sample partitions, as given below.

$$p(\mathbf{x}) = \int p(\mathbf{x} | \mathbf{z}) p(\mathbf{z}) d\mathbf{z} \quad (7)$$

As mentioned in the Section 2.2, the integral in (Equation 7) is hard to evaluate with respect to its parameters. Especially for large data sets, the (Equation 7) requires exponential time to calculate the integrals, as it must be evaluated over all configurations of latent variables and is intractable. To solve this problem, approximation techniques are required. The posterior can be approximated $p(\mathbf{z} | \mathbf{x})$ using a family of tractable distributions $q(\mathbf{z} | \mathbf{x})$, known as an inference model. The parameters of $q(\mathbf{z} | \mathbf{x})$ are chosen to be similar to $p(\mathbf{z} | \mathbf{x})$, enabling us to perform approximate inference for the intractable distribution. A distribution is tractable if any derived marginal probability can be calculated in linear time. The primary goal of the VAEs is to infer $p(\mathbf{z})$ using $p(\mathbf{z} | \mathbf{x})$, thereby aligning the latent variable with the observed data to generate valid outputs. To infer the unknown $p(\mathbf{z} | \mathbf{x})$, a method called VI is used, which is a key technique in Bayesian inference. VI approaches inference as an optimization problem, approximating the true distribution $p(\mathbf{z} | \mathbf{x})$ with a simpler Gaussian distribution and minimizing the divergence

between the two distributions using the Kullback–Leibler (KL) divergence metric. The KL divergence is a non-negative and asymmetric measure, and it is given by

$$D_{KL}(q(\mathbf{x}) || p(\mathbf{x})) = \sum_{\mathbf{z}} q(\mathbf{z} | \mathbf{x}) \log \left(\frac{q(\mathbf{z} | \mathbf{x})}{p(\mathbf{z} | \mathbf{x})} \right) \quad (8)$$

After all the mathematical derivations,^[32] the variational objective function or the loss function of VAE can be written as

$$L = \mathbb{E}[\log p(\mathbf{x} | \mathbf{z})] - D_{KL}[q(\mathbf{z} | \mathbf{x}) || p(\mathbf{z})] \quad (9)$$

The term $\mathbb{E}[\log p(\mathbf{x} | \mathbf{z})]$ represents the reconstruction likelihood, and $D_{KL}[q(\mathbf{z} | \mathbf{x}) || p(\mathbf{z})]$ ensures that the learned distribution q is similar to the true prior distribution p . The term $q(\mathbf{z} | \mathbf{x})$ refers to the encoder of the VAE, also known as the inference model. The encoder is used to compute the latent parameters based on the input data. In contrast, $p(\mathbf{x} | \mathbf{z})$ represents the decoder, or generator, which maps the learned latent parameters back to the original input space to generate new data samples.^[33,34]

2.4. GAN

GANs constitute a notable advancement in the field of DL, particularly in the realm of generating high-quality and diverse datasets. As shown in **Figure 3**, a GAN comprises two integral components: a generator network, which is tasked with producing plausible data, and a discriminator network, which evaluates and distinguishes between original data and the synthetic outputs generated by the generator.^[35] The discriminator imposes penalties on the generator for generating implausible results. At the beginning of GAN training, the generator produces fake data, and the discriminator can quickly learn to distinguish between real and fake data. As the training progresses, the

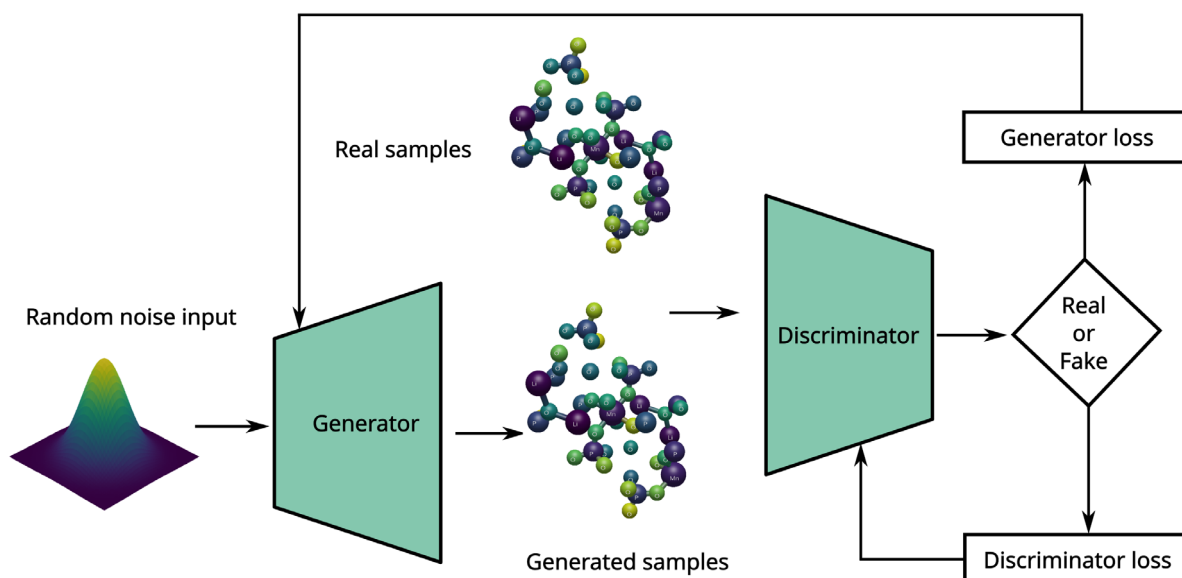


Figure 3. A schematic representation of a GAN. The generator network synthesizes realistic data from random noise vectors, while the discriminator distinguishes between authentic samples from the training dataset and synthetic samples produced by the generator. Both networks are trained in an adversarial manner, enhancing each other's performance over time.

generator learns to produce outputs that are increasingly closer to real data, while attempting to deceive the discriminator. At the end of the training, if the generator's training is successful, the discriminator becomes less effective at distinguishing between real and fake data. Both the generator and the discriminator are neural networks. Consider a dataset in d dimensions consisting of n data points, represented as $\{\mathbf{x}_i \in \mathbb{R}^d\}_{i=1}^n$, which are derived from a probability distribution $p(\mathbf{x})$. The objective is to generate new data points that closely resemble those in the original dataset. A common approach involves minimizing the distance between points to enhance their similarity. Since the generated points must belong to the same distribution as those in the original dataset, regardless of their distances, the distance between two points is not a suitable metric for this problem. Thus, the most effective measure of similarity in this case is the comparison of probability distributions to identify a probability distribution, denoted as $q(\mathbf{x})$, that closely approximates $p(\mathbf{x})$. This task is more complex than it initially appears because $p(\mathbf{x})$ remains unknown. To estimate $p(\mathbf{x})$, the GAN begins with a random noise, $\mathbf{z} \approx p_z(\mathbf{z})$, defined over \mathbb{R}^m . A common approach for modeling this noise is to assume it follows a Gaussian distribution. The goal is to discover an optimal mapping $G: \mathbb{R}^m \rightarrow \mathbb{R}^n$, such that if a random variable $\mathbf{z} \in \mathbb{R}^m$ is sampled from the probability distribution $p_z(\mathbf{z})$, then $G(\mathbf{z})$ has a distribution p_g which needs to be a good approximation of original distribution $p_{\text{data}}(\mathbf{x})$. To facilitate the realistic generative process, a discriminator function $D(\mathbf{x})$ is employed to categorize the outputs of $G(\mathbf{z})$ as either real or synthetic. In essence, $D(\mathbf{x})$ differentiates between the actual data $p(\mathbf{x})$ and the generated data $G(\mathbf{z})$, prompting the generator to modify its parameters accordingly. The interaction between generator and discriminator can be mathematically formalized using a min-max function as follows:

$$\min_G \max_D V(D, G) = \mathbb{E}_{\mathbf{x} \sim p_{\text{data}}(\mathbf{x})} [\log(D(\mathbf{x}))] + \mathbb{E}_{\mathbf{z} \sim p_z(\mathbf{z})} [\log(1 - D(G(\mathbf{z})))] \quad (10)$$

Here, the function $V(D, G)$ consists of two terms that need to be optimized. $\mathbb{E}_{\mathbf{x} \sim p_{\text{data}}(\mathbf{x})} [\log(D(\mathbf{x}))]$ represents the average log probability produced by discriminator when input is real and $\mathbb{E}_{\mathbf{z} \sim p_z(\mathbf{z})} [\log(1 - D(G(\mathbf{z})))]$ represents the average log probability produced by the discriminator when the input is generated.^[36] During the training of the discriminator, it is necessary to freeze the weights of the generator network, allowing errors to be back-propagated solely to update the discriminator. Conversely, during the generator training, the weights of the discriminator network are frozen, allowing errors to be backpropagated and improving the generator's performance. The alternate training process of GAN enables the continuous improvement of both the generator and discriminator models throughout each training step. Training should conclude upon reaching the Nash Equilibrium, characterized by $D(\mathbf{x}) = 0.5$ for all \mathbf{x} . This condition is attained when the generated points are so realistic that they become nearly indistinguishable from real data.^[37–39]

2.5. DMs

DMs are inspired by nonequilibrium thermodynamics.^[40] For example, consider adding a drop of red dye to a beaker of water. According to the law of physics, the particle diffuses throughout the water continuum until the system reaches equilibrium. DMs are defined as a class of deep generative models that learn to reverse the process that gradually destroys the structure of the given data distribution. As depicted in the **Figure 4**, the training of DMs consists of two phases: the forward and reverse diffusion processes. The forward diffusion process introduces noise

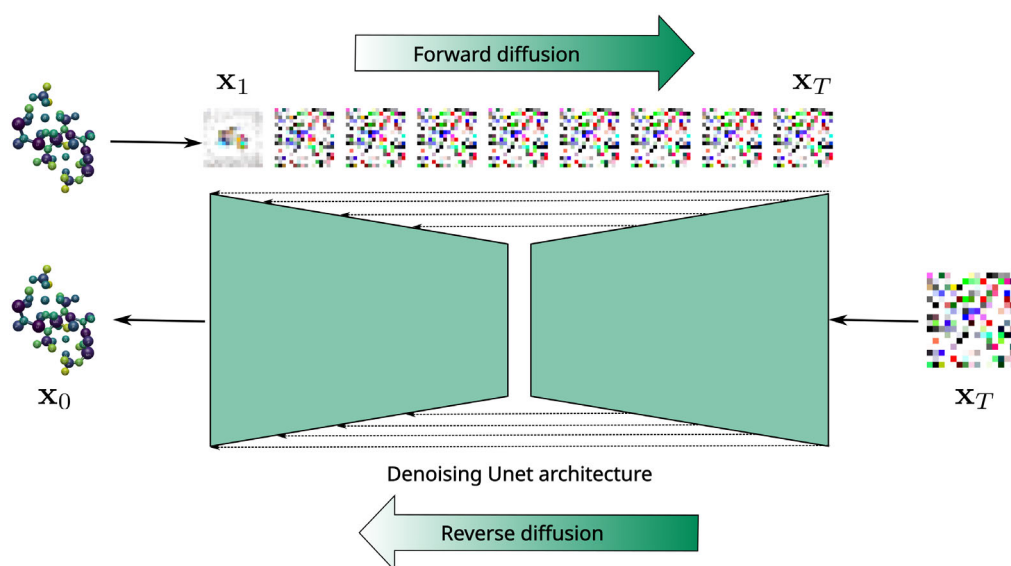


Figure 4. A visualization of a DM architecture showing the forward and reverse processes. In the forward process, Gaussian noise is added progressively to a clean data sample over a series of timesteps, transforming it into pure noise according to a fixed noise schedule. The reverse process involves a neural network, such as a U-Net, trained to predict and remove the added noise at each timestep. By iteratively denoising from random noise, the model learns to generate realistic data samples that closely resemble the original data distribution.

to the training data over a series of time steps, with the noise scale varying linearly with time. This is achieved by using linear noise scheduling at each step, until the training data is corrupted, resulting in pure Gaussian noise. In the forward diffusion process, noise is introduced using a Markov chain, indicating that the current state of the training data relies solely on its most recent state. Let $q(\mathbf{x}_0)$ be the probability density of the training data, where the index 0 denotes the data before adding any noise. Given an uncorrupted training sample $\mathbf{x}_0 \sim q(\mathbf{x}_0)$, the noised versions $\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_T$ are generated through the following Markovian process

$$q(\mathbf{x}_t | \mathbf{x}_{t-1}) = \mathcal{N}(\mathbf{x}_t; \mu = \sqrt{1 - \beta_t} \mathbf{x}_{t-1}, \Sigma = \beta_t \mathbf{I}) \quad \forall t \in \{1, \dots, T\}. \quad (11)$$

Here, T is the number of diffusion steps, β_1, \dots, β_T are the hyperparameters controlling the noise variance, and \mathbf{I} is the identity matrix with dimension equal to the input data dimension. This setup enables the sampling of \mathbf{x}_t when t is drawn from a uniform distribution $t \sim \mathcal{U}(\{1, \dots, T\})$:

$$q(\mathbf{x}_t | \mathbf{x}_0) = \mathcal{N}\left(\mathbf{x}_t; \mu = \sqrt{\hat{\beta}_t} \mathbf{x}_0, \Sigma = (1 - \hat{\beta}_t) \mathbf{I}\right) \quad (12)$$

where $\hat{\beta}_t = \prod_{i=1}^t 1 - \beta_i$. This shows that, given the original data \mathbf{x}_0 and variance schedule β_t , the noisy sample \mathbf{x}_t can be derived in a single step. The reparameterization trick allows sampling from $q(\mathbf{x}_t | \mathbf{x}_0)$ as follows

$$\mathbf{x}_t = \sqrt{\hat{\beta}_t} \cdot \mathbf{x}_0 + \sqrt{(1 - \hat{\beta}_t)} \cdot \mathbf{z}_t \quad (13)$$

where \mathbf{z}_t is a standard normal variable, enabling the faster sampling of the noisy version from the original sample. In the reverse diffusion process, the goal is to remove the noise added to the training dataset in a structured and controlled manner, thereby reconstructing the original data \mathbf{x}_0 . For example, if forward diffusion process can be reversed and sample from $q(\mathbf{x}_{t-1} | \mathbf{x}_t)$, will be able to recreate the original sample before adding noise from Gaussian noise input, $\mathbf{x}_T \sim \mathcal{N}(0, \mathbf{I})$. However, estimating $q(\mathbf{x}_{t-1} | \mathbf{x}_t)$ is difficult because it requires the use of the entire dataset. To achieve the reverse diffusion process, a neural network model p_θ is trained to approximate these conditional probabilities. The model learns to reverse this diffusion process during training, generating new data. Starting with pure Gaussian noise $p(\mathbf{x}_T) := \mathcal{N}(\mathbf{x}_T; 0, \mathbf{I})$, the model learns the reverse trajectory or joint distribution $p_\theta(\mathbf{x}_{0:T})$ as

$$p_\theta(\mathbf{x}_{0:T}) = p(\mathbf{x}_T) \prod_{t=1}^T p_\theta(\mathbf{x}_{t-1} | \mathbf{x}_t), \quad (14)$$

with, $p_\theta(\mathbf{x}_{t-1} | \mathbf{x}_t) = \mathcal{N}(\mathbf{x}_{t-1}; \mu_\theta(\mathbf{x}_t, t), \Sigma_\theta(\mathbf{x}_t, t))$

where $p_\theta(\mathbf{x}_{0:T})$ denotes the reverse diffusion trajectory. The reverse diffusion kernel $p_\theta(\mathbf{x}_{t-1} | \mathbf{x}_t)$ is defined by the mean $\mu_\theta(\mathbf{x}_t, t)$ and the covariance matrix $\Sigma_\theta(\mathbf{x}_t, t)$. Using the Markov chain, \mathbf{x}_0 is generated by first sampling a noise vector $\mathbf{x}_T \sim p(\mathbf{x}_T)$, then iteratively sampling from the learnable reverse

diffusion kernel $\mathbf{x}_{t-1} \sim p_\theta(\mathbf{x}_{t-1} | \mathbf{x}_t)$ until $t = 1$. The sampling process is improved by training the reverse Markov chain to match the forward Markov chain. In other words, the parameter θ has to be adjusted so that the joint distribution of the Markov chain closely approximates that of the forward process. Looking back at the explanation of the VAE in Section 2.3, the combination of q and p is similar to the formulations used in the VAE.^[41] After the necessary parameterization of the reverse diffusion process following Ho et al.,^[42] the objective formulation to train the diffusion model

$$\mathcal{L}_{\text{simple}} := \mathbb{E}_{\mathbf{x}_0, \mathbf{z}} [\|\mathbf{z} - \mathbf{z}_\theta(\sqrt{\alpha_t} \mathbf{x}_0 + \sqrt{1 - \alpha_t} \mathbf{z}, t)\|^2] \quad (15)$$

where \mathbb{E} refers to the expected value and \mathbf{z}_θ represents the neural network trained to predict the noise in given input \mathbf{x}_t . The α_t is calculated from $\alpha_t = 1 - \beta_t$ and $\bar{\alpha} = \prod_{s=0}^t \alpha_s$. The reparameterization of reverse diffusion, $p_\theta(\mathbf{x}_{t-1} | \mathbf{x}_t)$, simplifies the process by fixing the covariance at a constant value. The mean is then defined through variance scheduling, which denotes that it only depends on \mathbf{x}_t . Consequently, the neural network is trained to predict the noise added at each specific time step, rather than estimating both the mean and covariance.^[42–44] The main drawbacks of DMs include low inference speed and high training cost. These drawbacks can be mitigated by using the latent diffusion model (LDM), which involves diffusion in the latent space. Here, an autoencoder is used to encode the data space into a lower-dimensional latent space. The diffusion process is then implemented in the latent space, which makes the model more computationally efficient while capturing the high-frequency and imperceptible features of the training data. The LDM improves inference speed without compromising the quality of the prediction.^[45]

2.6. AR Models

AR models are among the most powerful generative models, representing current state-of-the-art architectures in likelihood-based image modeling, and serve as the basis for large language generation models, such as Generative Pretrained Transformer (GPT). AR models factorize the joint distribution over high-dimensional data by decomposing it into a product of conditionals.^[46–49] Consider a dataset \mathcal{D} of n -dimensional binary datapoints $\mathbf{x} \in \{0, 1\}^n$. The joint distribution can be factorized using the chain rule of probability as,

$$p(\mathbf{x}) = \prod_{i=1}^n p(x_i | x_1, x_2, \dots, x_{i-1}) = \prod_{i=1}^n p(x_i | \mathbf{x}_{<i}) \quad (16)$$

where $\mathbf{x}_{<i} = [x_1, x_2, \dots, x_{i-1}]$ denotes variable preceding x_i in a fixed ordering. Autoregression is a statistical method used in time-series analysis that assumes that the current values of a time series are a function of its past values. Similarly, an AR model determines the probabilistic correlation between elements in a sequence and uses this knowledge to predict the next element in an unknown sequence, as illustrated in the Figure 5. The order of an AR model is the number of preceding values in the series

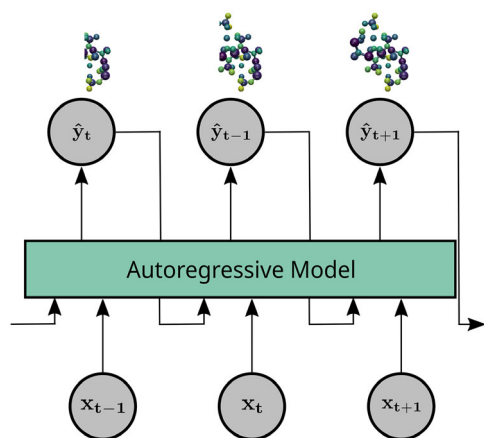


Figure 5. Workflow of an AR model, illustrating the sequential prediction process where each subsequent output substructure is generated from previously produced substructures and the model's learned parameters. Predictions are iteratively fed back as inputs until a complete sequence is generated.

used to predict the current value. If x_t is regressed only on x_{t-1} , it is a first-order autoregression, which is written as AR(1). Mathematically, an AR model of order p , denoted as AR(p), can be expressed as:

$$x_t = c + \phi_1 x_{t-1} + \phi_2 x_{t-2} + \dots + \phi_p x_{t-p} + \epsilon_t \quad (17)$$

where x_t is the value of the variable at time t , c is an intercept term that allows the time series to have a nonzero mean, ϕ_i are the AR coefficients, ϵ_t is a white noise error term assumed to have zero mean and constant variance, and ϕ is the lag order. The AR model assumes stationarity, meaning the statistical properties such as mean and variance of the time series do not change over time. Changing the parameters ϕ_1, \dots, ϕ_p results in different time series patterns. The variance of the error term ϵ_t will only be the scale of the series, not the patterns. While the chain rule of probability allows for exact representation of any distribution, the tabular specification of each conditional $p(x_i | x_{<i})$ is computationally intractable because for a n dimensional dataset, the conditional probability for the last dimension $p(x_n | x_{<n})$ require estimation of probability for 2^{n-1} configurations of the variables x_1, x_2, \dots, x_{n-1} and require $2^{n-1} - 1$ parameters. The exponential growth in the number of parameters makes tabular representation of the conditionals impractical for learning and inference. The AR model overcomes the problem of intractability by parameterizing the conditionals with neural networks. For example, consider the case of a neural network with one hidden layer. The mean function for variable i can be expressed as,

$$\begin{aligned} h_i &= \sigma(A_i x_{<i} + c_i), \\ f_i(x_1, x_2, \dots, x_{i-1}) &= \sigma(a^{(i)} h_i + b_i) \end{aligned} \quad (18)$$

where $h_i \in \mathbb{R}^d$ denotes the hidden layer activation for the neural network and $\sigma(\cdot)$ is the sigmoid function to restrict the output between 0 and 1. $a_j^{(i)} \in \mathbb{R}$ represent weights of the network and d is the hidden layer size. Integrating the neural

network with the AR model reduces the complexity to $O(n^2 d)$. The AR model is trained using maximum likelihood estimation, which maximizes the log-likelihood of the data under the model. Training a generative model involves optimizing the closeness between the model distribution p_θ and the data distribution p_{data} , typically measured using KL divergence:

$$\min_{\theta \in \mathcal{M}} KL(p_{\text{data}} || p_\theta) = \mathbb{E}_{x \sim p_{\text{data}}} [\log p_{\text{data}}(x) - \log p_\theta(x)] \quad (19)$$

Since p_{data} is fixed, minimizing the KL divergence is equal to maximizing the log-likelihood $\max_{\theta \in \mathcal{M}} \mathbb{E}_{x \sim p_{\text{data}}} [\log p_\theta(x)]$ of the data. The $\log p_\theta(x)$ refers to the log-likelihood of the datapoint x with respect to model distribution p_θ . In practice, this expectation is approximated using a dataset \mathcal{D} samples from p_{data} , yielding the maximum likelihood estimation objective as

$$\mathcal{L}(\theta | \mathcal{D}) = \max_{\theta \in \mathcal{M}} \frac{1}{|\mathcal{D}|} \sum_{x \in \mathcal{D}} \log p_\theta(x) \quad (20)$$

In the context of generative modeling, the parameterization of the conditional distributions in AR generative models can be implemented through different architectures, each designed to enhance the model capacity, receptive field, and memory. One of the simplest forms is the fully visible sigmoid belief network without a hidden layer, which uses functions as a linear combination of inputs followed by a sigmoid function to constrain the outputs between 0 and 1. The number of parameters of the FVSN model is $O(n^2)$. For computationally efficient parameterization of the AR generative model, the neural AR density estimator (NADE)^[50] introduces a hidden layer followed by a sigmoid function where parameters are shared across the functions for evaluating the conditionals, which reduces the order of parameters from $O(n^2 d)$ to $O(nd)$. The masked autoencoder for distribution estimation builds upon NADE by incorporating masking strategies in the standard autoencoder to preserve the AR property, ensuring each output dimension is conditioned only on previous outputs in a fixed ordering.^[51] AR image modeling progressed with the development of PixelRNN^[52] and PixelCNN^[53] by modeling spatial dependencies at the pixel level. PixelRNN captures long-range dependencies using recurrent Long Short-Term Memory (LSTM) layers, but trains slowly due to its sequential nature. PixelCNN replaces recurrence with masked convolutions that condition each pixel on previously generated ones, allowing fully parallel training while preserving the spatial structure of the image. The pixel-level autoregression is limited by high computational costs, particularly for high-resolution generation. Transformer-based AR models address this by using self-attention, which can consider all parts of an input at once.^[54] GPT follows this approach, predicting each next element in a sequence based on everything that has been seen before. Trained on enormous amounts of unlabeled text and images through a process called generative pretraining to identify patterns that can be applied to new inputs.^[55] The self-attention mechanism of the transformer architecture enables the model to process all portions of the input sequence in parallel.

3. Multimodal Data in Battery Research

Multimodal data is a collection of data that consists of information captured in various formats combined to draw insights about a specific domain.^[56] For example, in battery research, comprehending the performance of electrodes necessitates the application of multiple characterization techniques. These techniques include scanning electron microscopy, transmission electron microscopy, X-ray computed tomography, atomic force microscopy, time-of-flight secondary mass spectroscopy, and electron backscatter diffraction.^[57,58] Together, these methods help reveal the morphological, atomistic, mechanical, crystallographic, and chemical properties that influence the electrochemical performance of the electrodes. Multimodal data in battery research plays an important role in understanding the underlying, complex, and multiscale information of battery systems.

Improving battery performance requires a thorough understanding of how the various components of a battery interact and the properties of the materials used in it. To achieve optimal performance from a battery, how these properties affect it at various scales and over time has to be understood. The multiscale simulation and experiments yield heterogeneous data, which can be analyzed to determine how factors such as material composition, electrode microstructure, and solid electrolyte interphase (SEI) effects impact battery performance.^[16] Even though these characterization techniques provide the required information, the destructive nature of some of these techniques limits the acquisition of multimodal data from the same sample.

Several stochastic-based methods address these challenges by creating representative structures of battery material with information, for example, from two different characterization techniques.^[59,60] However, the effectiveness of these stochastic techniques is only extensible to a single type of material and cannot be generalized to a wide range of materials.^[61]

With the advent of ML-based techniques, a wider range of applications is now possible, allowing for the development of material-agnostic systems that require less expert fine-tuning. The quality and quantity of data play important roles in the performance and application of the aforementioned generative learning methods. High-quality datasets are essential for training robust models that accurately predict material properties and battery performance. Although the amount of data required for a ML study varies depending on the specific algorithm, data preparation methods, and the quality of the relevant information, generally, more data availability typically leads to improved ML

models.^[62,63] The advantages and drawbacks of different generative models are summarized in **Table 1** to address the applicability of these models in the different contexts of battery research.

Thus, the multiscale nature of battery research necessitates the development of a unified dataset to comprehend the comprehensive spatial description of all influencing material properties, where each property is acquired at different length scales of battery material characterization. To achieve this, deep generative models can be utilized to make the process purely data-driven, capturing the interrelationships between different modalities. This data-driven methodology helps to uncover characteristics of data that can explain the behaviors of a battery system from manufacturing to the end of life.^[64]

4. Generative Learning for Advanced Battery Material Analysis

The design and discovery rate of advanced structural and functional materials significantly influence the pace of technological innovation and addresses challenges in sustainable energy storage and conversion. The screening of novel battery materials based on traditional experimental methods is often time-consuming and resource-intensive. Recent advancements in high-throughput screening methods,^[65] open material databases,^[10,66,67] ML-based property predictors,^[68] and ML force fields^[69] have enabled the screening of hundreds of thousands of materials, representing only a tiny fraction of the possible stable materials. Generative models offer a complementary inverse design strategy, enabling the creation of materials with desired properties. The inverse design strategy of generative models maps the relationships between structure and properties by encoding the high-dimensional material space into an information-rich continuous latent space. The information-rich latent space enables the generation of new materials based on the embedded information from the encoding process. The primary advantage of generative models is their ability to design or discover new materials with desired properties, filling gaps between existing materials by learning their distribution in a continuous space.^[70,71]

In the context of battery materials, the crystal structures, molecular interphase design, and microstructure morphologies of the material dictate the electrochemical performance of the cell throughout its cycle life. Understanding the

Table 1. Comparison of advantages and drawbacks of generative DL models.

Model	Advantages	Drawbacks
VAEs	Stable training; structured latent space suitable for interpolation and data exploration; effective in anomaly detection and representation learning.	Blurry reconstructions and difficulty capturing fine-grained details in complex data.
GANs	High-resolution, realistic samples; fast inference once trained.	Training instability and prone to mode collapse.
DMs	Stable training; high-quality and diverse samples; suitable for complex distributions.	Computationally expensive inference due to iterative sampling.
ARs	Stable training; high-quality samples; versatile across different data modalities and tasks.	Slow inference; requires large datasets and computational resources.

structure-function relationship is critical in defining the operational limitations and degradation pathways. The design and discovery of the materials that constitute the components of batteries can be studied in different forms of representations, such as molecular, chemical, and structural space.

4.1. Design of Crystal Structures

The crystal structure of the electrode materials significantly influences battery performance by affecting ion diffusion pathways, electronic conductivity, and electron stability during the charge and discharge cycles. Generative models such as VAEs, GANs, and DMs can create candidate structures that achieve a balance of high ionic conductivity, structural stability, and electrochemical potential. These generative models expedite the search for materials beyond those naturally occurring or known in current chemistry, opening new avenues for next-generation batteries. Integrating first-principles calculations, such as density functional theory (DFT), with generative models helps validate and efficiently screen these AI-designed structures. Noh et al.^[72] developed iMatGen (image-based material generator), which was designed for the inverse design of inorganic solid-state materials using a continuous, invertible representation of crystal structures. The iMatGen framework is composed of two primary steps. In the first step, a standard autoencoder is employed to compress image representations of the unit cell and atomic basis into image fingerprints. In the second step, a VAE maps these image fingerprints into a latent space of material fingerprints. The VAE decoder is then used to regenerate the image fingerprints from this latent space. To encourage the generation of stable materials, a stability classifier is incorporated, which categorizes structures based on their formation energies. This classification is integrated into the VAE loss function, allowing the model to prioritize both known and metastable materials during the sampling process. The iMatGen framework, based on VAE, provides physically invertible image encoding, latent space optimization, and integrated stability classification, offering a robust path for discovering stable materials.

Nouira et al.^[73] introduced the GAN model for crystal structures, a two-step GAN designed to synthesize novel ternary crystallographic structures from binary compound inputs, particularly targeting hydrogen storage materials. It takes POSCAR files representing binary hydrides input, encodes them into tensors, and generates pseudo-binary compositions in the first step. These are then refined through a feature transfer process to separate components from different domains. A second GAN step incorporates geometric constraints based on atomic neighbor distances to produce chemically valid ternary outputs. However, compared to iMatGen, which employs a continuous and invertible latent space enabling property-guided sampling and broader chemical exploration, CrystalGAN lacks crystal invariance, and explicit symmetry awareness, relying instead on geometric constraints.

Zhao et al.^[74] developed a GAN-based framework, CubicGAN, to generate novel cubic crystal structures of ternary material compounds widely used in solar cells and lithium batteries.

While training, the generator of the framework is conditioned with space group representation for the crystal prototype and element embeddings, along with random noise. The generator of the CubicGAN framework generates material structures with corresponding specific space groups and element constituents. A total of 506 new ternary or quaternary material structure prototypes were rediscovered, which are validated by a DFT-based phonon dispersion stability check.

The crystal diffusion variational autoencoder (CDVAE) model developed by Xie et al.^[75] addresses material invariances by learning the distribution of stable materials to generate new periodic structures. The model consists of an encoder that utilizes a graph neural network to effectively learn the relationships between atoms, allowing it to encode complex structural information. In the decoding phase, a noise score network is employed with Langevin dynamics to refine atomic positions and types iteratively, thereby guiding the generation process toward lower energy states. This combination enables CDVAE to significantly outperform iMatGen, CrystalGAN, and CubicGAN in tasks such as reconstruction, generating diverse and realistic materials, and generating materials that optimize to specific properties.

Another diffusion-based model for the inverse design of crystal structures developed by Yang et al.^[76] at Google DeepMind uses UniMat, the unified representation of materials to represent the crystal structure of the materials across the periodic table as an input feature for the diffusion model, which allows the model to generate materials with any number of atoms. UniMat employs a unified representation of crystal structures by leveraging a four-dimensional tensor. This tensor represents the positions of atoms based on their locations in the periodic table, similar to an organized grid where each cell corresponds to a specific element. The denoising diffusion model, during training, learns to move atoms from random locations back to their original positions. Atoms not present in the crystal are relocated to the null location during the denoising process, resulting in crystals with an arbitrary number of atoms. The denoising diffusion model implemented in this work comprises interleaved attention and convolution layers across periods and groups of the periodic table, which allows the model to capture inter-atom relationships and thereby preserve the inductive bias of the groups in the periodic table. By utilizing a convolutional neural network rather than a graph neural network, the model effectively addresses scalability issues that arise with large, complex materials. It has been shown to outperform graph neural network-based crystal diffusion VAE^[75] in terms of state-of-the-art DFT metrics.

MatterGen is a diffusion-based generative model introduced by Microsoft AI4Science that employs a diffusion process specifically designed for crystalline materials. Unlike standard DMs that add Gaussian noise to corrupt data to learn the underlying distribution, MatterGen independently corrupts the input features, such as atom type, coordinates, and the lattice of the crystal structure, while also ensuring that physical constraints, including symmetry and periodicity, are maintained in the obtained noisy distribution. MatterGen is trained in two stages: pretraining phase and fine-tuning phase. In the pretraining phase, the base model, based on an equivariant graph neural network, is trained on a

large and diverse dataset of crystal structures known as Alexa MP 20, comprising 607,683 stable structures with up to 20 atoms. After the initial training, the pretrained model is fine-tuned with the adapter model using small labeled datasets. The fine-tuning adapts the model to generate materials with desired properties. Compared to earlier models, such as crystal DMs, CubicGAN, and CrystalGAN, MatterGen generates materials much closer to the local energy minimum, resulting in better stability, uniqueness, and novelty in the generated structures.^[77]

4.2. Design of Molecules and Interphases

Molecule and interphase design influence key processes such as ion transport, interfacial stability, and redox reactions, all of which are vital for battery efficiency, longevity, and safety. Focusing on molecular level design allows for the creation of electrolytes, additives, and polymeric materials that perform effectively under specific voltage ranges, temperatures, and mechanical stresses while maintaining chemical and electrochemical stability. Achieving better electrolyte and interphase chemistry in batteries requires exploring a vast chemical compound space, which is infeasible without efficient navigation across this chemical space.^[78]

Utilizing generative models in molecular design aims to model the underlying probability distributions of structures and properties and relate them in a nonlinear manner. This generative learning approach facilitates understanding of average structural features and complex patterns in molecular data, which aids in predicting structures that possess the desired properties. These models transform molecules into a continuous space, often referred to as the latent space, where adjustments can be made smoothly and efficiently. In this space, the generation process becomes a controlled optimization, allowing researchers to systematically adjust molecular features, much like tuning ingredients in a recipe until they find a candidate that meets the required functionality. Generative models rely on data representations that capture the inherent properties and structure of molecules. The ideal molecule representations used as input for the inverse design of molecules using generative models must capture all relevant chemical and physical characteristics, encompassing spatial coordinates, bonding patterns, and symmetry aspects, to permit accurate predictions and enable reverse mapping from desired functionalities to actual molecule structures.^[21,79]

Molecular representations can be categorized into discrete, graph-based, and continuous vector representations. Discrete representations encode molecules as sequences of symbols, simplifying complex chemical structures into a format that is machine-readable and comprehensible. One common example is SMILES (Simplified Molecular Input Line Entry System), where atoms and bonds are represented by specific characters, resulting in a unique string for each molecule that does not require explicit three-dimensional coordinates. Continuous representations, like vectors and tensors, capture molecular features in a multidimensional space. Weighted graphs represent molecules, with nodes representing atoms and edges representing bonds, enabling the incorporation of various features such as bond type and charge.^[71,80]

Khajeh et al.^[81] introduced a computational generative framework for discovering polymer electrolytes with high ionic conductivity. The framework comprises three core components: a conditional generative model that proposes potential polymer candidates, a computational evaluation module that assesses their properties through molecular dynamics simulations, and a feedback mechanism that iteratively refines the discovery process. The framework utilizes a conditional generative model based on minGPT (a minimal version of a GPT model) to create polymer electrolyte candidates, guided by desired ionic conductivity properties, using SMILES codes for representation. The model is trained to focus on high ionic conductivity by modifying SMILES strings with class labels for conductivity, ensuring effective guidance during polymer generation. The preprocessing of SMILES involves tokenizing the strings into sequences of tokens, which are then transformed into representations that capture both semantic meaning and positional context. Additionally, the model filters out exact and nonexact duplicates due to symmetries to ensure a diverse training dataset. The framework identified 14 distinct polymer repeating units that exhibited computed ionic conductivity.

Yang et al.^[82] introduced another inverse design framework of polymer electrolytes based on a generative model. They compared two generative models, the minGPT and the diffusion-based model, in generating novel, diverse, and valid polymers using SMILES-based molecule representation. The results demonstrate that the minGPT model outperforms the diffusion-based models in metric assessments based on novelty, uniqueness, validity, synthesizability, similarity, diversity, property distribution replication, and computational efficiency. Additionally, 17 out of 46 generated top polymer candidates show improved ionic conductivity and structural diversity, addressing the challenges of polymer design for energy applications.

Graph-based representations depict molecules as networks, with atoms as nodes and bonds as edges, organizing molecular information in a structured format. This aids in understanding the relationships between chemical structures and properties, making it useful for ML models.^[80] Yoon et al.^[83] proposed a DL framework for designing new electrolyte additives for lithium-ion batteries to reduce both time and cost in the discovery process by conventional methods. The study utilizes a fine-tuned version of the Natural Product Variational Autoencoder to generate novel molecular structures while predicting their key electronic properties: the Highest Occupied Molecular Orbital (HOMO) and the Lowest Unoccupied Molecular Orbital (LUMO). The study uses graph-based molecular structures derived from SMILES strings, which are processed into tree-structured formats with substructure nodes encoded using Extended Connectivity Fingerprints. The HOMO and LUMO values are crucial for determining the oxidative and reductive stability of electrolyte additives, which are required for forming a stable SEI and enhancing battery performance and longevity. The architecture employs a Tree LSTM-based encoder, a multi-layer perceptron decoder, and a regressor that are trained simultaneously to embed structural and property information into the latent space. The model achieved high reconstruction accuracy and generated

1000 new candidate molecules from the latent space through random sampling. The evaluation of generated molecules against metrics such as validity, novelty, and synthetic accessibility showed that these molecules were chemically valid, diverse, and suitable for further synthesis.

Continuous vector-based representations capture essential features of molecules, such as atom types, bonding patterns, and spatial arrangements, using DL models like autoencoders. The continuous representation enables smoother mathematical operations, where small changes in the vector correspond to small alterations in the actual molecule. By embedding molecules in a continuous vector space, ML techniques can efficiently navigate and optimize chemical space. These representations facilitate similarity searches, identifying candidate molecules with properties similar to those of known compounds. In de novo molecular design, continuous vectors can be optimized for specific property objectives and decoded back into valid molecular structures via generative models. This supports inverse design workflows that guide the creation of novel battery materials based on targeted electrochemical properties. Furthermore, analyzing the battery interphase in the latent space can help determine the optimal thickness and porosity for safety and performance. Creating a stable SEI and ensuring interfacial compatibility for achieving stable cycling, particularly in high-voltage or lithium-metal systems. Without optimizing these properties at the interphase level, batteries may face low efficiency, capacity fading, and safety risks, making inverse design a vital tool for discovering optimal formulations for advanced battery systems.^[79,84,85]

4.3. Design and Characterization of Microstructure Morphology

Microscale features of battery materials play a key role in determining the performance and efficiency of electrodes in battery systems. This encompasses the design of electrodes that possess optimal porosity to facilitate efficient ion transport while maximizing surface area and ensuring the mechanical integrity of the battery systems. Microscale enhancement directly impacts the battery's comprehensive performance, energy density, improved thermal management, and safety. The electrode design must strike a careful balance between facilitating ion movement through the pores and ensuring effective electronic conduction within the solid material. The characteristics of the electrode morphology, including the distribution of pore sizes and the tortuosity factor, can influence the flow of ions from the separator to the current collector. This intricate interplay of microstructural features ultimately influences the battery's efficiency and longevity, highlighting the importance of advanced characterization techniques and modeling approaches. Integrating generative learning into the design and characterization of battery microstructures across length scales helps capture the relationships between structure and function for electrodes, generating realistic microstructures to facilitate further electrochemical studies.^[64,86]

Gayon Lombardo et al.^[87] proposed a deep convolutional GAN (DCGAN) to generate realistic three-dimensional microstructural data for lithium-ion battery cathodes, thereby creating synthetic microstructures that closely resemble the real ones and facilitating the design and optimization of battery electrodes. The synthetic microstructures generated by the DCGAN closely match the real microstructures in terms of volume fraction and surface area. Additionally, the implemented framework demonstrated the ability to generate periodic microstructures, which reduces computational cost. This is because periodic structures enable the use of small volumes in simulations while still providing accurate results. Kench et al.^[88] implemented an approach to generate three-dimensional microstructures from two-dimensional images using a generative adversarial neural architecture called SliceGAN. Traditional 2D imaging techniques often fail to capture the volumetric properties necessary for understanding electrochemical processes, such as ion transport and fluid flow within battery electrodes. By utilizing SliceGAN, researchers can generate statistically realistic 3D representations of battery microstructures, which enables more effective simulations of electrochemical and mechanical behaviors critical for optimizing battery performance.

In this direction, Dahari et al.^[89] addressed imaging challenges in battery cathodes, which require high-resolution techniques to capture critical nanoscale features. Effective mesostructure modeling often requires 3D image data with sufficient contrast, resolution, and field of view, a combination rarely achievable with a single imaging technique. To overcome this, Dahari et al.^[89] developed a SuperRes model based on the SliceGAN framework, which generates 3D mesostructures from 2D images. The generator integrates random noise with low-resolution inputs for super resolution, while the discriminator evaluates 2D output slices. The SuperRes model demonstrated accurate mesostructure reconstruction, highlighting the importance of resolution characteristics in differentiating undersampled and undersolved data. Their findings indicate potential advancements in material characterization and optimization, highlighting areas for further refinement in capturing long-range data relationships.

5. Challenges and Future Directions

The integration of generative models in battery research encounters several critical challenges that impede the development of next-generation battery materials: 1) scarcity of high-quality datasets: one of the challenges is the limited availability of high-quality, diverse, and large-scale datasets specifically tailored to battery materials. The experimental battery data are often sparse, heterogeneous, and costly to acquire. This scarcity can result in overfitting, inadequate generalization, and a notable lack of robustness in generative models. The complexity inherent to battery systems, which includes intricate electrochemical and physical processes, underscores the need for comprehensive datasets that effectively capture multiscale interactions and dynamic behaviors; 2) poor model interpretability: generative models often operate as black boxes, making it challenging to

comprehend their decision-making processes and the underlying physical and chemical principles that govern material generation. This lack of interpretability in generative models creates a barrier for experimentalists and engineers, hindering their ability to effectively utilize these models in their research; 3) synthesizability and stability concerns: another significant challenge lies in ensuring that generated materials are not only novel but also synthesizable and stable under real-world operating conditions. One key challenge is to bridge the gap between predictions from generative models and validation through in situ experiments, which is required to translate computational designs into practical battery materials. In contrast, models may propose promising candidates, but their practical effectiveness relies on efficient and accurate experimental synthesis and characterization. The development of closed-loop autonomous experimentation platforms that integrate generative models for design, robotic systems for synthesis, and advanced characterization techniques for validation is still in its early stages of development; 4) high computational resource requirements: the training and deployment of generative models, especially those with large latent spaces and intricate architectures, can be computationally intensive. This requires significant hardware resources and energy, which can be a barrier for researchers who lack access to high-performance computing infrastructure; 5) multiscale complexity: additionally, the performance of batteries is influenced by interactions across multiple length scales, which involve coupled physical phenomena. Consequently, integrating these diverse data types and physical constraints into a unified generative framework is a highly complex implementation.

To address the challenges in realizing the potential of generative learning for battery materials analysis, several key areas have to be explored. Research should focus on developing advanced techniques for generating synthetic data, utilizing active learning, and enhancing transfer learning to address data scarcity. This includes using high-throughput computational simulations to generate extensive synthetic datasets that complement experimental findings. Additionally, establishing strong data standardization and sharing practices within the research community will encourage collaboration and improve data utility. Incorporating key principles of electrochemistry into the design and functions of generative models can significantly improve their interpretability, improve predictive accuracy, and produce battery materials that are both realistic and stable. Future battery research should prioritize the development of automated feedback loops between computational design and experimental validation to enhance the accuracy and efficiency of these processes. Autonomous laboratories equipped with robotic synthesis and characterization capabilities guided by generative models enhance the research process. Such advancements allow for rapid iteration through cycles of material design, synthesis, and testing, significantly expediting the discovery process. The development of explainable artificial intelligence techniques tailored for generative models enhances the understanding of how specific materials are generated and the roles that material features play in achieving desired properties. By focusing on how these models make predictions, researchers can gain valuable

insights into the accuracy of these predictions. The advancement of generative models in the field of materials science requires processing and generating information across multiple modalities, including chemical formulas, crystal structures, spectroscopic data, and performance curves, while concurrently performing various tasks such as material generation, property prediction, and synthesis pathway recommendations. Using robust uncertainty quantification in generative models to guide experimental efforts enables researchers to prioritize materials based on confidence in their predicted properties, thereby enhancing the understanding of the reliability of model outputs.

Acknowledgements

The authors acknowledge the financial support received from several esteemed organizations. The authors extend their specific appreciation to the Deutsche Forschungsgemeinschaft for their contribution to the "Cluster of Excellence" POLiS (project number 390874152) and the Helmholtz Association for the MTET program (no 38.02.01).

Open Access funding enabled and organized by Projekt DEAL.

Conflict of Interest

The authors declare no conflict of interest.

Author Contributions

Deepalaxmi Rajagopal: conceptualization (lead); methodology (lead); visualization (lead); writing—original draft (lead); writing—review & editing (lead). **Adrian Cierpka:** conceptualization (supporting); methodology (supporting); writing—original draft (supporting); writing—review & editing (supporting). **Britta Nestler:** conceptualization (supporting); funding acquisition (lead); resources (lead); validation (supporting); visualization (supporting); writing—original draft (supporting); writing—review & editing (equal). **Arnd Koepp:** conceptualization (supporting); methodology (equal); supervision (lead); visualization (supporting); writing—original draft (supporting); writing—review & editing (equal).

Keywords: batteries • deep learning • generative modeling • materials design • machine learning

- [1] M. R. Palacin, *Acc. Mater. Res.* **2021**, 2, 319.
- [2] A. Walsh, *Nat. Chem.* **2015**, 7, 274.
- [3] T. Placke, R. Kloepsch, S. Dühnen, M. Winter, *J. Solid State Electrochem.* **2017**, 21, 1939.
- [4] D. Miranda, C. Costa, S. Lanceros-Mendez, *J. Electroanal. Chem.* **2015**, 739, 97.
- [5] E. W. C. Spotte-Smith, S. M. Blau, X. Xie, H. D. Patel, M. Wen, B. Wood, S. Dwaraknath, K. A. Persson, *Sci. Data* **2021**, 8, 203.
- [6] N. Yao, X. Chen, Z.-H. Fu, Q. Zhang, *Chem. Rev.* **2022**, 122, 10970.
- [7] X. Tan, M. Chen, J. Zhang, S. Li, H. Zhang, L. Yang, T. Sun, X. Qian, G. Feng, *Adv. Energy Mater.* **2024**, 14, 2400564.

- [8] G. Yoon, S. Moon, G. Ceder, K. Kang, *Chem. Mater.* **2018**, *30*, 6769.
- [9] A. Jain, S. P. Ong, G. Hautier, W. Chen, W. D. Richards, S. Dacek, S. Cholia, D. Gunter, D. Skinner, G. Ceder, K. A. Persson, *Apl Mater.* **2013**, *1*, 011002.
- [10] S. Curtarolo, W. Setyawan, G. L. Hart, M. Jahnatek, R. V. Chepulskii, R. H. Taylor, S. Wang, J. Xue, K. Yang, O. Levy, M. J. Mehl, H. T. Stokes, D. O. Demchenko, D. Morgan, *Comput. Mater. Sci.* **2012**, *58*, 218.
- [11] C. R. Groom, I. J. Bruno, M. P. Lightfoot, S. C. Ward, *Struct. Sci.* **2016**, *72*, 171.
- [12] D. Zagorac, H. Müller, S. Ruehl, J. Zagorac, S. Rehme, *Appl. Crystallogr.* **2019**, *52*, 918.
- [13] R. Y. Choi, A. S. Coyner, J. Kalpathy-Cramer, M. F. Chiang, J. P. Campbell, *Transl. Vision Sci. Technol.* **2020**, *9*, 14.
- [14] T. Lombardo, M. Duquesnoy, H. El-Bouysidi, F. Årén, A. Gallo-Bueno, P. B. Jørgensen, A. Bhowmik, A. Demortière, E. Ayerbe, F. Alcaide, M. Reynaud, J. Carrasco, A. Grimaud, C. Zhang, T. Vegge, P. Johansson, A. A. Franco, *Chem. Rev.* **2021**, *122*, 10899.
- [15] A. Y.-T. Wang, R. J. Muddock, S. K. Kauwe, A. O. Olynyk, A. Gurlo, J. Brgoch, K. A. Persson, T. D. Sparks, *Chem. Mater.* **2020**, *32*, 4954.
- [16] S. Manna, P. Paul, S. S. Manna, S. Das, B. Pathak, *Chem. Mater.* **2025**, *37*, 1759.
- [17] D. Foster, *Generative Deep Learning*, O'Reilly Media, Inc. **2022**.
- [18] A. J. Y. Wong, X. Zhou, Y. Lum, Z. Yao, Y. C. Chua, Y. Wen, Z. W. Seh, *Batteries Supercaps* **2022**, *5*, e202200309.
- [19] P. De Luna, J. Wei, Y. Bengio, A. Aspuru-Guzik, E. Sargent, *Nature* **2017**, *552*, 23.
- [20] K. T. Butler, D. W. Davies, H. Cartwright, O. Isayev, A. Walsh, *Nature* **2018**, *559*, 547.
- [21] A. Bhowmik, I. E. Castelli, J. M. Garcia-Lastra, P. B. Jørgensen, O. Winther, T. Vegge, *Energy Storage Mater.* **2019**, *21*, 446.
- [22] S. Li, F. You, *Small* **2024**, *20*, 2406153.
- [23] A. Mistry, A. A. Franco, S. J. Cooper, S. A. Roberts, V. Viswanathan, *ACS Energy Lett.* **2021**, *6*, 1422.
- [24] A. Lamb, arXiv preprint arXiv:2103.00265 **2021**.
- [25] J. M. Tomczak, *Deep Generative Modeling*, Springer, Berlin **2024**, 1–13.
- [26] I. Goodfellow, Y. Bengio, A. Courville, Y. Bengio, in *Deep Learning*, Vol. 1, MIT press **2016**.
- [27] Y. LeCun, Y. Bengio, G. Hinton, *nature* **2015**, *521*, 436.
- [28] D. Bank, N. Koenigstein, R. Giryes, in *Autoencoders*, Springer International Publishing, Cham **2023**, pp. 353–374.
- [29] P. Li, Y. Pei, J. Li, *Appl. Soft Comput.* **2023**, *138*, 110176.
- [30] Y. Zhao, P. Altschuh, J. Santoki, L. Griem, G. Tosato, M. Selzer, A. Koeppel, B. Nestler, *Acta Mater.* **2023**, *253*, 118922.
- [31] Y. Ji, A. Koeppel, P. Altschuh, D. Rajagopal, Y. Zhao, W. Chen, Y. Zhang, Y. Zheng, B. Nestler, *Comput. Mater. Sci.* **2024**, *232*, 112628.
- [32] S. Odaibo, arXiv preprint arXiv:1907.08956 **2019**.
- [33] D. P. Kingma, M. Welling, arXiv preprint arXiv:1312.6114 **2022**.
- [34] D. P. Kingma, M. Welling, *Foundations and Trends in Machine Learning* **2019**, Vol. 12, p. 307.
- [35] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, Y. Bengio, *Commun. ACM* **2020**, *63*, 139.
- [36] I. Goodfellow, arXiv preprint arXiv:1701.00160 **2016**.
- [37] B. Ghogh, A. Ghodsi, F. Karray, M. Crowley, arXiv preprint arXiv:2111.13282 **2021**.
- [38] K. Wang, C. Gou, Y. Duan, Y. Lin, X. Zheng, F.-Y. Wang, *IEEE/CAA J. Autom. Sin.* **2017**, *4*, 588.
- [39] L. Gonog, Y. Zhou, *14th IEEE Conf. on Industrial Electronics and Applications (ICIEA)*, Vol. 1 **2019**, p. 505.
- [40] J. Sohl-Dickstein, E. Weiss, N. Maheswaranathan, S. Ganguli, *Int. Conf. Mach. Learn.* **2015**, *37*, 2256.
- [41] A. Q. Nichol, P. Dhariwal, *Int. Conf. Mach. Learn.* **2021**, *139*, 8162.
- [42] J. Ho, A. Jain, P. Abbeel, *Adv. Neural Inf. Process. Syst.* **2020**, *33*, 6840.
- [43] F.-A. Croitoru, V. Hondru, R. T. Ionescu, M. Shah, *IEEE Trans. Pattern Anal. Mach. Intell.* **2023**, *45*, 10850.
- [44] L. Yang, Z. Zhang, Y. Song, S. Hong, R. Xu, Y. Zhao, W. Zhang, B. Cui, M.-H. Yang, *Acm Comput. Surv.* **2023**, *56*, 1.
- [45] R. Rombach, A. Blattmann, D. Lorenz, P. Esser, B. Ommer, *Proc. of the IEEE/CVF Conf. on Computer Vision and Pattern Recognition* **2022**, Vol. 1, p. 10684.
- [46] Y. Bengio, R. Ducharme, P. Vincent, C. Jauvin, *J. Mach. Learn. Res.* **2003**, *3*, 1137.
- [47] M. Dalal, A. C. Li, R. Taori, arXiv preprint arXiv:1910.07737 **2019**.
- [48] J. Xiong, G. Liu, L. Huang, C. Wu, T. Wu, Y. Mu, Y. Yao, H. Shen, Z. Wan, J. Huang, C. Tao, S. Yan, H. Yao, L. Kong, H. Yang, M. Zhang, G. Sapiro, J. Luo, P. Luo, N. Wong, arXiv preprint arXiv:2411.05902 **2024**.
- [49] S. Bond-Taylor, A. Leach, Y. Long, C. G. Willcocks, *IEEE Trans. Pattern Anal. Mach. Intell.* **2021**, *44*, 7327.
- [50] H. Larochelle, I. Murray, *Proc. of the Fourteenth International Conf. on Artificial Intelligence and Statistics* **2011**, Vol. 1, p. 29.
- [51] M. Germain, K. Gregor, I. Murray, H. Larochelle, *Int. Conf. Mach. Learn.* **2015**, *37*, 881.
- [52] A. van den Oord, N. Kalchbrenner, K. Kavukcuoglu, arXiv preprint arXiv:1601.06759 **2016**.
- [53] A. van den Oord, N. Kalchbrenner, O. Vinyals, L. Espeholt, A. Graves, K. Kavukcuoglu, arXiv preprint arXiv:1606.05328 **2016**.
- [54] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, I. Polosukhin, *Adv. Neural Inf. Process. Syst.* **2017**, *30*, 5998.
- [55] A. Radford, K. Narasimhan, T. Salimans, I. Sutskever, OpenAI Technical Report **2018**.
- [56] J. Gao, P. Li, Z. Chen, J. Zhang, *Neural Comput.* **2020**, *32*, 829.
- [57] R. F. Ziesche, N. Kardjilov, W. Kockelmann, D. J. Brett, P. R. Shearing, *Joule* **2022**, *6*, 35.
- [58] R. Carter, B. Huhman, C. T. Love, I. V. Zhenyuk, *J. Power Sources* **2018**, *381*, 46.
- [59] O. Furat, L. Petrich, D. P. Finegan, D. Diercks, F. Usseglio-Viretta, K. Smith, V. Schmidt, *npj Comput. Mater.* **2021**, *7*, 105.
- [60] H. Xu, F. Usseglio-Viretta, S. Kench, S. J. Cooper, D. P. Finegan, *J. Power Sources* **2020**, *480*, 229101.
- [61] Y. Ning, F. Yang, Y. Zhang, Z. Qiang, G. Yin, J. Wang, S. Lou, *Matter* **2024**, *7*, 2011.
- [62] C. Ling, *npj Comput. Mater.* **2022**, *8*, 33.
- [63] X. Chen, X. Liu, X. Shen, Q. Zhang, *Angew. Chem.* **2021**, *133*, 24558.
- [64] D. P. Finegan, I. Squires, A. Dahari, S. Kench, K. L. Jungjohann, S. J. Cooper, *ACS Energy Lett.* **2022**, *7*, 4368.
- [65] S. Curtarolo, G. L. Hart, M. B. Nardelli, N. Mingo, S. Sanvito, O. Levy, *Nat. Mater.* **2013**, *12*, 191.
- [66] S. Kirklin, J. E. Saal, B. Meredig, A. Thompson, J. W. Doak, M. Aykol, S. Rühl, C. Wolverton, *npj Comput. Mater.* **2015**, *1*, 1.
- [67] L. Talirz, S. Kumbhar, E. Passaro, A. V. Yakutovich, V. Granata, F. Gargiulo, M. Borelli, M. Uhrin, S. P. Huber, S. Zoupanos, C. S. Adorf, C. W. Andersen, O. Schütt, C. A. Pignedoli, D. Passerone, J. VandeVondele, T. C. Schulthess, B. Smit, G. Pizzi, N. Marzari, *Sci. Data* **2020**, *7*, 299.
- [68] T. Xie, J. C. Grossman, *Phys. Rev. Lett.* **2018**, *120*, 145301.
- [69] O. T. Unke, S. Chmiela, H. E. Sauceda, M. Gastegger, I. Poltavsky, K. T. Schütt, A. Tkatchenko, K.-R. Müller, *Chem. Rev.* **2021**, *121*, 10142.
- [70] J. Noh, G. H. Gu, S. Kim, Y. Jung, *Chem. Sci.* **2020**, *11*, 4871.
- [71] B. Sanchez-Lengeling, A. Aspuru-Guzik, *Science* **2018**, *361*, 360.
- [72] J. Noh, J. Kim, H. S. Stein, B. Sanchez-Lengeling, J. M. Gregoire, A. Aspuru-Guzik, Y. Jung, *Matter* **2019**, *1*, 1370.
- [73] A. Nouri, N. Sokolovska, J.-C. Crivello, arXiv preprint arXiv:1810.11203, **2018**.
- [74] Y. Zhao, M. Al-Fahdi, M. Hu, E. M. Siriwardane, Y. Song, A. Nasiri, J. Hu, *Adv. Sci.* **2021**, *8*, 2100566.
- [75] T. Xie, X. Fu, O.-E. Ganea, R. Barzilay, T. Jaakkola, arXiv preprint arXiv:2110.06197 **2021**.
- [76] S. Yang, K. Cho, A. Merchant, P. Abbeel, D. Schuurmans, I. Mordatch, E. D. Cubuk, arXiv preprint arXiv:2311.09235 **2023**.
- [77] C. Zeni, R. Pinsler, D. Zügner, A. Fowler, M. Horton, X. Fu, Z. Wang, A. Shysheya, J. Crabbé, S. Ueda, R. Sordillo, L. Sun, J. Smith, B. Nguyen, H. Schulz, S. Lewis, C.-W. Huang, Z. Lu, Y. Zhou, H. Yang, H. Hao, J. Li, C. Yang, W. Li, R. Tomioka, T. Xie, *Nature* **2025**, *639*, 624.
- [78] Y. S. Meng, V. Srinivasan, K. Xu, *Science* **2022**, *378*, eabq3750.
- [79] A. S. Alshehri, F. You, *Chem. Eng. J.* **2022**, *444*, 136669.
- [80] L. David, A. Thakkar, R. Mercado, O. Engkvist, *J. Cheminform.* **2020**, *12*, 56.
- [81] A. Khajeh, X. Lei, W. Ye, Z. Yang, L. Hung, D. Schweigert, H.-K. Kwon, *Digit. Discovery* **2025**, *4*, 11.
- [82] Z. Yang, W. Ye, X. Lei, D. Schweigert, H.-K. Kwon, A. Khajeh, *npj Comput. Mater.* **2024**, *10*, 296.
- [83] D. Yoon, J. Lee, S. Lee, *Appl. Sci.* **2025**, *15*, 3640.
- [84] R. Gómez-Bombarelli, J. N. Wei, D. Duvenaud, J. M. Hernández-Lobato, B. Sánchez-Lengeling, D. Sheberla, J. Aguilera-Iparraguirre, T. D. Hirzel, R. P. Adams, A. Aspuru-Guzik, *ACS Cent. Sci.* **2018**, *4*, 268.

- [85] D. Rajagopal, A. Koeppe, M. Esmailpour, M. Selzer, W. Wenzel, H. Stein, B. Nestler, *Adv. Energy Mater.* **2023**, *13*, 2301985.
- [86] R. F. Ziesche, T. M. Heenan, P. Kumari, J. Williams, W. Li, M. E. Curd, T. L. Burnett, I. Robinson, D. J. Brett, J. M., et al Ehrhardt, *Adv. Energy Mater.* **2023**, *13*, 2300103.
- [87] A. Gayon-Lombardo, L. Mosser, N. P. Brandon, S. J. Cooper, *npj Comput. Mater.* **2020**, *6*, 82.
- [88] S. Kench, S. J. Cooper, *Nat. Mach. Intell.* **2021**, *3*, 299.
- [89] A. Dahari, S. Kench, I. Squires, S. J. Cooper, *Adv. Energy Mater.* **2023**, *13*, 2202407.

Manuscript received: June 30, 2025
Revised manuscript received: August 15, 2025
Version of record online: