# Physics-Constrained Predictive Molecular Latent Space Discovery with Graph Scattering Variational Autoencoder

Navid Shervani-Tabar[1, a)] and Nicholas Zabaras[1, b)]

*Scientific Computing and Artificial Intelligence (SCAI) Laboratory, University of Notre Dame, Notre Dame, IN 46556, USA*

(Dated: 30 September 2020)

Recent advances in artificial intelligence have propelled the development of innovative computational materials modeling and design techniques. In particular, generative deep learning models have been used for molecular representation, discovery and design with applications ranging from drug discovery to solar cell development. In this work, we assess the predictive capabilities of a molecular generative model developed based on variational inference and graph theory. The encoder network is based on the scattering transform, which allows for a better generalization of the model in the presence of limited training data. The scattering layers incorporate adaptive spectral filters which are tailored to the training dataset based on the molecular graphs' spectra. The decoding network is a one-shot graph generative model that conditions atom types on molecular topology. We present a quantitative assessment of the latent space in terms of its predictive ability for organic molecules in the QM9 dataset. To account for the limited size training data set, a Bayesian formalism is considered that allows us capturing the uncertainties in the predicted properties.

a)Electronic mail: nshervan@nd.edu

b)Electronic mail: nzabaras@gmail.com; https://www.zabaras.com/

## I. INTRODUCTION

Optimizing molecules for desired properties is a challenging task. Molecular property optimization has applications in a broad range of fields ranging from drug discovery[1] to organic solar cells[2]. Computer simulations of molecular systems remain the most prominent method for guiding molecular design[3]. In drug design, molecular dynamics (MD) methods are used to predict how strongly would a given molecule bind to a biological target, such as a protein or a nucleic acid. These computationally expensive methods usually rely on density functional theory (DFT).

With the recent advances in Artificial Intelligence (AI), deep learning techniques have become the lead contender for future direction in molecular design optimization[4]. AI approaches give accurate predictions of molecular properties faster than computer simulations[5]. Moreover, deep learning methods have ignited automatic exploration of chemical space for molecular design by providing an underlying low-dimensional latent representation of the molecular space through generative models[6]. These have significantly reduced the required resources for synthesizing new drugs and molecules.

The first attempts of such methods[6] were based on Simplified Molecular-Input Line-Entry System (SMILES)[7] and used variational autoencoders to learn a latent representation of molecules. One major problem with such methods was the validity of the generated molecules. To overcome this problem, Kusner et. al.[8] proposed grammar Variational AutoEncoder (VAE), which took advantage of parse trees to represent SMILES string structures and subsequently, learned these parse trees instead of SMILES. While this helped with syntactical validity, that is, the set of rules that define a correct combination of symbols for a particular representation, they do not guarantee semantic validity, which are the set of physical constraints that define a chemically viable molecule. With the recent advances in geometric deep learning[9], however, researchers have shifted their focus to models utilizing graph based molecular representations[10]. This shift was influenced by the richness of graph representation in topological information of the molecules and flexibility of this representation in adding desired chemical parameters to each node. Moreover, molecules can be uniquely represented by a graph.

In a molecular graph, each atom is represented by a vertex and each covalent bond is defined by an edge that connects the corresponding atoms. Furthermore, weighted graphs can account for the proximity and strength of bonds (valence of bonds).

Various deep learning frameworks have been used for generation of molecular graphs. These include Generative Adversarial Networks (GAN)[11] and VAEs[12]. More recently, a flow-based generative model[13] was used[14] for generating molecules. Depending on the molecular representation used, there have been many encoding networks proposed in recent years. These include Cartesian,[15–17] SMILES,[6,18,19] and non-Euclidean (geometric deep learning)[20–22] representations.

Wavelet scattering transform[23] is a deep convolutional neural network that uses a cascade of wavelet filters followed by a nonlinearity operator to generate invariant features from an input signal. These features are constructed by linearizing the variations within predefined local symmetry groups[24]. A map of these local invariants to linear space can handle learning tasks such as regression and classification. In these networks, the number of nodes, layers, filters, and non-linearity layers are predefined. This eliminates the need for training the network and hence, limits the uncertainty introduced by the encoder. Moreover, scattering improves the generalization of a network in the presence of limited data[25]. In recent years, the scattering transform has been extended to non-Euclidean domains including graphs[26–28]. A graph scattering network takes the weight matrix of the graph and signals residing on its vertices as input and transforms it to features that are invariant to permutation and stable to graph and signal manipulation[27]. This transform typically uses spectral-design wavelets to perform convolutions; that is, wavelets designed in the eigenspace defined by the matrix representation of the graph. Gama et al.[28] used diffusion wavelets[29], which are defined using diffusion processes on graphs as filters. On the other hand, Zou and Lerman[27] used Hammond et al.'s[30] spectral wavelets for feature extraction. These wavelets, however, are only adapted to the length of the graph spectrum. Therefore, in the cases where the graph eigenvalues are irregularly spaced, this may result in highly-correlated wavelets.

The graph VAEs, which generate a latent space from input graphs, can roughly be divided into two categories based on their decoding network. The first group are the autoregressive models[31,32], which generate a molecule by sequentially adding components. The second group includes one-shot models[12,33], where the model simultaneously outputs weighted adjacency matrix and signals residing on its nodes. In the former, each iteration is conditioned on the previous iterations. These models become harder to train as the length of the sequence increases. The latter is not suitable for generating graphs larger than order of 10 nodes, however, it is faster and computationally more efficient. This type of models, however, need the maximum graph size to be predetermined[12]. Many recent works on one-shot graph VAEs assume independence of nodes and edges in their graph model. Despite their success in many applications, these implementations may not learn validity constraints that dictate certain combination of nodes and edges. As a result, they do not guarantee the semantic validity of the generated molecular graph, i.e. a single connected graph which complies with the valency of the nodes. Simonovsky and Komodakis[12] investigated a remedy by defining node presence probability as a function of the existence probabilities of the incident edges. While this helps with such problems as isolated nodes, it does not consider valency of the atoms. Furthermore, methods have been proposed to impose chemical constraints on the decoding network[34]. In this work, Ma et al. have reformulated a constrained optimization as an unconstrained regularization problem by imposing validity constraints using penalty terms on the VAE's standard objective function, which penalize the network for semantically invalid generated outputs. Although this method can significantly increase the validity of the molecule, at the same time, it considerably reduces the uniqueness of the generated molecules.

Bayesian approaches have been used to model predictive uncertainty in neural networks[35]. It is common to approximate the full posterior over model parameters using the Laplace approximation[36]. In practice, however, approximating the distribution of model parameters of a graph generative model with a Gaussian distribution may not be feasible as the sampled model could suffer from low validity of the outputs. In order to develop a predictive method for cases in which an accurate model over parameters is not available, Harris[37] proposed a bootstrap predictive distribution as an approximation to the predictive distribution. Fushiki et al.[38] further extended this to non-parametric bootstrap distribution. Fushiki recently proposed Bayesian bootstrap prediction[39], which takes advantage of Rubin's[40] Bayesian bootstrap by imposing a non-informative prior over bootstrap sampling weights.

In this work, we are interested in computing the statistics of molecular properties given a small size training data set. We use a deep generative model based on variational inference and perform accurate Monte Carlo estimation of molecular properties. We consider Bayesian bootstrap resampling to yield a predictive distribution over estimated properties of the generated molecules. We show our probabilistic confidence on the estimated properties. As physical constraints are hard to satisfy when sampling nodes and edges independently, in this work, we formulate the probabilistic output as the joint distribution of nodes and edges, where probabilities of the nodes are defined as conditional distributions given the edges. This provides a direct map from the edge type scores to the node probabilities. This map implicitly encourages learning the physical constraints on consistent node and edge types without imposing explicit physical constraints. We have developed a hybrid neural network which is constructed of a graph scattering network coupled with a fully-connected network (FCN). Furthermore, in order to remedy the issues arising from the non-uniform spectra of the molecular graphs, in this work we take advantage of adaptive spectral filters[41] to construct our inference network. These filters are designed to adapt to the given training dataset, which increases the discriminatory power of the encoding network.

The rest of this work is structured as follows. In Section II we describe the inference problem and the general setup. Section III discusses the encoding and decoding network architectures used. Lastly, in Section IV, we train the network to discover a latent representation for the QM9 molecular database and to generate new molecules from samples in the latent space. We further assess the uncertainty of the network providing probabilistic evaluation of a number of properties.

## II. PROBLEM DEFINITION

Graphs[42,43], along with SMILES[7], are two of the most common ways to represent a molecule in computational chemistry. Graphs $G = (V,E)$ consist of a set of vertices $V = \{v_1,\dots,v_N\}$ with $N = |V|$ and a set of edges $E \subseteq V \times V$, defined by distinctive pairs of vertices $v_m v_n \in V \times V$ with $1 \leq m,n, \leq N$ and $m \neq n$. A weighted graph $G = (E,V,W)$, is made of weights assigned to each edge $W = \{w_{mn} | v_m v_n \in E\}$. The assigned value $w_{mn}$ can represent proximity or strength of the relationship between the pair of vertices $v_m v_n$. In a molecular graph, atoms are represented by graph vertices while edges represent the atomic bonds. In this setting, the type of the bond is shown by the weights assigned to the edges. Graph is a domain on which we can represent a signal $f : V \to \mathbb{R}^N$. In a molecular graph, signals are the type of the atoms sitting on the corresponding vertices. In some applications, these signals are extended to include extra atomic features, including hybridization, hydrogen neighbors, etc.

Variational Graph AutoEncoders[44] are designed to analyze molecular data defined on a graph domain. In the present work, the encoding network has a hybrid architecture which takes advantage of the graph scattering transform[27,28] in combination with a fully-connected neural network to extract features from the given input graphs. The graph scattering transform is a type of deep neural network with predefined parameters that takes the graph as input and performs convolutions using spectral graph filters, combined with modulus non-linearity to generate features[23,26]. These features are invariant to permutation and stable with respect to graph and signal manipulation[27].

For a molecular graph $\mathcal{G}$, we define the structure of the graph using a weight matrix $\boldsymbol{W}$ and the data on the graph using a signal vector $\boldsymbol{f}$. Each element of the signal vector $f_i$ represents the atom label for the corresponding vertex of the graph. A categorical probability $\tilde{f}_i = p(f_i)$ for the type of the atom on node $i$ of the graph $\mathcal{G}$ is a vector of probability values for each atom type class, such that

$$f_i = \arg\max(\tilde{f}_i). \tag{1}$$

Similarly, each element of the weight matrix $W_{i,j}$ is a sample of a discrete variable that gives the label for the possible weight values. These values correspond to the type of the covalent bond. If $\tilde{W}_{i,j} = p(W_{i,j})$ is the categorical probability distribution for $W_{i,j}$, then it gives a vector of probability values of each bond type, such that

$$W_{i,j} = \arg\max(\tilde{W}_{i,j}). \tag{2}$$

With these two components in hand, we can write the probability distribution $p(\mathcal{G})$ of a molecular graph as a joint distribution of atom and bond types for all the nodes and edges in the molecular graph as follows:

$$p(\mathcal{G}) = p(\boldsymbol{f}, \boldsymbol{W}) \tag{3}$$

Given a finite-size dataset of $K$ molecular graphs $\mathscr{G} = \{\mathcal{G}^{(i)}\}_{i=1}^{K}$, the objective is to reveal an underlying latent representation for this dataset. Denoting the latent space variable with $\boldsymbol{z}$ and given a prior distribution of $p(\boldsymbol{z})$, the joint distribution over the observed data $\mathcal{G}$ and latent variable $\boldsymbol{z}$ can be written as

$$p(\mathcal{G}, \boldsymbol{z}) = p(\mathcal{G}|\boldsymbol{z})p(\boldsymbol{z}). \tag{4}$$

In Eq. 4, $p(\mathcal{G}|\boldsymbol{z})$ represents the probability of the molecular graph $\mathcal{G}$ conditioned on its $J$-dimensional latent representation $\boldsymbol{z}$. A generative model $p(\mathcal{G})$ for the molecular graph can be achieved by marginalizing the joint distribution $p(\mathcal{G}, \boldsymbol{z})$ over the latent representation

$$p(\mathcal{G}) = \int p(\mathcal{G}, \boldsymbol{z})d\boldsymbol{z} = \int p(\mathcal{G}|\boldsymbol{z})p(\boldsymbol{z})d\boldsymbol{z}. \tag{5}$$

To address the intractable calculation of the marginal likelihood $p_{\boldsymbol{\theta}}(\mathscr{G})$, where the model is parameterized by $\boldsymbol{\theta}$, we introduce standard variational inference in the context of a VAE framework and maximize the following lower-bound on the marginal log-likelihood:

$$\begin{aligned}\mathscr{L}(\boldsymbol{\theta}, \boldsymbol{\phi}; \mathscr{G}) =& \sum_{i=1}^{K} \mathscr{L}\left(\boldsymbol{\theta}, \boldsymbol{\phi}; \mathcal{G}^{(i)}\right) \\ =& \sum_{i=1}^{K} \mathbb{E}_{q_{\boldsymbol{\phi}}(\boldsymbol{z}^{(i)}|\mathcal{G}^{(i)})}\left[\log p_{\boldsymbol{\theta}}(\mathcal{G}^{(i)}|\boldsymbol{z}^{(i)})\right] \\ &- \sum_{i=1}^{K} \mathrm{D}_{\mathrm{KL}}\left[q_{\boldsymbol{\phi}}(\boldsymbol{z}^{(i)}|\mathcal{G}^{(i)}) \| p_{\boldsymbol{\theta}}(\boldsymbol{z}^{(i)})\right].\end{aligned} \tag{6}$$

Here, $p_{\boldsymbol{\theta}}(\mathcal{G}|\boldsymbol{z})$ is a decoding distribution parametrized by $\boldsymbol{\theta}$, $q_{\boldsymbol{\phi}}(\boldsymbol{z}|\mathcal{G})$ represents the variational distribution of the encoding network that approximates the posterior of $\boldsymbol{z}$ and $\mathrm{D}_{\mathrm{KL}}$ refers to the Kullback-Leibler distance between two distributions. The first term in the lower-bound represents the negative expected reconstruction error, while the second term introduces regulation by forcing the posterior of the latent variables to be close to the prior $p_{\boldsymbol{\theta}}(\boldsymbol{z})$, here taken as standard Gaussian. For a Gaussian $q_{\boldsymbol{\phi}}$, it can be shown[45]

$$-\mathrm{D}_{\mathrm{KL}}\left[q_{\boldsymbol{\phi}}(\boldsymbol{z}|\mathscr{G})\|p_{\boldsymbol{\theta}}(\boldsymbol{z})\right]=\frac{1}{2}\sum_{j=1}^{J}\left(1+\log\left((\sigma_j)^2\right)-(\mu_j)^2-(\sigma_j)^2\right),$$

where $q_{\boldsymbol{\phi}}(\boldsymbol{z}|\mathscr{G})=\mathcal{N}\left(\boldsymbol{z}|\boldsymbol{\mu},\mathrm{diag}\left(\boldsymbol{\sigma}^2\right)\right)$ and the prior model $p_{\boldsymbol{\theta}}(\boldsymbol{z})$ is a standard Gaussian distribution. The reconstruction loss term in $\mathscr{L}$ is discussed in Section III B.

Maximizing the lower-bound will lead to the desired maximum likelihood estimate for the parameters $\boldsymbol{\phi}$ and $\boldsymbol{\theta}$ of the variational posterior $q_{\boldsymbol{\phi}}$ and decoding distribution $p_{\boldsymbol{\theta}}$, respectively. In a Variational AutoEncoder (VAE) setting, $q_{\boldsymbol{\phi}}$ provides the embedding of molecular graphs, whereas $p_{\boldsymbol{\theta}}$ allows the generation of molecular graphs using different samples of $\boldsymbol{z}$ in the latent space.

## III. MODEL

With the rise of signal processing methods on graphs[9,46,47], various graph-based networks have been introduced for encoding in VAEs. Similarly, there are various decoding networks used for graph generation. We discuss next the encoding and decoding networks developed in our VAE framework.

### A. Encoding

In this work, we use a hybrid scattering network for the embedding of molecular graphs. The encoding network is initialized with layers of scattering transform[23] which is a generic feature extraction network that uses predefined (non-trainable) parameters to construct feature maps from the input. These parameters include multiresolution filters and modulus non-linearity. These layers are followed by an FCN. In this sense, the encoding network has a hybrid structure, where the graph scattering transform acts as an interface between graph input and the conventional FCN layers that output a latent representation of the molecular graph.

Graph filters can roughly be divided into two categories: (i) Vertex-design filters, which are defined on the vertices of the graph as the linear combination of k-hop neighbors of each vertex and (ii) spectral-design filters, which are defined in the spectral domain of the graph. The latter is used in this work to define the kernels. The spectral domain of a graph is defined based on the graph Laplacian $\mathcal{L}$, which is a symmetric, positive semidefinite matrix defined as

$$\mathcal{L} := \boldsymbol{\Delta} - \boldsymbol{W}, \tag{7}$$

where $\boldsymbol{\Delta}$ is the vertex-degree matrix with diagonal elements $\Delta_{ii}$ defined by the sum of the weights of the edges incident to node $v_i$ and off-diagonal elements equal to zero. The eigen-decomposition of the Laplacian matrix yields real, non-negative eigenvalue $0 = \lambda_0 \leq \lambda_1 \leq \ldots \leq \lambda_{N-1}$ and the corresponding eigenvectors $\chi_\ell$, which compose the graph spectral domain.

An important application of graph spectral theory is to provide tools for adapting the Fourier transform to graphs. In the graph domain, the Fourier transform is defined using the correspondence between the eigen-decomposition of the Laplace operator in the Euclidean domain and of the Laplacian matrix in the graph domain[46]:

$$\hat{f}(\ell) := \sum_{i=1}^{N} \chi_\ell^*(i) f(i), \tag{8}$$

where $i$ is the index in the vertex domain and $\ell$ is the index in the spectral domain. The corresponding inverse transform is defined as

$$f(i) := \sum_{\ell=0}^{N-1} \hat{f}(\ell)\chi_\ell(i). \tag{9}$$

The convolution $\boldsymbol{f} * \boldsymbol{g}$ of a signal $\boldsymbol{f}$ with a filter $\boldsymbol{g}$ in the vertex domain is equivalent to the multiplication of their Fourier transforms $\hat{\boldsymbol{f}}$ and $\hat{\boldsymbol{g}}$ in the frequency domain. This can be used to reformulate the convolution in the spectral domain as follows:

$$\begin{aligned} \boldsymbol{f} * \boldsymbol{g} &= \boldsymbol{\chi}\hat{\boldsymbol{g}}(\boldsymbol{\Lambda})\boldsymbol{\chi}^*\boldsymbol{f} \\ &= \hat{\boldsymbol{g}}(\mathcal{L})\boldsymbol{f}. \end{aligned} \tag{10}$$

Here, the Laplacian is decomposed as $\mathcal{L} = \boldsymbol{\chi}\boldsymbol{\Lambda}\boldsymbol{\chi}^*$, $\boldsymbol{\Lambda}$ is the diagonal matrix of the eigenvalues $\lambda_\ell$ of $\mathcal{L}$, and the diagonal filtering matrix $\hat{\boldsymbol{g}}(\boldsymbol{\Lambda})$ is defined as

$$\hat{\boldsymbol{g}}(\boldsymbol{\Lambda}) = \begin{bmatrix} \hat{g}(\lambda_0) & & 0 \\ & \ddots & \\ 0 & & \hat{g}(\lambda_{N-1}) \end{bmatrix}. \tag{11}$$

The filter-bank used in this work consists of a list of filters localized around different frequencies. We adapt the method introduced in 41 to construct band-pass filters by translating a main window in the spectral domain. We select half-cosine kernel to define the main window as

$$\hat{g}^{'}(\lambda) := \sum_{k=0}^{\mathcal{K}} d_k \left[ \cos\left(2\pi k \left(\frac{\lambda}{a} - \frac{1}{2}\right)\right) \cdot \mathbb{1}_{\{0 \leq \lambda < a\}} \right], \tag{12}$$

where

$$a = \frac{R\gamma}{\mathcal{J} - R + 1}, \tag{13}$$

is dilation factor, $R$ is the kernel overlap which tunes the kernel's width in the spectral domain, $\mathcal{J}$ is the number of filters in the filter-bank, and $\gamma$ is the largest eigenvalue in the spectrum $\lambda_{\max}$. With the main window selected, we define the system of filters $\{\hat{g}^{'}_j\}_{j=1}^{\mathcal{J}}$ as

$$\hat{g}^{'}_j(\lambda) := \hat{g}^{'}\left(\lambda - \frac{a(j - R + 1)}{R}\right), \tag{14}$$

which are uniform translates of one another in the spectral domain. Note that the use of $\gamma$ in Eq. 13 adapts these filters to the length of the spectrum.

As the eigenvalues of molecular graphs are unevenly spaced throughout the spectrum of the dataset, Eq. 14 may result in kernels that are highly correlated with the ones at the neighboring nodes and scales. To overcome this issue, we adapt the kernels to the spectrum of the molecular graphs by the means of a warping function. The warping function $\omega$ is defined as an approximation of the cumulative spectral density function of the union of the eigenvalues of the molecular graphs from dataset $\mathcal{G}$. In 41, this method is used for designing filters on a single graph or a family of random graphs with known empirical spectral distribution. In this work, given the training data, we use kernel density estimation (KDE) to approximate the empirical spectral distribution of the Laplacian eigenvalues of the molecular graphs. Using the warping function $\omega$, the adaptive filter-bank is defined as

$$\hat{g}_j(\lambda) := \hat{g}^{'}_j(\omega(\lambda)). \tag{15}$$

Consequently, we use $\gamma = \omega(\lambda_{max})$ in Eq. 13. Note that while the filter-bank is not adapted to each training data, overall, the density of the kernels is higher in the dense regions of the spectral domain. This effectively boosts the discriminatory capabilities of the encoding network. Figure 1 shows the histogram of the eigenvalues, the empirical spectral cumulative distribution function, and the resulting filter-bank kernels.

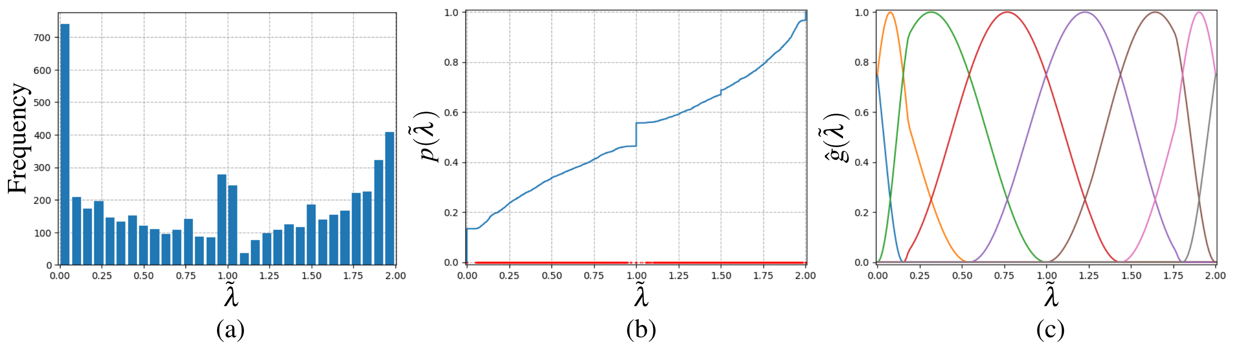


FIG. 1. Spectral graph filters: (a) histogram of the eigenvalues $\tilde{\lambda}$ of the normalized graph Laplacians for a training set of $K = 600$ graphs and (b) empirical spectral cumulative distribution function of eigenvalues $\tilde{\lambda}$ in Fig. 'a'. We construct the warping function $\omega$ as a smooth approximation of the CDF function. (c) Adaptive kernels $\{\hat{g}_j\}_{j=1}^{\mathcal{J}}$ which are tailored to the union of normalized Laplacian spectra shown in Fig. 'a', with $\mathcal{J} = 8$. We set the parameters $\mathcal{K} = 1$, $d_0 = d_1 = 0.5$, and $R = 3$ to define the main window (Eq. 12).

In the vertex domain, kernels are localized on each vertex $v_i$ by filtering a Kronecker delta function $\delta_i$ placed on the specified vertex $v_i$, leading to the dictionary of kernels defined as

$$g_{i,j} = \sqrt{N}\delta_i * g_j = \sqrt{N}\sum_{\ell=0}^{N-1} \hat{g}_j(\lambda_\ell)\chi_\ell^*(i)\chi_\ell. \tag{16}$$

Using this, we find analysis coefficients for a convolution with kernel $\boldsymbol{g}_j$ of scale $j$ as

$$\{\langle \boldsymbol{f}, g_{i,j}\rangle\}_{i=1}^{N} = \hat{\boldsymbol{g}}_j(\mathcal{L})\boldsymbol{f}, \tag{17}$$

where $\hat{\boldsymbol{g}}_j(\mathcal{L})$ is a matrix function which defines a frame for the associated spectral filter $\hat{\boldsymbol{g}}_j$ as shown in Eqs. 10 and 11. For a filter-bank $\{\hat{\boldsymbol{g}}_j(\cdot)\}_{j=1}^{\mathcal{J}}$, the frame consists of a collection of frames for each kernel $\hat{\boldsymbol{g}}_j$

$$\hat{\boldsymbol{g}}(\mathcal{L}) = \begin{bmatrix} \hat{\boldsymbol{g}}_1(\mathcal{L}) \\ \vdots \\ \hat{\boldsymbol{g}}_{\mathcal{J}}(\mathcal{L}) \end{bmatrix}. \tag{18}$$

Analysis with a filter-bank of $\mathcal{J}$ kernels extracts $\mathcal{J}$ features from the signal. Given Eq. 18, we can extend Eq. 17 to all kernels in a filter-bank

$$\{\langle \boldsymbol{f}, g_{i,j}\rangle\}_{i=1,\dots,N,\; j=1,\dots,\mathcal{J}} = \hat{\boldsymbol{g}}(\mathcal{L})\boldsymbol{f}, \tag{19}$$

where $\mathcal{L}$ is the graph Laplacian. Alternatively, one can use the normalized Laplacian

$$\tilde{\mathcal{L}} = \boldsymbol{\Delta}^{-1/2}\mathcal{L}\boldsymbol{\Delta}^{-1/2}, \tag{20}$$

to define the Fourier basis[41]. By using the latter, strength of filtering is not affected by the degree of the node and hence in this work we take advantage of the normalized Laplacian matrix as the tool for performing convolution on input signal.

After defining the spectral filters, we can describe the scattering layers. Mallat[23] introduced scattering networks as networks that use wavelet transform building blocks to generate invariants with respect to different groups of symmetry. In each layer, the scaling function creates feature maps of the input data and the wavelet filters extract higher frequency information from the input and propagate it forward to the next layer after application with a non-linear function. In the graph domain[28,48,49], we are interested in graph-level feature maps. Hence, instead of scaling function, we adapt the average pooling operator from 49 to construct feature maps at each layer

$$\eta = \frac{\mathbf{1}^T}{N}. \tag{21}$$

Thus, the zeroth-order scattering coefficient $\mathcal{S}_0 \boldsymbol{f}$ is achieved by averaging the signals across all the nodes of the graph.

In order to create more feature maps, higher frequency information are retrieved from the signal using the spectrum-adapted band-pass filters $\{\hat{\boldsymbol{g}}_j(\cdot)\}_{j=1}^{\mathcal{J}}$. Zou and Lerman[48] perform this using spectral wavelets[30] while Gama et al.[49] use tight frame wavelets and scaling function[41]. Unlike 49, we don't include scaling function while retrieving the higher frequency information and do so with the band-pass filters (Eqs. 12-15) that are localize in the vertex and spectral domain. Moreover, in our work filters are tailor made for the particular dataset so as to minimize the correlation among neighboring filters. This high frequency data is propagated to the next layer after application of the non-linearity operator $\rho$ to generate more feature maps. The non-linearity prevents the network from generating trivial feature maps by averaging oscillatory outputs of the convolution.

In summary, layer $m$ of the scattering transform can be written as

$$\mathcal{W}_m \boldsymbol{f} = \{\mathcal{S}_{m-1}\boldsymbol{f},\, \mathcal{U}_m \boldsymbol{f}\} = \left\{\eta(\mathcal{U}_{m-1}\boldsymbol{f}),\, \rho\left(\hat{\boldsymbol{g}}_j(\tilde{\mathcal{L}})\mathcal{U}_{m-1}\boldsymbol{f}\right)\right\}_{j\in\Gamma}, \tag{22}$$

where $\mathcal{S}_{m-1}\boldsymbol{f}$ denotes the $(m-1)$th-order scattering coefficients, $\mathcal{U}_m\boldsymbol{f}$ shows the remaining high frequencies that are propagated to the next layer of the network after applying non-linearity $\rho$, and $\Gamma$ is the set of indices for filters in the filter-bank.

Fig. 2 illustrates the feature space generated by the scattering transform, which is the hidden layer input to the FCN layers. To visualize this space, we take advantage of principal component analysis (PCA) and project the high-dimensional feature space into a two-dimensional space.

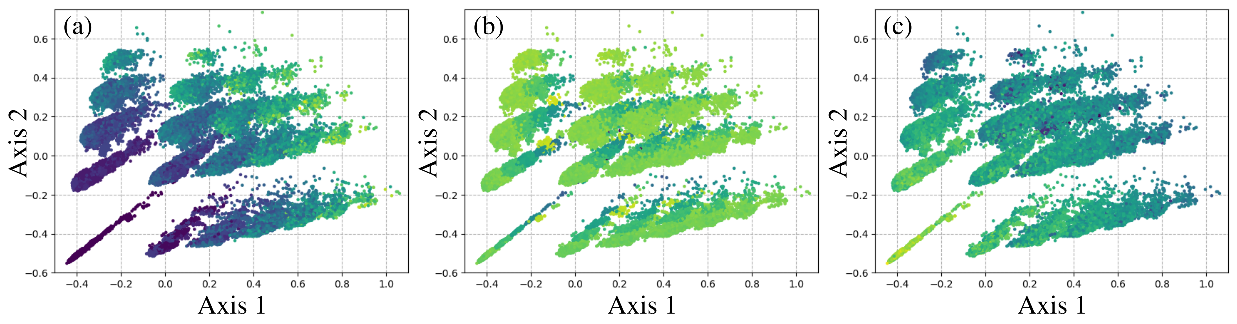


FIG. 2. Feature space generated using graph scattering network from molecular graphs with atom type information as signal for 100000 molecules from QM9 dataset. Illustration is colored by different molecular properties, including (a) PSA, (b) MolWt, and (c) LogP.

**Algorithm 1** Graph scattering transform.

```
Input molecular graph training dataset $\{\mathcal{G}^{(i)}\}_{i=1}^{K} = \{\boldsymbol{W}^{(i)}, \boldsymbol{f}^{(i)}\}_{i=1}^{K}$ and system of filters $\{\hat{\boldsymbol{g}}'_j\}_{j=1}^{\mathcal{J}}$.
for $i = 1, \ldots, K$ do
  Compute the normalized Laplacian matrix $\tilde{\mathcal{L}}^{(i)}$ (Eq. 7).
  Compute the eigenvalues $\tilde{\lambda}_{\ell}^{(i)}$ and eigenvectors $\tilde{\chi}_{\ell}^{(i)}$ of the normalized Laplacian $\tilde{\mathcal{L}}^{(i)}$.
end for
Compute the warping function $\omega$ from the normalized Laplacian spectra $\{\tilde{\lambda}_{\ell}^{(i)}\}_{i=1,\ldots,K,\ell=0,\ldots,N-1}$.
Form adaptive kernels $\{\hat{\boldsymbol{g}}_j\}_{j=1}^{\mathcal{J}}$ using the warping function $\omega$ (Eq. 15).
for $i = 1, \ldots, K$ do
  Compute frame $\hat{\boldsymbol{g}}(\tilde{\mathcal{L}})$ (Eq. 18).
  Compute 0-th order scattering coefficient $\mathcal{S}_0$ (Eq. 22).
  for $m = 1, \ldots, M$ do
    Compute $\mathcal{U}_m$ (Eq. 22).
    Compute $m$-th order scattering coefficient $\mathcal{S}_m$ (Eq. 22).
  end for
end for
```

In Table I we compare the proposed method with benchmark methods[27,49] within a regression problem setting. We generate features from molecular graphs using each method and use these features to predict molecular properties using ridge regression. We use $K = 10000$ data with $5-$fold cross-validation to compute errors. In the present work, node features consist of atom

TABLE I. Prediction performance of scattering coefficients.

| Model | TPSA | | MolWt | | LogP | |
|---|---|---|---|---|---|---|
| | MAE | RMSE | MAE | RMSE | MAE | RMSE |
| Spectral Wavelet Scattering[a] | 4.92 | 6.54 | 1.38 | 1.78 | 0.40 | 0.54 |
| Tight Frame Wavelet Scattering[b] | 4.38 | 5.55 | 1.34 | 1.69 | 0.37 | 0.50 |
| Adaptive Kernel Scattering[c] | 2.52 | 3.31 | 1.04 | 1.34 | 0.31 | 0.43 |

[a] Scattering is performed using Ref. 48's implementation.

[b] Scattering performed with scaling function and wavelets[41] as $\{\hat{g}_j\}_{j=1}^{\mathcal{J}}$.

[c] Scattering performed with adaptive kernels $\{\hat{g}_j\}_{j=1}^{\mathcal{J}}$ (Current work).

types. It is noteworthy that we can improve the performance of the prediction model by including additional atom features. However, this is out of the scope of the current work. The reader is referred to Ref. 5 for a more detailed discussion.

After extracting features in the scattering layers, we pass them to conventional FCN layers followed by batch normalization. In this sense, the encoder is regarded as a hybrid network. In the first FCN layer, a linear layer followed by a batch normalization and an activation function are used to extract features from the input scattering coefficients. In other words, the additional trainable layer learns features that were not extracted in the predefined scattering layers.

Finally, two linear layers are used to find the hyperparameters for the variational approximate posterior distribution $q_\phi$ of the latent variable $\boldsymbol{z}$. These layers take the high-dimensional data from the previous hidden layer and project it to a lower-dimensional latent space. In this sense, these linear layers learn a probabilistic projection of the extracted feature maps to the latent space. It is worth emphasizing that the parameters for the classical FCN layers of the encoder are learned during the training, whereas scattering layers do not require training.

## B. Decoding

After discussing the encoding architecture of the model, we focus on detailing the decoding network. In this work, the decoding network takes samples from the latent space and generates molecular graphs. Hence, the output includes two components: (i) a weight matrix that represents the topology of the graph and (ii) a signal vector that indicates the atoms of the molecule. To fulfill this task, the decoder is constructed of two generators that are jointly trained on a graph dataset. The input to the decoder is from the latent space constructed by the encoder. Given a sample from the latent space, $D_1$ generates a graph by means of its weighted adjacency matrix $\boldsymbol{W}$. Given the output from this generator, the signal decoder $D_2$ generates a signal $\boldsymbol{f}$ which describes the atom sitting on each node. The output from $D_1$ provides a domain for the signal from $D_2$ to reside on.

The decoder $D_1$ defines the discrete probability distribution $p_{\boldsymbol{\theta}}(\boldsymbol{W}|\boldsymbol{z})$ of the discrete variable $\boldsymbol{W}$. This can be seen as the joint distribution of bond type for all the edges $\varepsilon_{i,j}$ in the graph. The categorical distribution $p_{\boldsymbol{\theta}}(W_{i,j}|\boldsymbol{z})$ gives the probability value for each possible category of types of the covalent bond between a pair of atoms

$$\widetilde{\boldsymbol{W}} = p_{\boldsymbol{\theta}}(\boldsymbol{W}|\boldsymbol{z}) = \prod_{i=1}^{N} \prod_{\substack{j=1 \\ j\neq i}}^{N} p_{\boldsymbol{\theta}}\left(W_{i,j}|\boldsymbol{z}\right). \tag{23}$$

Then, the argmax function in Eq. 2 is used to turn these values into a weighted adjacency matrix by mapping discrete variable $\boldsymbol{z}$ to the highest probability class of weights.

The decoder $D_2$ yields a conditional probability distribution $p_{\boldsymbol{\theta}}(\boldsymbol{f}|\boldsymbol{z},\boldsymbol{W})$ of the discrete variable $\boldsymbol{f}$ given the discrete variable $\boldsymbol{W}$ for each point in the latent space $Z$. A categorical distribution $\tilde{\boldsymbol{f}}$ contains the probability values for each class of the variable $\boldsymbol{f}$.

$$\tilde{\boldsymbol{f}} = p_{\theta}(\boldsymbol{f}|\boldsymbol{z},\boldsymbol{W}) = \prod_{i=1}^{N} p_{\theta}\left(f_i|\boldsymbol{z},\boldsymbol{W}\right). \tag{24}$$

Finally, the argmax function (Eq. 1) is used to convert the discrete probability vectors $\tilde{\boldsymbol{f}}$ into a set of one-hot vectors $\boldsymbol{f}$. This maps variable $\boldsymbol{z}$ to the class with the highest probability, which is the mode of the categorical distribution. Here, an FCN followed by a softmax layer is used as the atom generator. As the FCN layers find the unnormalized scores for each category, this softmax layer turns these scores that can take any positive or negative values into normalized probability values.

The weight matrix decoding network is constructed in three steps. First, an FCN $\hat{D}_1$ takes samples from the latent space and output a non-symmetric tensor. Then, unnormalized score values for each class are constructed by

$$\bar{\boldsymbol{W}}(\boldsymbol{z}) = \sigma(\hat{D}_1(\boldsymbol{z})\hat{D}_1(\boldsymbol{z})^T), \tag{25}$$

where $\sigma$ is a non-linear layer. The output from the first FCN layers is multiplied by its transpose to ensure symmetry of the final output probability tensor of the weight matrix for the undirected molecular graph. Lastly, a softmax layer turns these scores $\bar{\boldsymbol{W}}$ into probability values $\tilde{\boldsymbol{W}}$.

To sum it up, $D_1$, along with $D_2$, define a probabilistic mapping from the latent space representation to the molecular graph domain

$$p_\theta(\mathcal{G}|\boldsymbol{z}) = p_\theta(\boldsymbol{W}, \boldsymbol{f}|\boldsymbol{z}) = p_\theta(\boldsymbol{f}|\boldsymbol{z}, \boldsymbol{W}) p_\theta(\boldsymbol{W}|\boldsymbol{z}). \tag{26}$$

This yields a probability distribution from which we take graph samples

$$\boldsymbol{W}, \boldsymbol{f} \sim p_\theta(\mathcal{G}|\boldsymbol{z}) \tag{27}$$

When considering a one-to-one mapping, we map the $\boldsymbol{z}$ variable to the most probable class in the categorical distribution for each possible node and edge in the graph using Eqs. 1 and 2 as follows:

$$\mathcal{G} = (\text{arg max}(\widetilde{\boldsymbol{W}}), \text{arg max}(\widetilde{\boldsymbol{f}})). \tag{28}$$

In essence, the decoding network tackles a classification problem for every node and edge in the graph. When dealing with molecular graphs, for the node signal $\boldsymbol{f}$, the target class includes the heavy atom types in the dataset along with the case of empty node. An empty node, or null vertex, means that no atoms reside on this node and the molecule has less atoms than the predefined maximum possible number of atoms. In a similar manner, the classes for each edge include the possible types of the covalent bond between the respective atoms plus null, which means that there are no covalent bonds between the corresponding pair of atoms.

Using the probabilistic graph model (Eq. 26), we can write the negative expected reconstruction loss term in Eq. 6 as

$$\begin{aligned}
\mathbb{E}_{q_\phi}\left[\log p_{\boldsymbol{\theta}}(\mathcal{G}|\boldsymbol{z})\right] =& \mathbb{E}_{q_\phi}\left[\log p_{\boldsymbol{\theta}}(\boldsymbol{W}, \boldsymbol{f}|\boldsymbol{z})\right] \\
=& \mathbb{E}_{q_\phi}\left[\log\left(p_{\boldsymbol{\theta}}(\boldsymbol{W}|\boldsymbol{z})\, p_{\boldsymbol{\theta}}(\boldsymbol{f}|\boldsymbol{z}, \boldsymbol{W})\right)\right] \\
=& \mathbb{E}_{q_\phi}\left[\log p_{\boldsymbol{\theta}}(\boldsymbol{W}|\boldsymbol{z}) + \log p_{\boldsymbol{\theta}}(\boldsymbol{f}|\boldsymbol{z}, \boldsymbol{W})\right] \\
=& \mathbb{E}_{q_\phi}\left[\sum_{i=1}^{N}\sum_{\substack{j=1 \\ j\neq i}}^{N} \log p_{\boldsymbol{\theta}}\left(W_{i,j}|\boldsymbol{z}\right) + \sum_{i=1}^{N} \log p_{\boldsymbol{\theta}}\left(f_i|\boldsymbol{z}, \boldsymbol{W}\right)\right] \\
=& \sum_{i=1}^{N}\sum_{\substack{j=1 \\ j\neq i}}^{N} \mathbb{E}_{q_\phi}\left[\log p_{\boldsymbol{\theta}}\left(W_{i,j}|\boldsymbol{z}\right)\right] + \sum_{i=1}^{N} \mathbb{E}_{q_\phi}\left[\log p_{\boldsymbol{\theta}}\left(f_i|\boldsymbol{z}, \boldsymbol{W}\right)\right].
\end{aligned} \tag{29}$$

Note that computing the loss for each node and each edge is a multi-class classification problem where we want to find the correct class for each node and edge in the graph. We take advantage of generalized form of cross-entropy for multi-class problems to compute the reconstruction error.

This computes the relative entropy between the predicted probability and the true probability over node and edge classes. Given probability vector $\tilde{\boldsymbol{W}}_{i,j}$ for edge $\varepsilon_{i,j}$, we can write the negative reconstruction error term for edge $\varepsilon_{i,j}$ as

$$\log p_{\boldsymbol{\theta}}\left(W_{i,j}|\boldsymbol{z}\right) = -\mathcal{H}(\boldsymbol{t}^{\varepsilon_{i,j}}, \tilde{\boldsymbol{W}}_{i,j}) = \sum_{c=1}^{C_W} t_c^{\varepsilon_{i,j}} \log\left(\tilde{W}_{i,j,c}\right), \tag{30}$$

where $\mathcal{H}$ denotes the cross-entropy between the target distribution $\boldsymbol{t}^{\varepsilon_{i,j}}$ and predicted distribution $\tilde{\boldsymbol{W}}_{i,j}$ for node $\varepsilon_{i,j}$, index $c$ denotes class index, and $C_W$ is the total number of classes for edges. Furthermore, given the probability vector $\tilde{\boldsymbol{f}}_i$ for node $v_i$, we compute the negative reconstruction error term for node $v_i$ as

$$\log p_{\boldsymbol{\theta}}\left(f_i|\boldsymbol{z},\boldsymbol{W}\right) = -\mathcal{H}(\boldsymbol{t}^{v_i}, \tilde{\boldsymbol{f}}_i) = \sum_{c=1}^{C_f} t_c^{v_i} \log\left(\tilde{f}_{i,c}\right), \tag{31}$$

where $\boldsymbol{t}^{v_i}$ and $\tilde{\boldsymbol{f}}_i$ are the target and predicted distributions for node $v_i$, respectively, and $C_f$ is the total number of classes for nodes.

Hence, we can write the reconstruction loss of the decoding network by summing the loss over all possible nodes and edges in the graph. Note that this biases the loss toward the edges. To overcome this issue, we average the edge reconstruction losses and bond reconstruction loss and then add these two terms to obtain the reconstruction loss for the whole network. Hence, the reconstruction loss for data point $t$ has the form

$$L^{(t)}\left(D_1, D_2\right) = \frac{1}{N}\sum_{i=i}^{N}\sum_{c=1}^{C_f} -t_c^{v_i}\log(D_2(z^{(t)})_{i,c}) + \frac{1}{N(N-1)}\sum_{i=1}^{N}\sum_{\substack{j=1\\ j\neq i}}^{N}\sum_{c=1}^{C_W} -t_c^{\varepsilon_{i,j}}\log(D_1(z^{(t)})_{i,j,c}). \tag{32}$$

## IV. RESULTS

In this section, we provide details of the training of the encoder and graph and signal decoders. The implementation would be available at `https://github.com/zabaras/GSVAE` after publication. The latent representation of the molecular dataset is obtained and the generative model is used to produce realistic molecules. We assess the model in terms of the chemical validity of the generated molecules, their novelty, and uniqueness. In addition, we provide probabilistic estimates of molecular properties. Our focus is on computing these statistics using a limited number of training data points.

In this work, we use a subset of 600 molecular graphs from the QM9 dataset to train our network. The QM9 database[50,51] consists of 133885 small drug-like organic molecules. These molecules are constructed of a maximum of 9 heavy atoms, which include Carbon, Oxygen, Nitrogen, and Florine. To visualize the latent space, we use a subset of 30000 molecular graphs from the test set, which were not seen by the network before. As the latent space dimension is higher than 2, we cannot directly visualize this space on the plane. For the illustration purposes, we perform PCA to transform the data to a two-dimensional space. These points are then colored based on various corresponding molecular properties.

Furthermore, we have computed a number of molecular properties to show the performance of the model. The physicochemical properties used here include Polar Surface Area (PSA)[52], which is a measure of polarity of a molecule and is the sum of the surface areas of all polar atoms in the molecule, Molecular Weight (MolWt), which is the sum of atomic weights for the atoms of the molecule, where the atomic weights are the weighted average of atomic isotopes based on their abundance in nature, and octanol-water partition coefficient (LogP), which amounts for lipophilicity of the molecule.

## 1. *Model specification*

We provide here details of the generative model and the approximate variational posterior inference network. The generative model is constructed of a prior on the latent space variable $p_{\boldsymbol{\theta}}(\boldsymbol{z})$ and a mapping from the latent space to the graph domain $p_{\boldsymbol{\theta}}(\mathcal{G}|\boldsymbol{z})$. The variational posterior $q_{\boldsymbol{\phi}}(\boldsymbol{z}|\mathcal{G})$ approximates the posterior on $\boldsymbol{z}$.

In this model, we assume a standard normal prior distribution over $\boldsymbol{z}$

$$p_{\boldsymbol{\theta}}(\boldsymbol{z}) = \mathcal{N}(\boldsymbol{z};\boldsymbol{0},\boldsymbol{I}). \tag{33}$$

The probabilistic mapping from the latent space to the molecular domain $p_{\boldsymbol{\theta}}(\mathcal{G}|\boldsymbol{z})$ consists of two components: (i) a map from the latent space to the graph signals and (ii) a projection from the latent space to the weighted graph adjacency matrix. These are modeled with the probabilities

$$p_{\boldsymbol{\theta}}(\boldsymbol{W}|\boldsymbol{z}) = \text{softmax}(h_{\boldsymbol{\theta}}^{\boldsymbol{W}}(\boldsymbol{z})) \qquad \text{and} \qquad p_{\boldsymbol{\theta}}(\boldsymbol{f}|\boldsymbol{z},\boldsymbol{W}) = \text{softmax}(h_{\boldsymbol{\theta}}^{\boldsymbol{f}}(\boldsymbol{z})), \tag{34}$$

where the non-linear mapping $\boldsymbol{z} \mapsto h_{\boldsymbol{\theta}}(\boldsymbol{z})$ is parameterized by a deep neural network and the superscript indicates the output. Moving to the inference network, we parameterize the variational approximate $q_{\boldsymbol{\phi}}(\boldsymbol{z}|\mathcal{G})$ of the posterior $p_{\boldsymbol{\theta}}(\boldsymbol{z}|\mathcal{G})$ by a Gaussian distribution

$$q_{\boldsymbol{\phi}}(\boldsymbol{z}|\mathcal{G}) = \mathcal{N}(\boldsymbol{z};\boldsymbol{\mu}_{\boldsymbol{\phi}}(\mathcal{G}),\boldsymbol{S}_{\boldsymbol{\phi}}(\mathcal{G})). \tag{35}$$

We assume that the covariance matrix $\boldsymbol{S}_{\boldsymbol{\phi}}$ is a diagonal matrix. The hyperparameters of the distribution are found using the networks

$$\boldsymbol{\mu}_{\boldsymbol{\phi}}(\mathcal{G}) = h^{\mu}_{\boldsymbol{\phi}}(\mathcal{G}) \qquad \text{and} \qquad \log \boldsymbol{\sigma}^2_{\boldsymbol{\phi}}(\mathcal{G}) = h^{\sigma}_{\boldsymbol{\phi}}(\mathcal{G}). \tag{36}$$

Note that the variance has to be a positive value by definition. In order to make training easier, we train the model to find $\log \boldsymbol{\sigma}^2_{\boldsymbol{\phi}}$ instead and use its exponential to define the covariance matrix as $\boldsymbol{S}_{\boldsymbol{\phi}} = diag(\boldsymbol{\sigma}^2_{\boldsymbol{\phi}}(\mathcal{G}))$.

Using Eq. 36, we can generate a latent space $Z$. We reformulate the variational approximation model in Eq. 35 as

$$\boldsymbol{z} = \boldsymbol{\mu}_{\boldsymbol{\phi}}(\mathcal{G}) + \boldsymbol{S}_{\boldsymbol{\phi}}(\mathcal{G}) \odot \boldsymbol{\epsilon}, \tag{37}$$

where $\boldsymbol{\epsilon}$ has a standard normal distribution

$$p(\boldsymbol{\epsilon}) = \mathcal{N}(\boldsymbol{\epsilon}; \boldsymbol{0}, \boldsymbol{I}), \tag{38}$$

with $\odot$ denoting element-wise product. Given the input latent space variable $\boldsymbol{z}$, the decoding network $h^{\boldsymbol{W}}_{\boldsymbol{\theta}}(\boldsymbol{z})$ is trained to generate the graph weight matrix $\boldsymbol{W}$. To this end, we incorporate the following structure

$$h^{\boldsymbol{W}}_{\boldsymbol{\theta}}(\boldsymbol{z}) = \left(a^{(5)} \circ h_{\boldsymbol{\theta}} h^{T}_{\boldsymbol{\theta}}\right)(\boldsymbol{z}), \tag{39}$$

where

$$h_{\boldsymbol{\theta}}(\boldsymbol{z}) = \left(a^{(4)} \circ l^{(4)}_{\boldsymbol{\theta}} \circ a^{(3)} \circ l^{(3)}_{\boldsymbol{\theta}} \circ a^{(2)} \circ l^{(2)}_{\boldsymbol{\theta}} \circ a^{(1)} \circ l^{(1)}_{\boldsymbol{\theta}}\right)(\boldsymbol{z}). \tag{40}$$

The term $h_{\boldsymbol{\theta}} h^{T}_{\boldsymbol{\theta}}$ ensures the symmetry of the weight matrix for the output undirected graph. In the next step, given the weight matrix, the network computes the signal $\boldsymbol{f}$ residing on the graph's nodes, as illustrated in Fig. 3. As $h^{\boldsymbol{f}}_{\boldsymbol{\theta}}(\boldsymbol{z})$ shares layers with $h^{\boldsymbol{W}}_{\boldsymbol{\theta}}(\boldsymbol{z})$, we can write

$$h^{\boldsymbol{f}}_{\boldsymbol{\theta}}(\boldsymbol{z}) = (a^{(6)} \circ l^{(6)}_{\boldsymbol{\theta}} \circ h^{\boldsymbol{W}}_{\boldsymbol{\theta}})(\boldsymbol{z}). \tag{41}$$

After going through a cascade of convolutions with spectral kernels $\hat{\boldsymbol{g}}$ and modulus non-linear activation function $\rho$, information from different layers is aggregated in layer $\mathcal{A}$, where an average pooling operation $\eta$ constructs invariant feature maps from the input graphs. For 3 layer scattering, the output can be formulated as

$$\mathcal{A}(\mathcal{G}) = (\eta \circ \rho \circ \hat{\boldsymbol{g}} \circ \rho \circ \hat{\boldsymbol{g}} \frown \eta \circ \rho \circ \hat{\boldsymbol{g}} \frown \eta)(\mathcal{G}), \tag{42}$$

where $\frown$ shows concatenation which gathers $0-$th, $1-$st, and $2-$nd order scattering coefficients. As discussed earlier, the scattering layers use predefined filters $\hat{\boldsymbol{g}}$ to extract feature maps and act

TABLE II. Decoding network architecture specification.

| Linear layer | Input dimension | Output dimension | Activation layer | Activation function |
|---|---|---|---|---|
| $l_\theta^{(1)}$ | $J$ | 60 | $a^{(1)}$ | Leaky ReLU |
| $l_\theta^{(2)}$ | 60 | 120 | $a^{(2)}$ | Leaky ReLU |
| $l_\theta^{(3)}$ | 120 | 240 | $a^{(3)}$ | Leaky ReLU |
| $l_\theta^{(4)}$ | 240 | 288 | $a^{(4)}$ | Leaky ReLU |
| - | – | – | $a^{(5)}$ | Leaky ReLU |
| $l_\theta^{(6)}$ | 324 | 45 | $a_f^{(6)}$ | Leaky ReLU |

as a bridge between the traditional FCN encoder and the input molecular graphs. However, the scattering layers reduce the need for a deeper FCN. As a result, we merely incorporate a single linear layer followed by a batch normalization layer and a non-linear layer to learn features that were not captured in the scattering layers, after which two linear layers $l_\phi^{(2)}$ and $l_\phi^{(3)}$, the former of which is followed by a batch normalization layer $n_\phi^{(2)}$, are used to compute the parameters $\boldsymbol{\mu}_\phi$ and $\boldsymbol{\sigma}_\phi$ of the variational posterior $q_\phi$ of the variable $\boldsymbol{z}$. We can summarize these networks as

$$h_\phi^\mu(\mathcal{G}) = (l_\phi^{(3)} \circ h_\phi)(\mathcal{G}) \qquad \text{and} \qquad h_\phi^\sigma(\mathcal{G}) = (n_\phi^{(2)} \circ l_\phi^{(2)} \circ h_\phi)(\mathcal{G}), \tag{43}$$

with the shared network structured as

$$h_\phi(\mathcal{G}) = (\tilde{a}^{(1)} \circ n_\phi^{(1)} \circ l_\phi^{(1)} \circ \mathcal{A})(\mathcal{G}). \tag{44}$$

In these equations, a linear layer is representing the operation $l(\boldsymbol{x}) = \mathscr{W}\boldsymbol{x} + \boldsymbol{b}$, where $\mathscr{W}$ and $\boldsymbol{b}$ are learnable parameters and batch normalization represents $n(\boldsymbol{x}) = \boldsymbol{\tau} \odot \hat{\boldsymbol{x}} + \boldsymbol{\beta}$, where $\hat{\boldsymbol{x}}$ is normalized input, $\boldsymbol{\tau}$ and $\boldsymbol{\beta}$ are learnable parameters, and $\odot$ is element-wise product. The indices $\boldsymbol{\theta}$ and $\boldsymbol{\phi}$ distinguish between the weights and biases of the decoder $\boldsymbol{\theta} = \{\mathscr{W}_\theta^{(1)}, \mathscr{W}_\theta^{(2)}, \mathscr{W}_\theta^{(3)}, \mathscr{W}_\theta^{(4)}, \mathscr{W}_\theta^{(6)}, \boldsymbol{b}_\theta^{(1)}, \boldsymbol{b}_\theta^{(2)}, \boldsymbol{b}_\theta^{(3)}, \boldsymbol{b}_\theta^{(4)}, \boldsymbol{b}_\theta^{(6)}\}$ and the encoder $\boldsymbol{\phi} = \{\mathscr{W}_\phi^{(1)}, \mathscr{W}_\phi^{(2)}, \mathscr{W}_\phi^{(3)}, \boldsymbol{b}_\phi^{(1)}, \boldsymbol{b}_\phi^{(2)}, \boldsymbol{b}_\phi^{(3)}, \boldsymbol{\tau}_\phi^{(1)}, \boldsymbol{\tau}_\phi^{(2)}, \boldsymbol{\beta}_\phi^{(1)}, \boldsymbol{\beta}_\phi^{(2)}, \}$. $a$ denotes the non-linear activation functions for the decoding network and $\tilde{a}$ shows the non-linear layers of the encoding network. Fig. 3 summarizes the encoding and decoding network architectures and the training procedure.

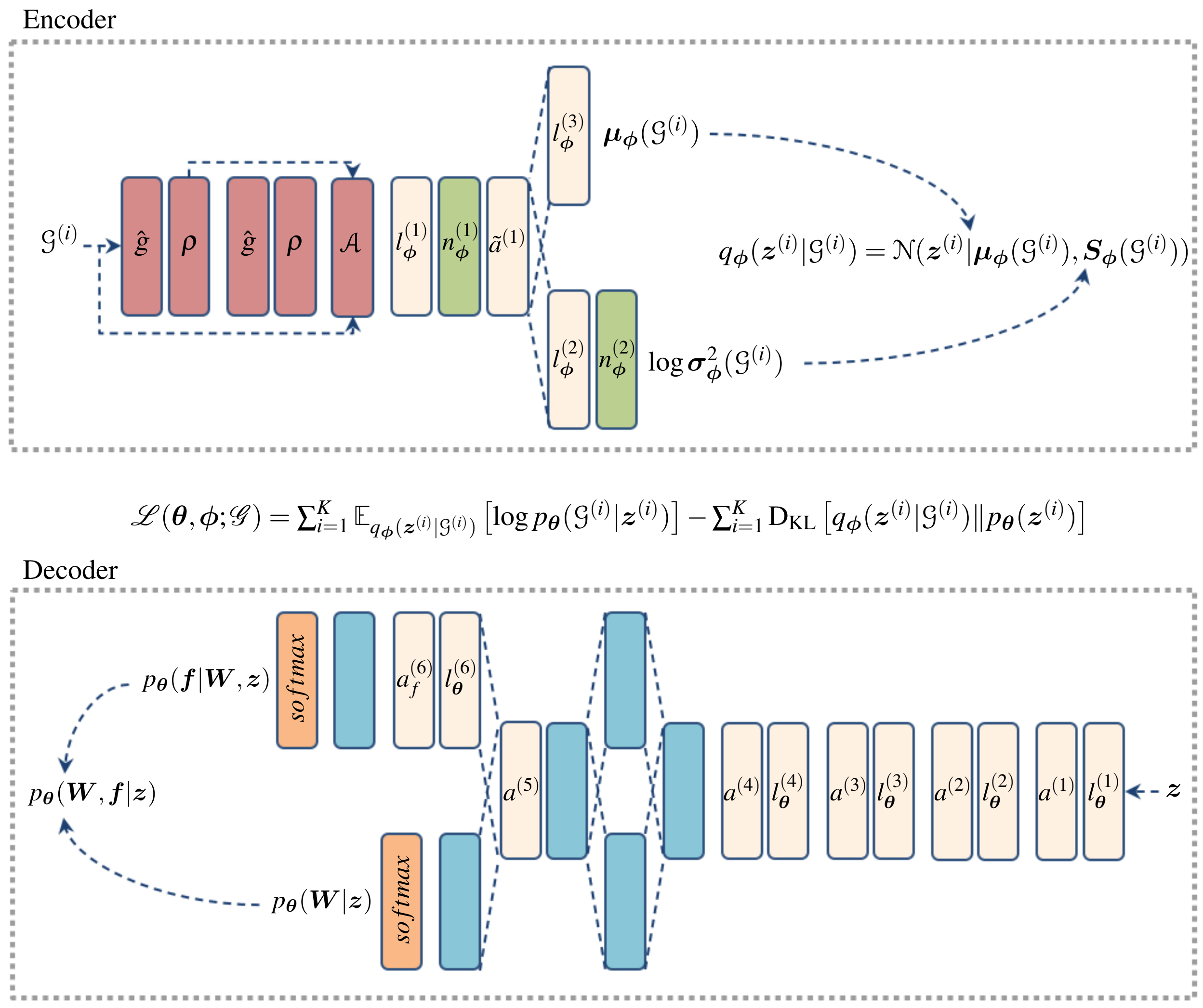

Encoder
$\mathcal{G}^{(i)}$
$\hat{g}$ $\rho$ $\hat{g}$ $\rho$ $\mathcal{A}$
$l_\phi^{(1)}$ $n_\phi^{(1)}$ $\tilde{a}^{(1)}$
$l_\phi^{(3)}$ $\boldsymbol{\mu}_\phi(\mathcal{G}^{(i)})$
$l_\phi^{(2)}$ $n_\phi^{(2)}$ $\log \boldsymbol{\sigma}_\phi^2(\mathcal{G}^{(i)})$
$q_\phi(\boldsymbol{z}^{(i)}|\mathcal{G}^{(i)}) = \mathcal{N}(\boldsymbol{z}^{(i)}|\boldsymbol{\mu}_\phi(\mathcal{G}^{(i)}), \boldsymbol{S}_\phi(\mathcal{G}^{(i)}))$
$\mathscr{L}(\boldsymbol{\theta},\boldsymbol{\phi};\mathscr{G}) = \sum_{i=1}^K \mathbb{E}_{q_\phi(\boldsymbol{z}^{(i)}|\mathcal{G}^{(i)})}\left[\log p_{\boldsymbol{\theta}}(\mathcal{G}^{(i)}|\boldsymbol{z}^{(i)})\right] - \sum_{i=1}^K \mathrm{D}_{\mathrm{KL}}\left[q_\phi(\boldsymbol{z}^{(i)}|\mathcal{G}^{(i)}) \| p_{\boldsymbol{\theta}}(\boldsymbol{z}^{(i)})\right]$
Decoder
$p_{\boldsymbol{\theta}}(\boldsymbol{f}|\boldsymbol{W},\boldsymbol{z})$
$p_{\boldsymbol{\theta}}(\boldsymbol{W},\boldsymbol{f}|\boldsymbol{z})$
$p_{\boldsymbol{\theta}}(\boldsymbol{W}|\boldsymbol{z})$
softmax
softmax
$a_f^{(6)}$ $l_\theta^{(6)}$
$a^{(5)}$
$a^{(4)}$ $l_\theta^{(4)}$ $a^{(3)}$ $l_\theta^{(3)}$ $a^{(2)}$ $l_\theta^{(2)}$ $a^{(1)}$ $l_\theta^{(1)}$
$\boldsymbol{z}$

FIG. 3. Schematic representation of the network architecture. We show the scattering layers in red. Shown in light beige, $l$ represents the fully-connected layers and $a$ and $\tilde{a}$ show the activation function for the decoding and encoding networks, respectively. Batch normalization layers are shown in green. Permutation, transpose, and multiplication layers are shown in blue, and softmax layer is shown in orange. Model is trained by maximizing the lower-bound $\mathscr{L}(\boldsymbol{\theta},\boldsymbol{\phi};\mathscr{G})$ for the training dataset $\mathscr{G} = \{\mathcal{G}^{(i)}\}_{i=1}^K$.

In order to visualize the latent space $Z$, we start with projecting molecular graphs to the latent space using the encoding network. These points however are in a $J$-dimensional feature space and cannot be shown on the 2D plane. Hence, we use Principle Component Analysis (PCA) to map this $J$-dimensional data onto a $2-$dimensional plane. In Fig. 4 we have trained the model using $K = 600$ training data and encoded 30000 molecular graphs from test set to a $J = 30$ dimensional feature space and used 2 principal axis to represent the feature space on the plane. We notice that in Fig. 4-b, molecules with different number of heavy atoms are distinct in the shades of green color, where molecules with 9 heavy atoms are clustered in the center with a light green color. As we move in the positive direction of principle axis 1, molecules have lower number of heavy atoms. Fig. 8 compares the latent space of the model trained with two different training data size $K$. We can observe similar pattern to Fig. 4-b with the model trained with $K = 200$ molecular graphs.

In order to observe the variability within each cluster of particular heavy atom number, in Fig. 5, we have isolated the visualization of molecules with 9 heavy atoms from Fig. 4. Comparing Fig. 5 to Fig. 4, we see that the patterns in Figs. 5-a and 5-b are repeated in each molecule size in Figs. 4-a and 4-c, respectively.

Comparing these results with the latent representation of SMILES VAEs in 6, we notice that unlike SMILES VAEs that do not have an organized latent space and need to be hooked with a property predictor to arrange the molecules based on their properties, graph VAEs have some degree of organization in their latent representation. This is due to the fact that graph VAEs also incorporate the graph that represents the underlying structure of the molecule, which provides some of the information needed to estimate the molecular properties. This can be improved further by incorporating additional atomic features such as hybridization.

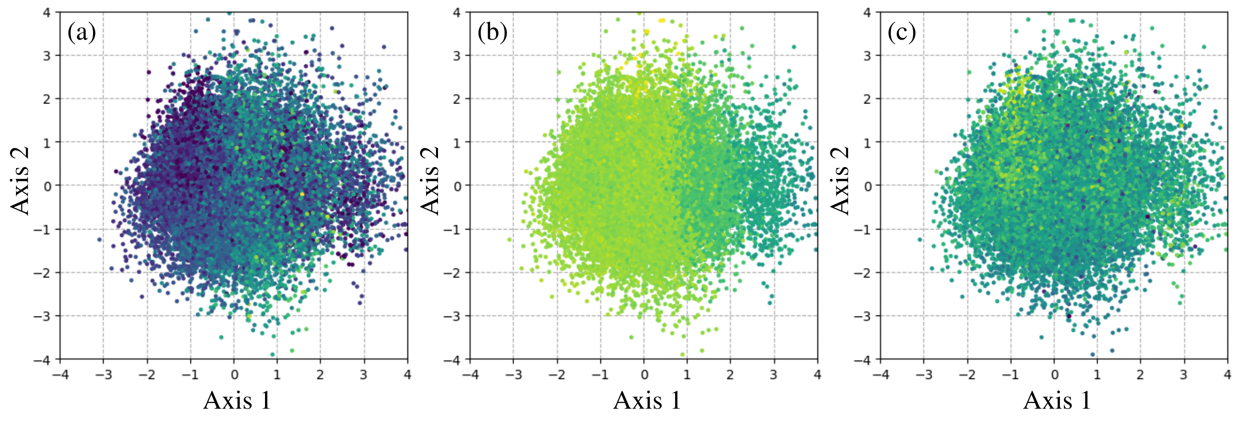


FIG. 4. Feature space generated by graph scattering VAE using 30000 molecular graphs from QM9 database as test set. The model is trained with $K = 600$ data. Here, the graph signals are consist of atom type information. Latent representations are colored by different properties (a) Polar surface area, (b) Molecular Weight, and (c) Octanol-water partition coefficient.

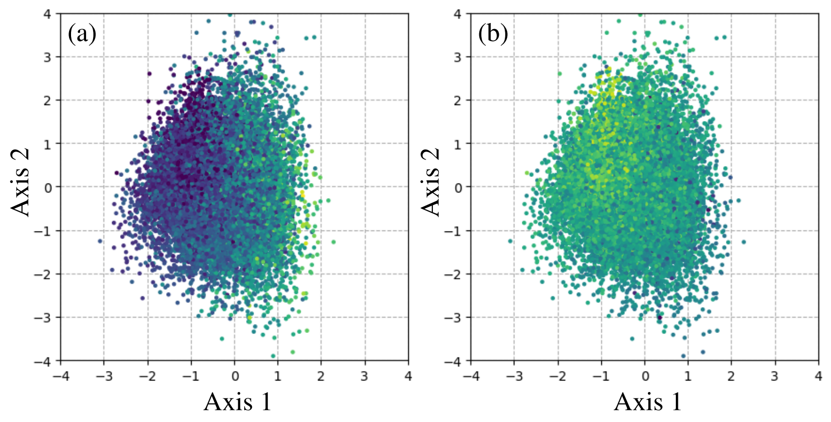


FIG. 5. Feature space generated in Fig. 4 with only the molecular graphs with 9 heavy atoms visible. Each molecule's representation is colored by different properties (a) Polar surface area and (b) Octanol-water partition coefficient.

We further analyze the feature space by constructing a latent contour map in Fig. 6. We construct a $2-$dimensional grid and project it onto the $J-$dimensional latent space using a projection matrix. In order to find the projection map, we encode the training data onto the $J-$dimensional latent space and find a projection matrix to the $2-$dimensional space using PCA. We use the transpose of this matrix to yield points in the feature space whose projection is a grid on the plan of the principle axis. In addition, this helps the grid to represent the same region of the $J-$dimensional space that the molecules are present. We then pass the $J-$dimensional grid points to the decoding network which in return outputs molecular graphs. As discussed later on in this section, there is no guarantee that the generated molecules have a chemically valid combination of atoms and bonds. As a result, some grid points might correspond to molecules that are chemically invalid, for which the physicochemical properties cannot be estimated. After filtering the invalid molecules, we estimate the property values which gives us the property map. Note that since the grid points corresponding to the invalid molecules have been eliminated, Gaussian Process (GP) regression is used to provide a smooth contour map.

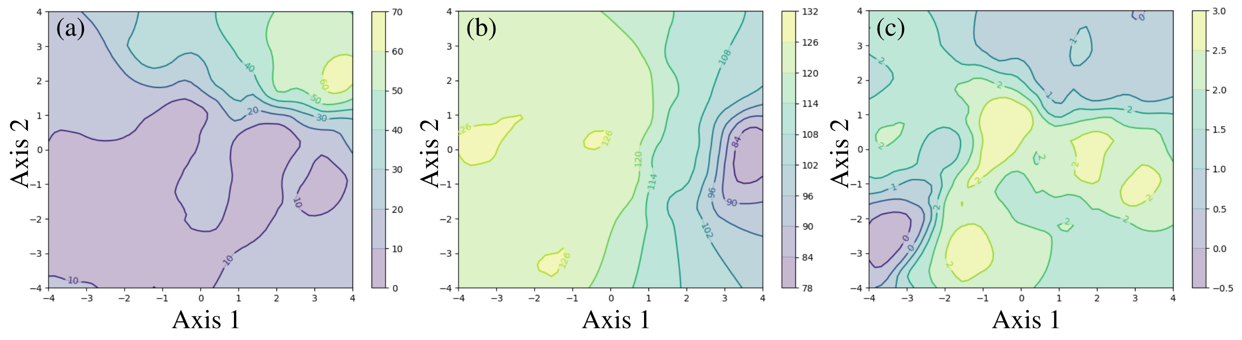


FIG. 6. Feature space map of the QM9 dataset generated by graph scattering VAE. We have used $K = 600$ training data to obtain the model. Contours are colored by different physicochemical properties (a) Polar Surface Area, b) Molecular Weight, and c) Octanol-water partition coefficient).

The chemical space of the molecules defined by their physicochemical properties plays an important role in drug design[53]. As an example, Lipinski's rule of five (Ro5)[54] suggests limits on LogP, molecular weight, and hydrogen bond donors and acceptors to ensure that the drug-like properties are maintained during the optimization of the molecular structure. As a result, it is important for the predictions of the model to give a reliable estimate of these chemical spaces. Figure 7 compares the chemical space distribution of the training set constructed by LogP and molecular weight with the one for the predicted molecules. The last column is the chemical space of the database, which is constructed by 108000 samples from the QM9 database.

In Fig. 7, we further compare the models trained with various sizes of the training set. We observe that the model performs well with a training size as low as $K = 50$ training data and can yield a reliable estimate of the chemical space of the database. Figure 8 shows a comparison between the latent space of the models with two different training size $K = 200$ and $K = 600$.

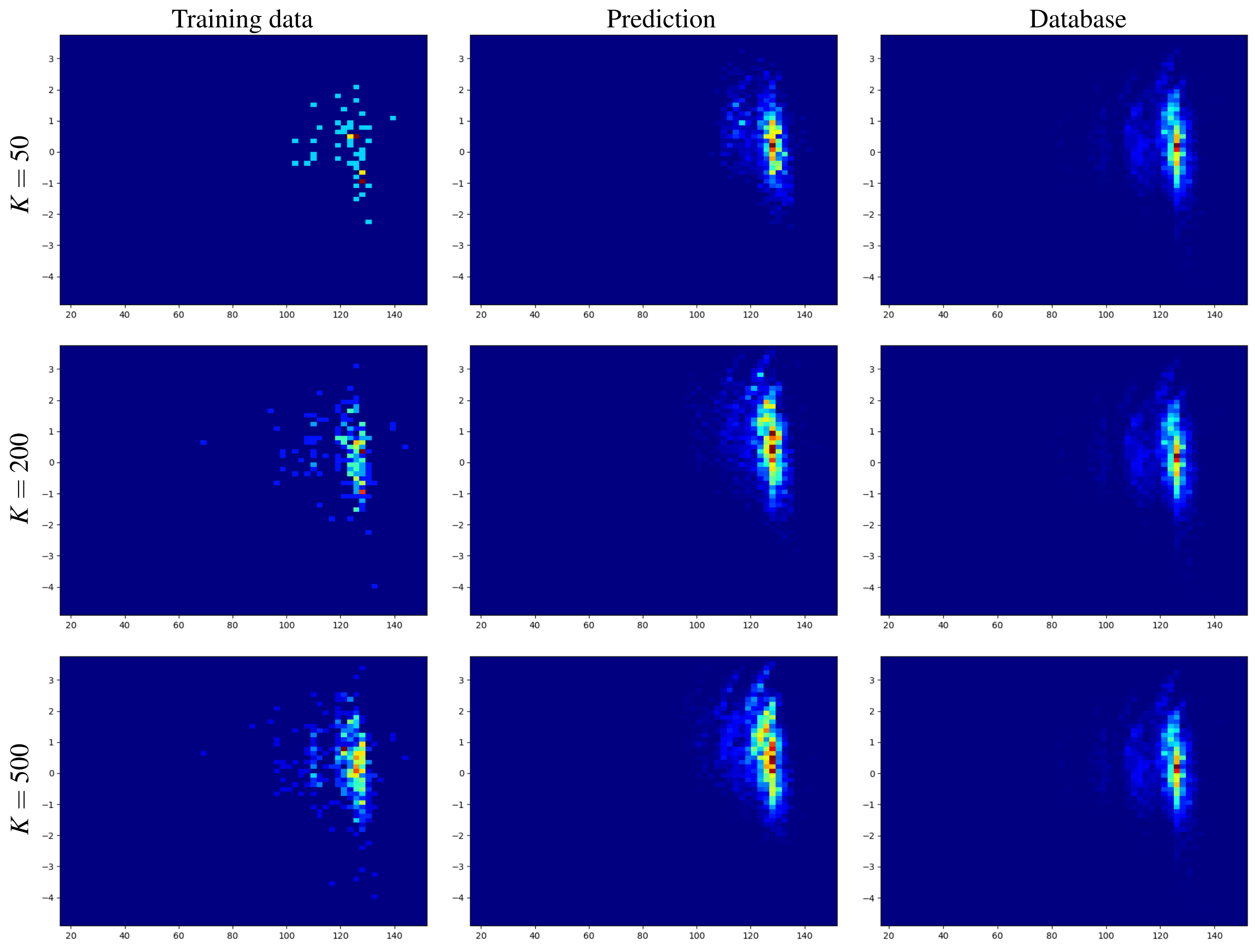


FIG. 7. Predicted chemical space of QM9 dataset defined by octanol-water partition coefficients (LogP) on the $y-$axis based on molecular weight ($x-$axis) for various number of training data $K$. Plots are generated by sampling 10000 molecules. On the right column, 108000 samples from QM9 database are used to plot the joint map.

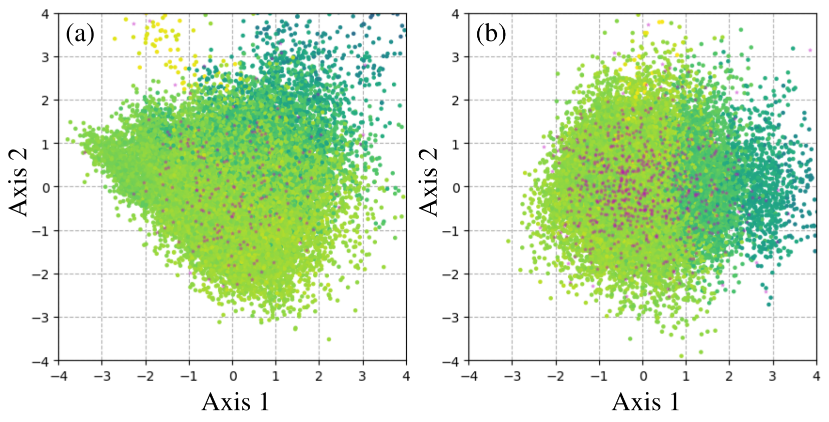


FIG. 8. Latent space representation of graph scattering VAE trained with different training dataset sizes (a) $K = 200$, and (b) $K = 600$. The latent space is constructed by 30000 molecular graphs from the test set, which are colored by molecular weight value. The latent representation of the training data are shown in magenta.

Finally, in Fig. 9 we examine the latent space by interpolating between the feature space representation of two molecules. We randomly select two molecules from the dataset and project them onto the latent space using the encoding network. Then, we take samples in equal intervals on the path that connects the two representations. These latent space samples are projected to the graph domain using the decoding network. Comparing the generated molecules we see the transition from one molecule to the other in the latent space.

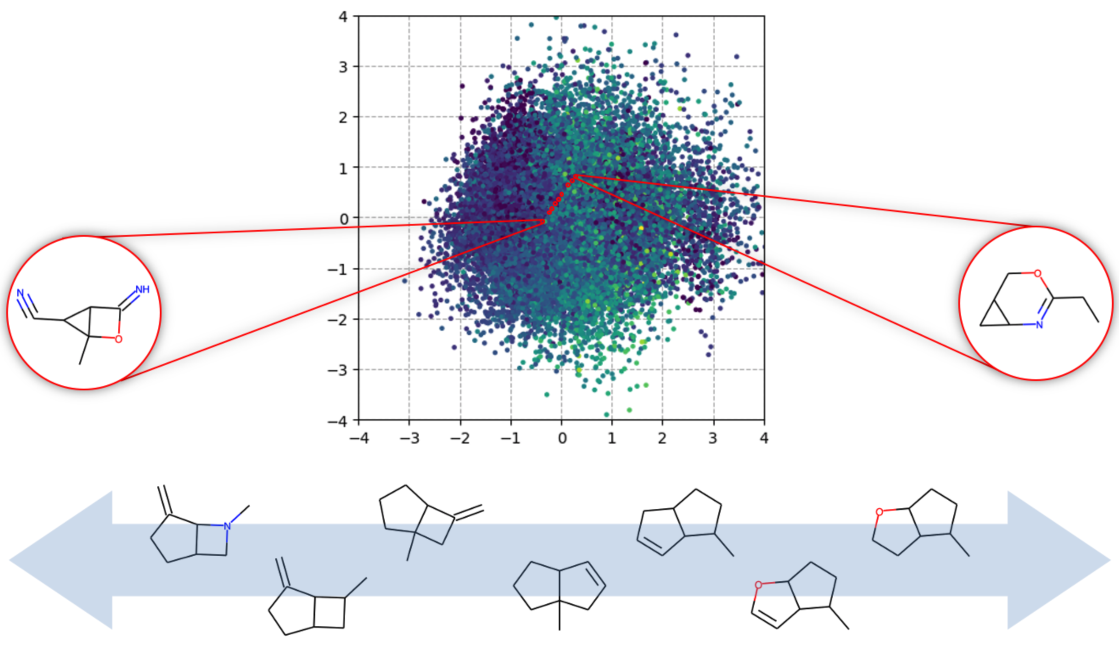


FIG. 9. Interpolation between the latent representations of two molecules. The model is trained using $K = 600$ data and the latent space is visualized using 30000 molecules from QM9 database, which are colored by their PSA value.

As noted earlier, not all the molecular graphs sampled using the generative model have chemically valid structures. To assess the quality of the results, we use triple quality scores including validity, uniqueness, and novelty scores. The validity score indicates if a predicted graph has a valid molecular representation

$$\text{Validity} = \frac{|valid(\bar{\mathscr{G}})|}{|\bar{\mathscr{G}}|}, \tag{45}$$

where $\bar{\mathscr{G}} = \{\bar{\mathfrak{G}}^{(i)}\}_{i=1}^{T}$ denotes the collection of sampled molecules. Note that a valid molecule refers to a single connected graph with no violation of the valency of the atoms. Figure 10 illustrates examples of valid molecules sampled using the generative model. Moreover, the uniqueness score indicates what portion of the molecular graphs are unique among the sampled outputs

$$\text{Uniqueness} = \frac{|\bar{\mathscr{G}}^*|}{|valid(\bar{\mathscr{G}})|}. \tag{46}$$

Here, $\bar{\mathscr{G}}^*$ shows the set of sampled molecules. Lastly, the novelty score indicates if the generated molecular graphs were present in the training dataset or if they are novel molecules

$$\text{Novelty} = \frac{|valid(\bar{\mathscr{G}})| - |\mathscr{G} \cap valid(\bar{\mathscr{G}})|}{|valid(\bar{\mathscr{G}})|}, \tag{47}$$

where $\mathscr{G} = \{\mathcal{G}^{(i)}\}_{i=1}^{K}$ stands for the training molecules.

In Table III, we examine the quality scores for $T = 10000$ molecules generated by the model, which is trained using $K = 600$ molecular graphs. In this table we have distinguished the percentage of the molecules with each validity issues, namely connectivity and valency. Note that the number of the valid molecules and molecules with each validity issue might not sum to 10000 as some molecules might have both issues. Table IV further compares the present work with benchmark models. It can be seen that the current model outperforms the benchmarks in novelty and uniqueness scores, while it is second only to MolGAN[11] in validity score. We believe that a balance between quality scores of the model is preferred over a high score in one category and a low score in the other, as in 11.

We further examine the performance of the model under physical constraints. We use the method proposed by 34 to impose validity constraints, including connectivity and valency. As can be seen in Table III, using these constraints generally results in a decrease in the uniqueness score. In addition, 34 uses a trial and error approach to tweak the Lagrange multipliers for yielding the best validity score, while this does not guarantee satisfaction of the Karush-Kuhn-Tucker (KKT) conditions. Addressing these issues is subjected to further investigation.

TABLE III. Quality metrics for graph scattering variational autoencoder.

| Type | Validity | | | Uniqueness | Novelty |
|---|---|---|---|---|---|
| | Total | Connectivity | Valency | | |
| No constraint | 73.86% | 5.86% | 20.92% | 76.73% | 95.91% |
| With constraint | 78.16% | 5.79% | 16.36% | 65.62% | 98.67% |

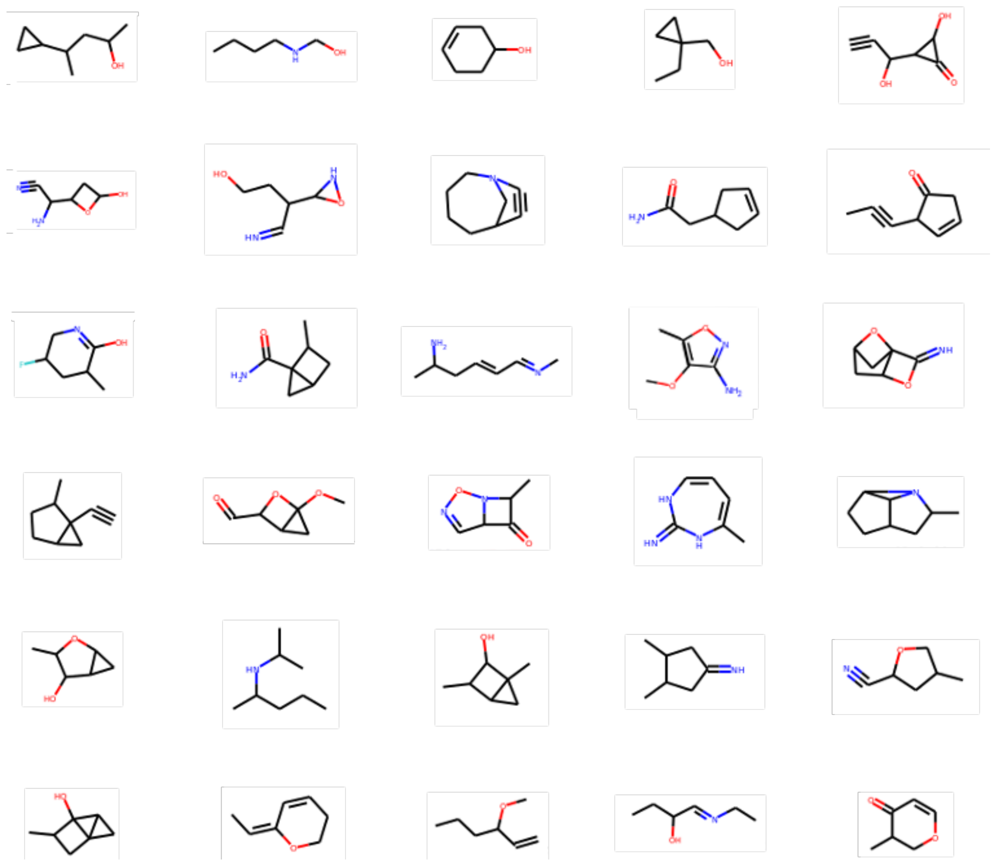


FIG. 10. Example of valid molecules sampled using the generative model with training set size $K = 600$. Atom colors follow the CPK coloring convention[55].

### 2. *Model uncertainty*

Bayesian inference has been used for quantifying uncertainty in VAE models[35]. Quantifying uncertainties in model parameters results in error bars over estimated physicochemical properties which shows our confidence over them.

TABLE IV. Comparison of quality metrics for different molecular generative models.

| Method | Validity | Uniqueness | Novelty |
|---|---|---|---|
| CVAE[6a] | 10.3% | 67.5% | 90.0% |
| GVAE[8a] | 60.2% | 9.3% | 80.9% |
| GraphVAE[12a] | 55.7% | 76.0% | 61.6% |
| MolGAN[11a] | 98.1% | 10.4% | 94.2% |
| GSVAE | 73.86% | 76.73% | 95.91% |

[a] Baseline values are reported from Ref. 11.

Given a set of $K$ i.i.d. observations $\mathscr{G} = \left\{\mathcal{G}^{(1)},\ldots,\mathcal{G}^{(K)}\right\}$ sampled from $p_{target}(\mathcal{G})$, we are interested in finding a model $p(\mathcal{G})$ parameterized by $\boldsymbol{\theta}$ that closely resembles $p_{target}(\mathcal{G})$. This can be achieved by minimizing the KL distance between $p(\mathcal{G})$ and $p_{target}(\mathcal{G})$.

$$D_{KL}(p_{target}(\mathcal{G})||p(\mathcal{G})) = -\int p_{target}(\mathcal{G})\log p(\mathcal{G})d\mathcal{G} + \int p_{target}(\mathcal{G})\log p_{target}(\mathcal{G})d\mathcal{G}. \tag{48}$$

It can be shown that we can equivalently train a model $p(\mathcal{G}|\boldsymbol{\theta})$ by maximizing marginal log-likelihood

$$\boldsymbol{\theta}_{\text{MLE}} = \arg\max_{\boldsymbol{\theta}} \log p(\mathscr{G} \mid \boldsymbol{\theta}) = \arg\max_{\boldsymbol{\theta}} \sum_{k=1}^{K} \log p\left(\mathcal{G}^{(k)} \mid \boldsymbol{\theta}\right). \tag{49}$$

Computation of marginal log-likelihood requires intractable integration. To overcome this, Eq. 6 defines a lower-bound on marginal log-likelihood by introducing an auxiliary density $q_{\boldsymbol{\phi}}$ parameterized by $\boldsymbol{\phi}$. Therefore, we find MLE estimate by maximizing the lower-bound on marginal log-likelihood

$$\boldsymbol{\theta}_{\text{MLE}}, \boldsymbol{\phi}_{\text{MLE}} = \arg\max_{\boldsymbol{\theta},\boldsymbol{\phi}} \mathscr{L}(\boldsymbol{\theta},\boldsymbol{\phi};\mathscr{G}) = \arg\max_{\boldsymbol{\theta},\boldsymbol{\phi}} \sum_{k=1}^{K} \mathscr{L}(\boldsymbol{\theta},\boldsymbol{\phi};\mathcal{G}^{(k)}). \tag{50}$$

To study the uncertainty in the trained model, we can take advantage of predictive distribution

$$p(\bar{\mathcal{G}} \mid \mathscr{G}) = \int p(\bar{\mathcal{G}} \mid \boldsymbol{\theta}) p(\boldsymbol{\theta} \mid \mathscr{G}) d\boldsymbol{\theta}. \tag{51}$$

In Eq. 51, quantifying uncertainties in $\boldsymbol{\theta}$ enables us to capture the epistemic uncertainties introduced by the training data. We can summarize the procedure in the following steps:

- Draw a posterior sample $\boldsymbol{\theta}^j \sim p(\boldsymbol{\theta}|\mathscr{G})$.
- Obtain predictive samples $\bar{\mathcal{G}}_j^{(t)}$, with $t = 1, \ldots, T$, given $\boldsymbol{\theta}^j$.

As noted above, this requires a full posterior over model parameters $p(\boldsymbol{\theta}|\mathscr{G})$. Finding this posterior is computationally impractical. One common way is to approximate the posterior distribution $p(\boldsymbol{\theta}|\mathscr{G})$ with a Gaussian distribution using Laplace method. This method requires computation of Hessian of the log-posterior.

In some problems, this normal distribution gives a poor approximation to the full posterior of model parameters. We observed that the models with $\boldsymbol{\theta}$ drawn from the approximated Gaussian distribution suffer from extremely low validity of the sampled molecules. Additionally, computation of Hessian can be overwhelmingly expensive. Taking these into consideration, Newton and Raftery[56] proposed weighted likelihood bootstrap (WLB) as a way to simulate approximately from the posterior distribution. This method is a direct extension of Rubin's Bayesian bootstrap[40].

Given dataset $\mathscr{G} = \{\mathcal{G}^{(1)}, \ldots, \mathcal{G}^{(K)}\}$, bootstrap method[57] generates multiple samples $\tilde{\mathscr{G}}_1, \ldots, \tilde{\mathscr{G}}_B$ by sampling from $\mathscr{G}$, with replacement. In classical bootstrap, the sampling weights $\pi_k$ associated with data $\mathcal{G}^{(k)}$ are drawn from the discrete set $\{0, \frac{1}{K}, \ldots, \frac{K}{K}\}$, where the numerator is the number of times $n_k$ that data $\mathcal{G}^{(k)}$ is in the resampled dataset and the denominator is the size of the dataset $K = |\tilde{\mathscr{G}}|$.

Rubin[40] presented a Bayesian analog for bootstrap by treating sampling weight $\boldsymbol{\pi}$ as an unknown variable and drawing them from a posterior distribution over $\boldsymbol{\pi}$. By imposing an improper, non-informative, Dirichlet prior distribution

$$p(\boldsymbol{\pi}) = \mathcal{D}ir(\boldsymbol{\pi}; \boldsymbol{\alpha}), \quad \text{with } \boldsymbol{\alpha} = [0, \ldots, 0] \in \mathbb{R}^K. \tag{52}$$

over $\boldsymbol{\pi}$ and observing sample $\mathscr{G}$, Bayes rule can be used to derive the posterior distribution over sampling weights

$$\begin{aligned} p(\boldsymbol{\pi}|\mathscr{G}) &\propto p(\mathscr{G}|\boldsymbol{\pi}) p(\boldsymbol{\pi}) \\ &\propto \prod_k \pi_k^{n_k} \prod_k \pi_k^{\alpha_k - 1} \\ &\propto \prod_k \pi_k^{n_k + \alpha_k - 1}. \end{aligned} \tag{53}$$

where for observation $\mathscr{G}$, $n_1 = \cdots = n_K = 1$. From Eqs 52 and 53, the posterior over bootstrap sampling weights follows a Dirichlet distribution over the bounded finite-dimensional space

$$p(\boldsymbol{\pi}|\mathscr{G}) = \mathcal{D}ir(\boldsymbol{\pi}; \boldsymbol{\alpha}'), \quad \text{with } \boldsymbol{\alpha}' = [1, \ldots, 1] \in \mathbb{R}^K. \tag{54}$$

A resampled dataset $\tilde{\mathscr{G}}$ can be denoted by the original dataset $\mathscr{G}$ and the associated resampling weights $\boldsymbol{\pi}$ as $\tilde{\mathscr{G}} = (\mathscr{G}, \boldsymbol{\pi})$.

In the problems where the solution for model parameters can be computed using a Maximum Likelihood Estimate (MLE), Newton and Raftery[56] reformulate maximizing the likelihood as maximizing a weighted likelihood estimate

$$\boldsymbol{\theta}_{\text{MWLE}}(\boldsymbol{\pi}) = \underset{\boldsymbol{\theta}}{\arg\max} \sum_{k=1}^{K} \pi_k \log p(\mathcal{G}^{(k)}|\boldsymbol{\theta}). \tag{55}$$

Since

$$\log p(\mathcal{G}^{(k)}|\boldsymbol{\theta}) \geq \mathscr{L}(\boldsymbol{\theta}, \boldsymbol{\phi}; \mathcal{G}^{(k)}), \tag{56}$$

and because $\pi_k$ has a positive value, we can define a lower-bound on the weighted marginal log-likelihood. Therefore, $\boldsymbol{\theta}_{\text{MWLE}}$ can be computed by

$$\boldsymbol{\theta}_{\text{MWLE}}, \boldsymbol{\phi}_{\text{MWLE}} = \underset{\boldsymbol{\theta},\boldsymbol{\phi}}{\arg\max} \sum_{k=1}^{K} \pi_k \mathscr{L}(\boldsymbol{\theta}, \boldsymbol{\phi}; \mathcal{G}^{(k)}). \tag{57}$$

Newton and Raftery[56] simulate approximately from a posterior distribution over $\boldsymbol{\theta}$ by repeated sampling from posterior distribution $p(\boldsymbol{\pi}|\mathscr{G})$ and maximizing a weighted likelihood to calculate $\boldsymbol{\theta}_{\text{MWLE}}$. The method can be summarized as

- Draw a posterior sample $\boldsymbol{\pi} \sim p(\boldsymbol{\pi}|\mathscr{G}) = \mathcal{D}ir(1, \ldots, 1)$.
- Calculate $\boldsymbol{\theta}_{\text{MWLE}}$ from weighted sample $\tilde{\mathscr{G}} = (\mathscr{G}, \boldsymbol{\pi})$

Fushiki[39] takes advantage of this approximation to propose Bayesian bootstrap predictive distribution

$$p_{BB}(\bar{\mathcal{G}} \mid \mathscr{G}) = \int p\left(\bar{\mathcal{G}} \mid \boldsymbol{\theta}_{\text{MWLE}}(\boldsymbol{\pi})\right) p(\boldsymbol{\pi} \mid \mathscr{G}) d\boldsymbol{\pi}. \tag{58}$$

A Monte Carlo (MC) estimate of the predictive distribution (Eq. 58)

$$p(\bar{\mathcal{G}}|\mathscr{G}) = \frac{1}{B} \sum_{b=1}^{B} p(\bar{\mathcal{G}}|\boldsymbol{\theta}_{\text{MWLE}}(\boldsymbol{\pi}^b)), \tag{59}$$

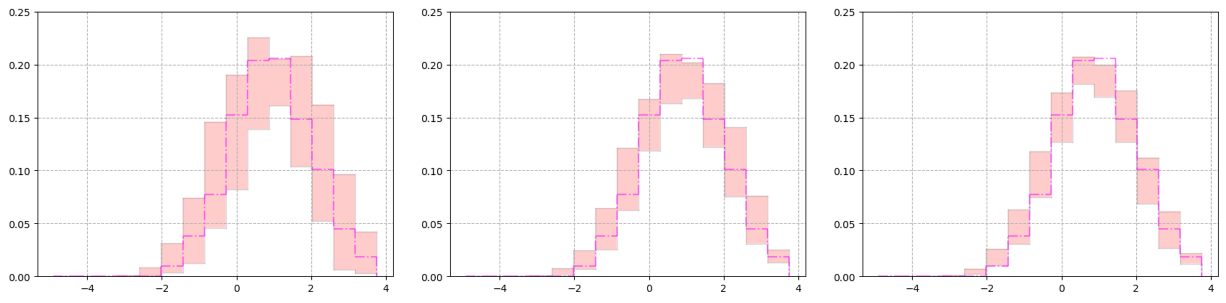
is obtained by drawing $B$ sampling weights $\boldsymbol{\pi}^{\boldsymbol{b}}$ of size $K$ from Eq. 54. Following the suggested interval in the literature[58,59], we use $B = 25$ to estimate the credible intervals. Algorithm 2 describes computation of credible intervals.

**Algorithm 2** Estimation of credible intervals.

```
**Input** $B$ the number of bootstrap samples to be drawn and $T$ the number of predictive samples.
for $b = 1,\dots,B$ do
    Draw a posterior sample $\boldsymbol{\pi}^b \sim p(\boldsymbol{\pi}|\mathscr{G})$ (Eq. 54).
    Calculate $\boldsymbol{\theta}^b_{\mathrm{MWLE}}$ from weighted sample $(\mathscr{G}, \boldsymbol{\pi}^b)$ (Eq. 55).
    Obtain predictive samples $\bar{\mathcal{G}}^{(t)}_b$, with $t = 1,\dots,T$, given $\boldsymbol{\theta}^b_{\mathrm{MWLE}}$ (Eq. 58).
    Estimate properties $\hat{a}(\boldsymbol{\theta}^b_{\mathrm{MWLE}}) = \frac{1}{T}\sum_{t=1}^{T} a(\bar{\mathcal{G}}^{(t)}_b)$, given the predictive samples $\bar{\mathcal{G}}^{(t)}_b$, with $t = 1,\dots,T$.
end for
Estimate credible intervals from $\hat{a}(\boldsymbol{\theta}^{1:B}_{\mathrm{MWLE}})$.
```

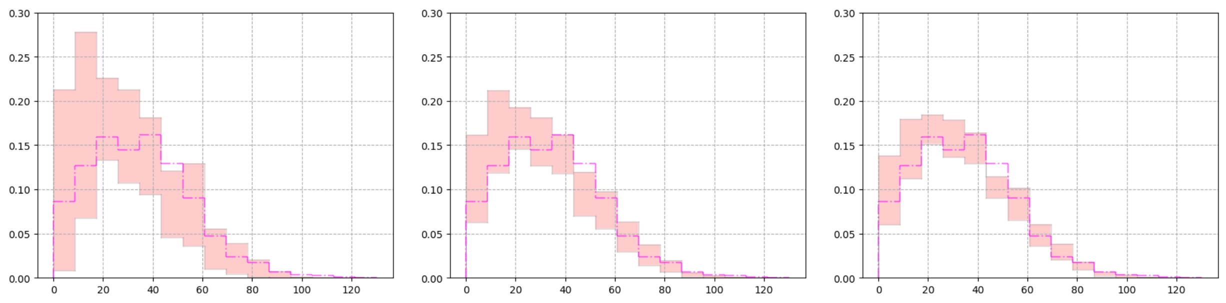
In this section, we are interested in developing a predictive model for physicochemical properties of interest. We can use samples from the trained model to compute the predictive estimates of physicochemical molecular properties. To demonstrate this, we begin by selecting properties that can be readily estimated from the generated molecules: i) octanol-water partition coefficient (LogP) which is computed using an atomic contribution scheme[60] and ii) polar surface area (PSA)[52]. We estimate these properties using RDKit[61], which is an open-source cheminformatics software.

Furthermore, we can use samples from the predictive distribution to estimate credible intervals over physicochemical molecular properties, as detailed in Algorithm 2. Figures 11 and 12 illustrate the computed error bars over LogP and PSA, respectively. In these figures, we have trained the model with 3 different small training dataset sizes and simulated approximately from the posterior distribution using WLB to construct the credible intervals. To this end, we sample 10000 molecules from the model and use the valid molecules to plot the error bars. We compare this to the reference solution computed using $K = 10000$ training data. We observe that the reference solution falls within the shaded 90% credible interval, which represents the epistemic uncertainty in the model. As we increase the size of the training dataset, the probabilistic confidence over estimated properties increases.



(a) $K = 50$ (b) $K = 200$ (c) $K = 500$

FIG. 11. Predicted octanol-water partition coefficients (logP) with latent space dimension $J = 30$ for various sizes $K$ of the training dataset for the QM9 database. The reference solution, indicated in magenta (—·—), is estimated by sampling from the model trained by $K = 10000$ molecules. The shaded area represents the 90% credible interval, reflecting the induced epistemic uncertainty from the limited amount of the training data.



(a) $K = 50$ (b) $K = 200$ (c) $K = 500$

FIG. 12. Predicted Polar Surface Area (PSA) with latent space dimension $J = 30$ for various sizes $K$ of the training dataset for the QM9 database. The reference solution, indicated in magenta (—·—), is estimated by sampling from the model trained by $K = 10000$ molecules. The shaded area represents the 90% credible interval, reflecting the induced epistemic uncertainty from the limited amount of the training data.

## V. CONCLUSIONS

In this work, we presented a generative model for constructing molecular graphs based on a VAE framework. As the inference network, this model uses a hybrid graph scattering network which consists of layers of graph scattering transform followed by fully-connected layers. We constructed scattering layers using adaptive spectral filters, which are tailored to the training dataset.

This increases the discriminatory power of the encoding network by reducing the correlation between the neighbouring kernels. We further assessed the feature maps generated by the scattering layers in a regression task and show that the features generated with adaptive filters yield lower prediction error than the benchmark methods. For the decoding network, we use a one-shot graph generation model that first computes the connections between the atoms and then conditions atom features on the underlying topology of the molecules. Therefore, the FCN that takes unnormalized scores for edge types as input and outputs the probabilistic atom types is implicitly learning the relation and physical constraints for chemically valid combinations of atoms and bonds. We further analyzed the latent space of the generative model by visualizing it with different physicochemical properties of the molecule. We see that unlike VAEs based on SMILES representation, graph based models show a degree of organization in the latent space with respect to the molecular properties. This is due to incorporating the underlying structure of the molecules in the input graph. We assessed the performance of the model in the form of quality metrics and predictive capabilities. We compare the generated molecules with benchmark models in three categories of chemical validity, uniqueness, and their novelty. Results show that the present model outperforms benchmarks in novelty and uniqueness and is second only to one other model in the validity score, which, however, performs particularly low on the uniqueness score. We computed predictive estimates and plotted histograms for physicochemical properties of the molecules by sampling from the trained model. We further investigated the predictive capabilities of the model by means of Bayesian bootstrap prediction. We impose a prior distribution on the sampling weights of the resampling scheme and yield predictions by directly passing these weights to the model, which yields error bars over the predicted properties. These error bars quantify the epistemic uncertainties in the model.

## Appendix A: Bootstrap predictive distribution

In this section, we estimate the credible intervals using the traditional bootstrap method. Given an empirical distribution

$$\hat{p}(\mathcal{G}) = \frac{1}{K}\sum_{i=1}^{K} \delta(\mathcal{G} - \mathcal{G}_i) \tag{A1}$$

from which we can sample bootstrap data $\tilde{\mathscr{G}}$, Fushiki et al.[38] introduce a Bootstrap predictive distribution

$$p_B(\bar{\mathcal{G}} \mid \mathscr{G}) = \int p\left(\bar{\mathcal{G}} \mid \boldsymbol{\theta}_{\text{MLE}}(\tilde{\mathscr{G}})\right) \hat{p}(\tilde{\mathscr{G}}) d\tilde{\mathscr{G}} \tag{A2}$$

as an approximation of the Bayesian predictive distribution.

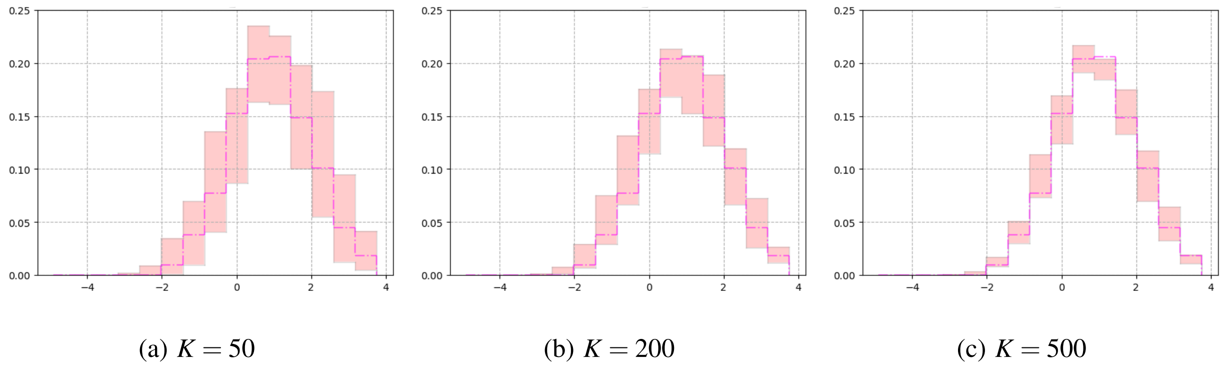
Given bootstrap data $\tilde{\mathscr{G}}$, we train the model using Eq. 49 to obtain MLE solution $\boldsymbol{\theta}_{\text{MLE}}$. By repeated sampling from Eq. A1 and computing the MLE solution, one can write an MC estimate of the predictive distribution (Eq. A2)

$$p(\bar{\mathcal{G}}|\mathscr{G}) = \frac{1}{B}\sum_{b=1}^{B} p(\bar{\mathcal{G}}|\boldsymbol{\theta}_{\text{MLE}}(\tilde{\mathscr{G}}^b)), \tag{A3}$$

The procedure is detailed in Algorithm 3. Figures 13 and 14 show the estimated error bars over LogP and PSA, respectively.

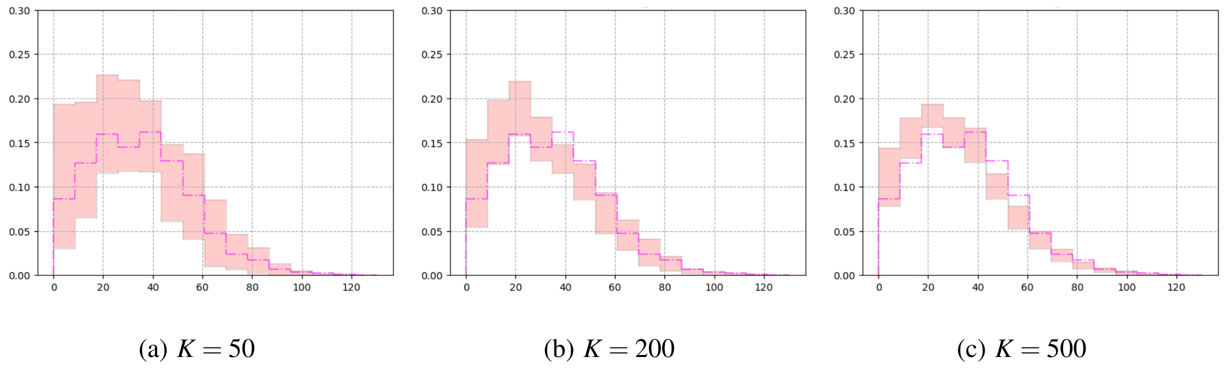
**Algorithm 3** Estimation of credible intervals using bootstrap predictive distribution.

```
Input $B$ the number of bootstrap samples to be drawn and $T$ the number of predictive samples.
for $b = 1, \dots, B$ do
    Generate bootstrap data $\tilde{\mathscr{G}}_b$ from the empirical distribution $\hat{p}(\tilde{\mathscr{G}})$.
    Calculate $\theta^b_{\text{MLE}}$ from bootstrap data $\tilde{\mathscr{G}}_b$ (Eq. 49).
    Obtain predictive samples $\bar{\mathcal{G}}_b^{(t)}$, with $t = 1, \dots, T$, given $\theta^b_{\text{MLE}}$ (Eq. A2).
    Estimate properties $\hat{a}(\theta^b_{\text{MLE}}) = \frac{1}{T}\sum_{t=1}^{T} a(\bar{\mathcal{G}}_b^{(t)})$, given the predictive samples $\bar{\mathcal{G}}_b^{(t)}$, with $t = 1, \dots, T$.
end for
Estimate credible intervals from $\hat{a}(\theta^{1:B}_{\text{MLE}})$.
```



(a) $K = 50$ (b) $K = 200$ (c) $K = 500$

FIG. 13. Predicted octanol-water partition coefficients (logP) with latent space dimension $J = 30$ for various sizes $K$ of the training dataset for the QM9 database. The reference solution, indicated in magenta (—·—), is estimated by sampling from the model trained by $K = 10000$ molecules. The shaded area represents the 90% credible interval, reflecting the induced epistemic uncertainty from the limited amount of training data.



(a) $K = 50$ (b) $K = 200$ (c) $K = 500$

FIG. 14. Predicted Polar Surface Area (PSA) with latent space dimension $J = 30$ for various sizes $K$ of the training dataset for the QM9 database. The reference solution, indicated in magenta (—·—), is estimated by sampling $K = 10000$ molecules. The shaded area represents the 90% credible interval, reflecting the induced epistemic uncertainty from the limited amount of training data.

## REFERENCES

[1] J. Drews, "Drug discovery: a historical perspective," Science **287**, 1960–1964 (2000).

[2] H. Hoppe and N. S. Sariciftci, "Organic solar cells: An overview," Journal of materials research **19**, 1924–1945 (2004).

[3] W. L. Jorgensen, "Efficient drug lead discovery and optimization," Accounts of chemical research **42**, 724–733 (2009).

[4]D. C. Elton, Z. Boukouvalas, M. D. Fuge, and P. W. Chung, "Deep learning for molecular design—a review of the state of the art," Molecular Systems Design & Engineering **4**, 828–849 (2019).

[5]F. A. Faber, L. Hutchison, B. Huang, J. Gilmer, S. S. Schoenholz, G. E. Dahl, O. Vinyals, S. Kearnes, P. F. Riley, and O. A. Von Lilienfeld, "Prediction errors of molecular machine learning models lower than hybrid dft error," Journal of chemical theory and computation **13**, 5255–5264 (2017).

[6]R. Gómez-Bombarelli, J. N. Wei, D. Duvenaud, J. M. Hernández-Lobato, B. Sánchez-Lengeling, D. Sheberla, J. Aguilera-Iparraguirre, T. D. Hirzel, R. P. Adams, and A. Aspuru-Guzik, "Automatic chemical design using a data-driven continuous representation of molecules," ACS central science **4**, 268–276 (2018).

[7]D. Weininger, "Smiles, a chemical language and information system. 1. introduction to methodology and encoding rules," Journal of chemical information and computer sciences **28**, 31–36 (1988).

[8]M. J. Kusner, B. Paige, and J. M. Hernández-Lobato, "Grammar variational autoencoder," in *Proceedings of the 34th International Conference on Machine Learning-Volume 70* (JMLR. org, 2017) pp. 1945–1954.

[9]M. M. Bronstein, J. Bruna, Y. LeCun, A. Szlam, and P. Vandergheynst, "Geometric deep learning: going beyond euclidean data," IEEE Signal Processing Magazine **34**, 18–42 (2017).

[10]J. Gilmer, S. S. Schoenholz, P. F. Riley, O. Vinyals, and G. E. Dahl, "Neural message passing for quantum chemistry," in *Proceedings of the 34th International Conference on Machine Learning-Volume 70* (JMLR. org, 2017) pp. 1263–1272.

[11]N. De Cao and T. Kipf, "Molgan: An implicit generative model for small molecular graphs," arXiv preprint arXiv:1805.11973 (2018).

[12]M. Simonovsky and N. Komodakis, "Graphvae: Towards generation of small graphs using variational autoencoders," in *International Conference on Artificial Neural Networks* (Springer, 2018) pp. 412–422.

[13]L. Dinh, D. Krueger, and Y. Bengio, "Nice: Non-linear independent components estimation," arXiv preprint arXiv:1410.8516 (2014).

[14]K. Madhawa, K. Ishiguro, K. Nakago, and M. Abe, "Graphnvp: An invertible flow model for generating molecular graphs," arXiv preprint arXiv:1905.11600 (2019).

[15]N. Thomas, T. Smidt, S. Kearnes, L. Yang, L. Li, K. Kohlhoff, and P. Riley, "Tensor field net-

works: Rotation-and translation-equivariant neural networks for 3d point clouds," arXiv preprint arXiv:1802.08219 (2018).

[16]M. Weiler, M. Geiger, M. Welling, W. Boomsma, and T. S. Cohen, "3d steerable cnns: Learning rotationally equivariant features in volumetric data," in *Advances in Neural Information Processing Systems* (2018) pp. 10381–10392.

[17]R. Kondor, "N-body networks: a covariant hierarchical neural network architecture for learning atomic potentials," arXiv preprint arXiv:1803.01588 (2018).

[18]S. Jastrzębski, D. Leśniak, and W. M. Czarnecki, "Learning to smile (s)," arXiv preprint arXiv:1602.06289 (2016).

[19]M. H. Segler, T. Kogej, C. Tyrchan, and M. P. Waller, "Generating focused molecule libraries for drug discovery with recurrent neural networks," ACS central science **4**, 120–131 (2018).

[20]H. T. Son, S. Trivedi, H. Pan, B. Anderson, and R. Kondor, "Covariant compositional networks for learning graphs," in *Proceedings of the 15th International Workshop on Mining and Learning with Graphs (MLG)* (2019).

[21]C. Chen, W. Ye, Y. Zuo, C. Zheng, and S. P. Ong, "Graph networks as a universal machine learning framework for molecules and crystals," Chemistry of Materials **31**, 3564–3572 (2019).

[22]C. W. Coley, W. Jin, L. Rogers, T. F. Jamison, T. S. Jaakkola, W. H. Green, R. Barzilay, and K. F. Jensen, "A graph-convolutional neural network model for the prediction of chemical reactivity," Chemical science **10**, 370–377 (2019).

[23]S. Mallat, "Group invariant scattering," Communications on Pure and Applied Mathematics **65**, 1331–1398 (2012).

[24]S. Mallat, "Understanding deep convolutional networks," Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences **374**, 20150203 (2016).

[25]E. Oyallon, S. Zagoruyko, G. Huang, N. Komodakis, S. Lacoste-Julien, M. Blaschko, and E. Belilovsky, "Scattering networks for hybrid representation learning," IEEE transactions on pattern analysis and machine intelligence **41**, 2208–2221 (2018).

[26]X. Chen, X. Cheng, and S. Mallat, "Unsupervised deep haar scattering on graphs," in *Advances in Neural Information Processing Systems* (2014) pp. 1709–1717.

[27]D. Zou and G. Lerman, "Graph convolutional neural networks via scattering," Applied and Computational Harmonic Analysis (2019).

[28]F. Gama, A. Ribeiro, and J. Bruna, "Diffusion scattering transforms on graphs," arXiv preprint arXiv:1806.08829 (2018).

[29]R. R. Coifman and M. Maggioni, "Diffusion wavelets," Applied and Computational Harmonic Analysis **21**, 53–94 (2006).

[30]D. K. Hammond, P. Vandergheynst, and R. Gribonval, "Wavelets on graphs via spectral graph theory," Applied and Computational Harmonic Analysis **30**, 129–150 (2011).

[31]W. Jin, R. Barzilay, and T. Jaakkola, "Junction tree variational autoencoder for molecular graph generation," arXiv preprint arXiv:1802.04364 (2018).

[32]Y. Li, O. Vinyals, C. Dyer, R. Pascanu, and P. Battaglia, "Learning deep generative models of graphs," arXiv preprint arXiv:1803.03324 (2018).

[33]X. Bresson and T. Laurent, "A two-step graph convolutional decoder for molecule generation," arXiv preprint arXiv:1906.03412 (2019).

[34]T. Ma, J. Chen, and C. Xiao, "Constrained generation of semantically valid graphs via regularizing variational autoencoders," in *Advances in Neural Information Processing Systems* (2018) pp. 7113–7124.

[35]M. Schöberl, N. Zabaras, and P.-S. Koutsourelakis, "Predictive collective variable discovery with deep bayesian models," The Journal of chemical physics **150**, 024109 (2019).

[36]D. J. MacKay and D. J. Mac Kay, *Information theory, inference and learning algorithms* (Cambridge university press, 2003).

[37]I. R. Harris, "Predictive fit for natural exponential families," Biometrika **76**, 675–684 (1989).

[38]T. Fushiki, F. Komaki, K. Aihara, *et al.*, "Nonparametric bootstrap prediction," Bernoulli **11**, 293–307 (2005).

[39]T. Fushiki, "Bayesian bootstrap prediction," Journal of statistical planning and inference **140**, 65–74 (2010).

[40]D. B. Rubin, "The bayesian bootstrap," The annals of statistics , 130–134 (1981).

[41]D. I. Shuman, C. Wiesmeyr, N. Holighaus, and P. Vandergheynst, "Spectrum-adapted tight graph wavelet and vertex-frequency frames," IEEE Transactions on Signal Processing **63**, 4223–4235 (2015).

[42]P. Cayley, "Lvii. on the mathematical theory of isomers," The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science **47**, 444–447 (1874).

[43]J. J. Sylvester, "On an application of the new atomic theory to the graphical representation of the invariants and covariants of binary quantics, with three appendices," American Journal of Mathematics **1**, 64–104 (1878).

[44]T. N. Kipf and M. Welling, "Variational graph auto-encoders," NIPS Workshop on Bayesian

Deep Learning (2016).

[45]D. P. Kingma and M. Welling, “Auto-encoding variational bayes,” arXiv preprint arXiv:1312.6114 (2013).

[46]D. Shuman, S. Narang, P. Frossard, A. Ortega, and P. Vandergheynst, “The emerging field of signal processing on graphs: Extending high-dimensional data analysis to networks and other irregular domains,” IEEE Signal Processing Magazine **3**, 83–98 (2013).

[47]A. Ortega, P. Frossard, J. Kovačević, J. M. Moura, and P. Vandergheynst, “Graph signal processing: Overview, challenges, and applications,” Proceedings of the IEEE **106**, 808–828 (2018).

[48]D. Zou and G. Lerman, “Encoding robust representation for graph generation,” (2018), arXiv:1809.10851 [cs.LG].

[49]F. Gama, J. Bruna, and A. Ribeiro, “Stability of graph scattering transforms,” arXiv preprint arXiv:1906.04784 (2019).

[50]R. Ramakrishnan, P. O. Dral, M. Rupp, and O. A. Von Lilienfeld, “Quantum chemistry structures and properties of 134 kilo molecules,” Scientific data **1**, 140022 (2014).

[51]L. Ruddigkeit, R. Van Deursen, L. C. Blum, and J.-L. Reymond, “Enumeration of 166 billion organic small molecules in the chemical universe database gdb-17,” Journal of chemical information and modeling **52**, 2864–2875 (2012).

[52]P. Ertl, B. Rohde, and P. Selzer, “Fast calculation of molecular polar surface area as a sum of fragment-based contributions and its application to the prediction of drug transport properties,” Journal of medicinal chemistry **43**, 3714–3717 (2000).

[53]Y. Kwon, *Handbook of essential pharmacokinetics, pharmacodynamics and drug metabolism for industrial scientists* (Springer Science & Business Media, 2001).

[54]C. A. Lipinski, F. Lombardo, B. W. Dominy, and P. J. Feeney, “Experimental and computational approaches to estimate solubility and permeability in drug discovery and development settings,” Advanced drug delivery reviews **23**, 3–25 (1997).

[55]W. L. Koltun, “Space filling atomic units and connectors for molecular models,” (1965), uS Patent 3,170,246.

[56]M. A. Newton and A. E. Raftery, “Approximate bayesian inference with the weighted likelihood bootstrap,” Journal of the Royal Statistical Society: Series B (Methodological) **56**, 3–26 (1994).

[57]B. Efron, “Bootstrap methods: another look at the jackknife,” in *Breakthroughs in statistics* (Springer, 1992) pp. 569–593.

[58]L. Breiman, “Bagging predictors,” Machine learning **24**, 123–140 (1996).

[59]M. Clyde and H. Lee, "Bagging and the bayesian bootstrap." in *AISTATS* (2001).

[60]S. A. Wildman and G. M. Crippen, "Prediction of physicochemical parameters by atomic contributions," Journal of chemical information and computer sciences **39**, 868–873 (1999).

[61]G. Landrum, "Rdkit: Open-source cheminformatics software," (2016).

[62]M. Andreux, T. Angles, G. Exarchakis, R. Leonarduzzi, G. Rochette, L. Thiry, J. Zarka, S. Mallat, E. Belilovsky, J. Bruna, *et al.*, "Kymatio: Scattering transforms in python," arXiv preprint arXiv:1812.11214 (2018).

[63]B. Sanchez-Lengeling and A. Aspuru-Guzik, "Inverse molecular design using machine learning: Generative models for matter engineering," Science **361**, 360–365 (2018).

[64]N. Leonardi and D. Van De Ville, "Tight wavelet frames on multislice graphs," IEEE Transactions on Signal Processing **61**, 3357–3367 (2013).

[65]T. Sterling and J. J. Irwin, "Zinc 15–ligand discovery for everyone," Journal of chemical information and modeling **55**, 2324–2337 (2015).

[66]J. Snoek, H. Larochelle, and R. P. Adams, "Practical bayesian optimization of machine learning algorithms," in *Advances in neural information processing systems* (2012) pp. 2951–2959.

[67]M. Hirn, S. Mallat, and N. Poilvert, "Wavelet scattering regression of quantum chemical energies," Multiscale Modeling & Simulation **15**, 827–863 (2017).

[68]B. Huang and O. A. Von Lilienfeld, "Communication: Understanding molecular representations in machine learning: The role of uniqueness and target similarity," (2016).

[69]I. Osband, C. Blundell, A. Pritzel, and B. Van Roy, "Deep exploration via bootstrapped dqn," in *Advances in neural information processing systems* (2016) pp. 4026–4034.

[70]Q. Liu, M. Allamanis, M. Brockschmidt, and A. Gaunt, "Constrained graph variational autoencoders for molecule design," in *Advances in neural information processing systems* (2018) pp. 7795–7804.

[71]I. Daubechies, "Orthonormal bases of compactly supported wavelets," Communications on pure and applied mathematics **41**, 909–996 (1988).

[72]W. Sweldens, "The lifting scheme: A custom-design construction of biorthogonal wavelets," Applied and computational harmonic analysis **3**, 186–200 (1996).

[73]M. A. Newton, *The weighted likelihood bootstrap and an algorithm for prepivoting*, Ph.D. thesis, University of Washington (1991).

[74]T. N. Kipf and M. Welling, "Semi-supervised classification with graph convolutional networks," arXiv preprint arXiv:1609.02907 (2016).

[75]M. Fey and J. E. Lenssen, "Fast graph representation learning with PyTorch Geometric," in *ICLR Workshop on Representation Learning on Graphs and Manifolds* (2019).

[76]M. Defferrard, X. Bresson, and P. Vandergheynst, "Convolutional neural networks on graphs with fast localized spectral filtering," in *Advances in neural information processing systems* (2016) pp. 3844–3852.

[77]J. Haldane, "The precision of observed values of small frequencies," Biometrika **35**, 297–300 (1948).