Encoding robust representation for graph generation

Dongmian Zou
Institute for Mathematics and its Applications
University of Minnesota, Twin Cities
Minneapolis, USA
dzou@umn.edu

Gilad Lerman
School of Mathematics
University of Minnesota, Twin Cities
Minneapolis, USA
lerman@umn.edu

Abstract—**Generative networks have made it possible to generate meaningful signals such as images and texts from simple noise. Recently, generative methods based on GAN and VAE were developed for graphs and graph signals. However, the mathematical properties of these methods are unclear, and training good generative models is difficult. This work proposes a graph generation model that uses a recent adaptation of Mallat's scattering transform to graphs. The proposed model is naturally composed of an encoder and a decoder. The encoder is a Gaussianized graph scattering transform, which is robust to signal and graph manipulation. The decoder is a simple fully connected network that is adapted to specific tasks, such as link prediction, signal generation on graphs and full graph and signal generation. The training of our proposed system is efficient since it is only applied to the decoder and the hardware requirements are moderate. Numerical results demonstrate state-of-the-art performance of the proposed system for both link prediction and graph and signal generation.**

I. INTRODUCTION

Generative neural networks have been successfully applied to various tasks such as the generation of images and texts. Their development is based on fruitful methods of deep learning, such as convolutional and recurrent neural networks. These and other methods of deep learning, which were initially developed for problems in the Euclidean domain[1], have been successfully generalized to address supervised learning tasks in the graph domain. In particular, a variety of graph convolutional networks have been developed, including networks with prescribed parameters [1], [2] and trained networks [3]–[6]. It is natural to use these tools to build generative models in the graph domain.

Most generative graph networks directly use standard graph networks by following either of the following two generative frameworks: the generative adversarial network (GAN) [7] and the variational auto-encoder (VAE) [8]. In a GAN, a generator and an auxiliary adversarial discriminator are trained together. On the other hand, in VAE, an encoder and a decoder (or generator) are both trained according to Bayesian models. Both frameworks contain two components (generator and discriminator or encoder and decoder), where each of them requires training. For GAN, training two components corresponds to a difficult min-max problem. On the other hand, training the two components in VAE can be described as a relatively easier

[1]We remark that the Euclidean and graph domains include scenarios whose underlying datasets have Euclidean and graph structures, respectively.

non-convex minimization problem. However, it is a crude approximation to the motivating variational inference formulation. Given an encoding process with guaranteed mathematical properties, one can focus on training only the decoder, which in this case is the generator. In the Euclidean domain, [9] uses the scattering transform as an encoder, which is robust to deformations of input signals, and learn a generative model by minimizing the l_1-loss for reconstructing the training images. We adopt a similar method, using a graph scattering transform as an encoder for graph signals, which is robust to signal and graph manipulations, and train neural networks corresponding to respective tasks. Note that the graph scattering transform can be either carried out in the spectral domain [1] or in the graph domain [2]. More specifically, spectral-domain wavelets [10] are one-dimensional wavelets applied to the eigenvalues of the graph Laplacian, whereas graph-domain wavelets, or diffusion wavelets [11], are multi-scale functions on the vertices of the graph that use dyadic powers of the diffusion operator. The advantage of the former graph scattering transform is its guaranteed robustness to signal manipulation, which is a consequence of its energy preservation [1]. We therefore emphasize this transform here, but we also experiment with the other transform. We also remark that the spectral-domain wavelets of the former transform allow flexible choices of wavelet functions. We note that a conventional generative scattering network [9] is not as competitive as state-of-the-art results based on GAN and VAE for image generation tasks. This is probably due to the complexity of some Euclidean-type datasets and their nontrivial high frequency components. Nevertheless, graph-type datasets have a discrete nature, and they often do not exhibit high frequency components. Therefore, the graph scattering transform might be competitive and efficient for specific generative tasks in the graph domain.

We consider three types of graph generation tasks:

- *Link prediction*: In this task, one is interested in predicting whether two vertices from the same graph are connected. The common input includes a graph with missing edges and features of vertices. The goal is to decide whether an edge exists between any pair of vertices. This can be viewed as generating a graph from a latent representation of the partially available graph.
  A well-known application of this task is the prediction of citations. Common citation datasets were collected by [12]

and further pre-processed by [13]. These datasets of publications and citations contain features for each publication as well as citation linkage, which are modeled by an undirected graph. In the pre-processed data, one only partially knows the citation linkage and the task is to recover the citations for all pairs of publications.

- *Signal generation on graphs*: In this task, the graph is fixed and the set of vertices and edges is known. The goal is to generate signals on the given graph. Although we are unaware of convincing data of this type, we believe that this is a potentially useful task. Among the three tasks we review here, it is most similar to generation tasks in Euclidean domains. We can thus enforce some graph structure in special Euclidean-type or grid-type datasets. Here we pursue this idea with the Fashion-MNIST dataset of images of clothing items [14]. We do not consider the domain of a 28×28 pixel image as Euclidean, but associate to it a graph, where each pixel is connected by an edge with its nearby neighbors. Each image is then a signal on this graph. One then needs graph-based methods for generating these signals.

- *Graph and signal generation*: In this task, one needs to generate both the graph structure and the signals on the graph. An interesting application is the design of chemical molecules. A network learns from a given dataset, such as QM9 [15], to generate both the atoms (signals on vertices) and the bonds (edges) from a latent sample. This can be used as a purely machine learning-based approach for the design of new drugs [16].

Our proposed method is easy to implement. Furthermore, the adjustment of the structure of the decoder to the three types of tasks does not require a lot of effort. Unlike GAN or VAE, the model in this paper only requires training the generator. Meanwhile, there is flexibility in designing the graph wavelets and in choosing the decoder structure. We believe it is a highly adaptable method for various graph tasks and indeed our numerical experiments demonstrate competitive results.

II. BACKGROUND

We overview previous relevant works as follows: §II-A reviews generative scattering networks, §II-B reviews graph convolutional networks, and §II-C reviews some recent graph generative models.

A. Scattering networks for generative models

The generative scattering network [9] can be considered as an encoder-decoder system in which one only needs to train the decoder. The feature extraction part of the encoder is a scattering transform [17], [18] with fixed parameters. It provides multi-scale signal representation, which is Lipschitz continuous with respect to small deformations. The next part of the encoder aims to map the transformed signals into samples of a Gaussian latent variable. We refer to this step as Gaussianization. We later describe in §III two Gaussianization methods.
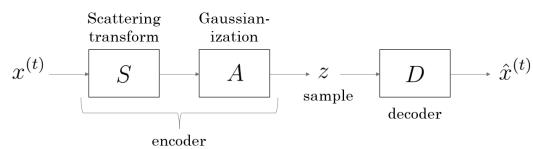

Fig. 1: The structure of a generative scattering network.

The decoder D can be taken to be a multi-layer perceptron (MLP) and is trained by minimizing the reconstruction loss. Fig. 1 shows the structure of a generative scattering model. In order to generate samples, initial samples are generated from the latent Gaussian variable and then the decoder is applied to them resulting in the final samples.

B. Graph convolutional networks

Convolution is a key contributor for the recent success of deep learning. In the Euclidean domain, convolutional networks are helpful in learning multi-scale representations. The same idea was introduced to graphs by exploiting the spectral graph representation, that is, the spectral decomposition of the graph weight matrix or graph Laplacian [3], [19]. A common proposal for a graph convolution uses pointwise multiplication of the graph Fourier-transformed signals, where the graph Fourier transform uses the basis of the spectral decomposition of the Graph Laplacian in place of the discrete Fourier basis. One can apply nonlinear functions such as the ReLU for each graph vertex. A variety of good approximations to the spectral approach are able to speed up the spectral decomposition process and maintain accuracy [4], [5].

A special type of graph convolutional network (GCN) is the graph scattering network [1], which does not require training and was proved to be approximately invariant to permutations and stable to sufficiently small signal or graph manipulations. A graph scattering network uses graph wavelets [10] defined on the eigenspace of the graph Laplacian to construct multi-layer models. Alternatively, [2] construct graph scattering transforms using an earlier graph wavelet transform [11]. In general, the graph wavelets of [10] are more flexible as one can choose different kinds of wavelets on the spectral domain according to different tasks. While the graph wavelets of [11] are not flexible, they use the normalized graph Laplacian and the corresponding diffusion map and metric, which might be natural for particular applications.

C. Graph generative networks

Several recent papers address graph generation networks with either a GAN or a VAE structure, where specific designs are often needed for different applications.

Recent graph generative models with a GAN structure include NetGAN [20], GraphGAN [21] and MolGAN [22]. NetGAN aims to generate graphs with properties, such as max degree and triangle count, that are similar to training samples. GraphGAN generates distribution of edges in order to solve node classification and recommendation problems whose tasks are very similar to that of the link prediction problem.

MolGAN is designed for molecule generation and generates signals with respect to both vertices and edges.

Recent graph generative models with a VAE structure include VGAE [13], GraphVAE [23] and JT-VAE [24]. VGAE aims to solve the link prediction problem by completing the adjacency matrix. GraphVAE is designed for molecule generation. It generates the adjacency matrix as well as the vertex and edge features. JT-VAE is specifically designed for generating complex molecules while enforcing validity.

III. GRAPH GENERATIVE SCATTERING NETWORK

Existing graph generation networks require training either GAN or VAE, which is a difficult task. Furthermore, the design of a graph generation network is often complex [20], [22], [24] and its hyperparameter selection might be difficult. In view of these obstacles, we propose here the graph generative scattering network. It is composed of two components: an encoder and a decoder. The encoder is a graph scattering network [1], which is followed by a Gaussianization step. It produces a latent Gaussianized representation for the graph signal. It is also used to form a latent "Gaussian distribution." The parameters of the graph scattering network are predetermined, unlike the parameters of a generative auto-encoder which are learned. The decoder is trained by using the Gaussianized latent representation of the data and minimizing a loss function chosen according to the corresponding task. In all of the graph-specific tasks we mentioned in §I, the decoder can be taken to be a network with fully-connected layers, whose structure is determined by the specific task. Generation is obtained by applying the trained decoder to the latent Gaussian distribution. More details on forming the encoder and decoder are provided in §III-A and §III-B, respectively.

A. Details of the Encoder

In order to fully understand the formation of the encoder, we review the graph scattering network of [1] and explain how to form a Gaussian distribution from its output. We consider a graph $G = (V, E)$ with $|V| = N$ vertices. A signal in $L^2(V; \mathbb{R}^K)$ can be regarded as a matrix $\boldsymbol{X} \in \mathbb{R}^{N \times K}$. The scattering transform can be regarded as a function that acts on the columns of $\boldsymbol{X}$. Let $\boldsymbol{L} \in \mathbb{R}^{N \times N}$ be the unnormalized graph Laplacian $\boldsymbol{L} = \boldsymbol{D} - \boldsymbol{W}$, where $\boldsymbol{D}$ is the diagonal matrix of degrees and $\boldsymbol{W}$ is the weight (adjacency) matrix whose (n, m)-th entry is the weight of the edge connecting vertices v_n and v_m. Its spectral decomposition is $\boldsymbol{L} = \sum_{l=0}^{N-1} \lambda_l \boldsymbol{u}_l \boldsymbol{u}_l^*$ with $0 = \lambda_0 \leq \cdots \leq \lambda_{N-1}$. We assume a limiting scale, $J \in \mathbb{Z}$, and scaling and wavelet functions, ϕ and ψ, respectively. Hammond et al. [10] define the graph wavelet transform of $\boldsymbol{f} \in \mathbb{R}^N$ as $(\boldsymbol{f} * \boldsymbol{\phi}_{-J}, \boldsymbol{f} * \boldsymbol{\psi}_{-J+1}, \boldsymbol{f} * \boldsymbol{\psi}_{-J+2}, \ldots)$, where

$$\boldsymbol{f} * \boldsymbol{\psi}_j = \sum_{l=0}^{N-1} \boldsymbol{u}_l \boldsymbol{u}_l^* \boldsymbol{f} \hat{\psi}(2^{-j} \lambda_l), \text{ for } j > -J ;$$

$$\boldsymbol{f} * \boldsymbol{\phi}_{-J} = \sum_{l=0}^{N-1} \boldsymbol{u}_l \boldsymbol{u}_l^* \boldsymbol{f} \hat{\phi}(2^J \lambda_l) .$$

For any m no larger than the number of layers, a path $p = (j_1, \cdots, j_m)$ is a vector of m scales of the graph wavelets, which satisfy $j_1, \cdots, j_m > -J$. In practice, we also set 0 as the finest scale and require that $-J < j_1, \cdots, j_m \leq 0$. The scattering propagator with respect to a path p is obtained by consecutive application of convolutions with wavelets of these scales and absolute values, which serve as nonlinearities, in the following way

$$\boldsymbol{U}[p]\boldsymbol{f} = \left| \left| \left| \boldsymbol{f} * \boldsymbol{\psi}_{j_1} \right| * \boldsymbol{\psi}_{j_2} \right| * \cdots * \boldsymbol{\psi}_{j_m} \right| .$$

The scattering transform with respect to the path p is obtained by $\boldsymbol{S}[p]\boldsymbol{f} = \boldsymbol{U}[p]\boldsymbol{f} * \boldsymbol{\phi}_{-J}$.

Let $\mathcal{P}$ denote the collection of all paths of length no larger than the number of layers. The scattering transform of $\boldsymbol{f}$ with respect to $\mathcal{P}$ is

$$\boldsymbol{S}[\mathcal{P}]\boldsymbol{f} = (\boldsymbol{S}[p]\boldsymbol{f})_{p \in \mathcal{P}}.$$

A simple illustration of the scattering transform is provided in Fig. 1 of [1]. Note that the scattering transform depends on the underlying graph. For simplicity, we exclude this dependence from our notation.

For the matrix signal $\boldsymbol{X} = [\boldsymbol{X}_1 | \cdots | \boldsymbol{X}_K] \in \mathbb{R}^{N \times K}$, the scattering transform is

$$\boldsymbol{S}[\mathcal{P}]\boldsymbol{X} = (\boldsymbol{S}[\mathcal{P}]\boldsymbol{X}_k)_{k=1}^K .$$

We remark that if the set $\mathcal{P}$ has L elements, then defining $M = LK$, $\boldsymbol{S}[\mathcal{P}](\boldsymbol{X})$ can be represented as a matrix in $\mathbb{R}^{N \times M}$ or a vector in $\mathbb{R}^{N \cdot M}$. We denote the latter vector by $\bar{\boldsymbol{X}}$.

Zou and Lerman [1] establish various theoretical properties of the scattering transform $\boldsymbol{S}[\mathcal{P}]$. In particular, they show that it preserves the energy—that is, the l_2 norm—of each signal. The usefulness of this property for graph generation will be discussed later. Another important property they establish is the robustness of the scattering transform to small perturbations of the signal and the graph [1, §5]. This robustness implies that similar signals and graphs are encoded as similar latent codes.

The last step of the encoder maps the transformed data points so that one may possibly generate them with a Gaussian distribution, as required by the generator. We earlier referred to this mapping as Gaussianization. We describe two possible mappings that were previously suggested for related tasks. Following [9], one such mapping is whitening. Specifically, let $\{\bar{\boldsymbol{X}}^{(t)}\}_{t=1}^T$ be the scattering transform vectors of the input samples and let $\bar{\boldsymbol{\mathcal{X}}}$ be the representing matrix of $\{\bar{\boldsymbol{X}}^{(t)}\}_{t=1}^T$. That is, $\bar{\boldsymbol{\mathcal{X}}} = (\bar{\boldsymbol{X}}^{(t)})_{t=1}^T \in \mathbb{R}^{T \times NM}$. As advocated in [9], a dimension reduction by PCA can be further applied to $\{\bar{\boldsymbol{X}}^{(t)}\}_{t=1}^T$. Next, using the following mean and sample covariance of the scattering transform vectors

$$\boldsymbol{\mu} = \frac{1}{T} \sum_{t=1}^T \bar{\boldsymbol{X}}^{(t)} \text{ and } \boldsymbol{\Sigma} = \frac{1}{T} \sum_{t=1}^T (\bar{\boldsymbol{X}}^{(t)} - \boldsymbol{\mu})(\bar{\boldsymbol{X}}^{(t)} - \boldsymbol{\mu})^* ,$$

the whitening map $\boldsymbol{A}$ is

$$\boldsymbol{A}\bar{\boldsymbol{\mathcal{X}}} = \boldsymbol{\Sigma}^{-1/2}(\bar{\boldsymbol{\mathcal{X}}} - \boldsymbol{\mu}) . \tag{1}$$

The whitened samples are uncorrelated and the hope in [9] is that their distribution will be close to that of a normal distribution with identity covariance.

The output of the encoder with a whitening transformation is guaranteed to have similar robustness results to signal and graph manipulations as the ones established in [1] for the graph scattering transform with additional factor of $\|\mathbf{\Sigma}^{-1/2}\|$, where $\|\cdot\|$ denotes the spectral norm of the corresponding matrix. We remark that due to the initial dimension reduction that removed small eigenvalues of the sample covariance of the transformed data points, $\|\mathbf{\Sigma}^{-1/2}\|$, is not expected to be large.

One problem with the whitening process is that the distribution of its output may not be close to Gaussian. While traditional Gaussianization methods [25], [26] can improve the distribution of the output, their encoding may not be robust to signal and graph manipulation.

An alternative to whitening is a spherization procedure inspired by [27]. That is, the data points $\{\bar{\mathbf{X}}^{(t)}\}_{t=1}^T$ are normalized by their Euclidean norm to lie on the unit sphere. The hope is that these mapped data points can be generated from a Gaussian with their sample mean and covariance. In general, this may not be the case, though [27] have a heuristic and incomplete argument for this property in a different setting. This argument can be made precise by using the Gaussian Annulus Theorem (see e.g., [28, Theorem 2.9]).

Due to the energy preservation property of the full graph scattering transform of [1], one can instead spherize the input data points and the result of the encoder will be the same. Therefore, in theory, the encoder is robust to signal and graph manipulation with respect to the original data after spherization. Due to this property and similarly to [27], we do not initialize the spherization by centering with the sample mean. In practice, since scattering is only applied to a finite number of layers, the energy is contracted. Therefore, it is better to normalize after the scattering and this is what we do in our experiments. The final stage of the encoder with spherization calculates the sample mean $\boldsymbol{\mu}_s$ and covariance $\boldsymbol{\sigma}_s$ for the spherized scattering output and fits a Gaussian $\mathcal{N}(\boldsymbol{\mu}_s, \boldsymbol{\sigma}_s)$, which is used as the latent distribution for sampling.

We denote the original samples by $\{\mathbf{X}^{(t)}\}_{t=1}^T$ and the output of the encoder (with either whitening or spherization) by $\mathbf{\Phi}[\mathcal{P}](\mathbf{X}^{(t)})$, or in short $\mathbf{\Phi}(\mathbf{X}^{(t)})$ for $t = 1, \cdots, T$. We note that the mapping $\mathbf{\Phi}$ can also be applied to any signal $\mathbf{X} \in \mathbb{R}^{N \times K}$, where $K \geq 1$. We denote the feature vector corresponding to $\mathbf{X}$ by $\mathbf{Z} = \mathbf{\Phi}[\mathcal{P}](\mathbf{X}) \in \mathbb{R}^{N \times M}$. We will refer to $\mathbf{Z}$ as a latent code.

B. Details of the decoder

Recall that the decoder is a network with fully-connected layers. We describe its architecture according to the following three different tasks.

1) Link prediction: For link prediction, we encode the features of the partially available graph into a latent vector, and use the same vector to generate the full graph via the learned decoder. Note that in this task only one fixed graph is given, and thus no Gaussianization procedure is applied
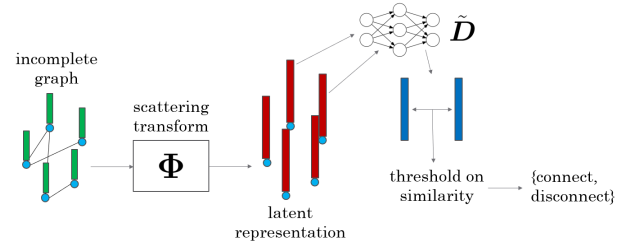

Fig. 2: Sketch of a graph scattering network for link prediction.
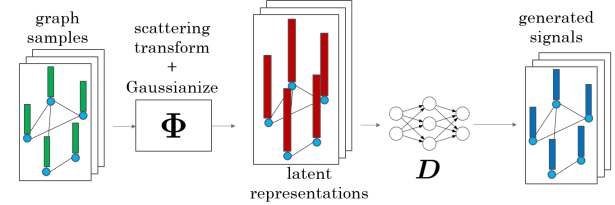

Fig. 3: Sketch of a graph scattering network for signal generation on graphs.

in the encoder. That is, the linear transformation $\mathbf{A}$ in (1) is the identity. The input includes a weight matrix $\mathbf{W}_{\text{train}}$, which contains weights for the partially available edges, and a matrix signal $\mathbf{X} \in \mathbb{R}^{N \times K}$. The encoder is a scattering network $\mathbf{\Phi}$ that maps $\mathbf{X}$ and $\mathbf{W}_{\text{train}} \in \mathbb{R}^{N \times N}$ into a latent representation $\mathbf{Z} \in \mathbb{R}^{N \times M}$. As in [13], the decoder is a simple network $\mathbf{D}$ such that $\mathbf{D}(\mathbf{Z}) = \sigma(\tilde{\mathbf{D}}(\mathbf{Z})\tilde{\mathbf{D}}(\mathbf{Z})^T)$, where σ is the softmax function and $\tilde{\mathbf{D}}$ is an MLP. The network $\mathbf{D}$, whose parameters are those of $\tilde{\mathbf{D}}$, is trained to minimize the cross-entropy loss function

$$L(\mathbf{D}) = \sum_{i,j:\mathbf{W}(i,j)\neq 0} \left[-\log \mathbf{D}(\mathbf{\Phi}(\mathbf{X},\mathbf{W}))(i,j)\right] . \quad (2)$$

The structure of the entire network is illustrated in Fig. 2.

2) Signal generation on graphs: For signal generation on graphs, which we also refer to as graph signal generation, one is given a fixed graph domain and different signals on the nodes of this graph and the goal is to generate similar signals. An input random variable $\mathbf{X} \in \mathbb{R}^{N \times K}$ is first mapped by the scattering transform to $\mathbf{S}[\mathcal{P}](\mathbf{X})$ and then to a Gaussian $\mathbf{Z} = \mathbf{\Phi}[\mathcal{P}](\mathbf{X}) \in \mathbb{R}^{N \times M}$. The decoder $\mathbf{D}$ is taken to be an MLP that maps $\mathbf{Z}$ to a matrix $\mathbf{D}(\mathbf{Z})$ in $\mathbb{R}^{N \times K}$. The parameters of $\mathbf{D}$ are obtained by minimizing the reconstruction loss function

$$L(\mathbf{D}) = T^{-1} \sum_{t=1}^{T} \left\| \mathbf{X}^{(t)} - \mathbf{D}(\mathbf{\Phi}(\mathbf{X}^{(t)})) \right\| , \quad (3)$$

where $\{\mathbf{X}^{(t)}\}_{t=1}^T$ are the training data points. The structure of the generative network is the same as in Fig. 1, where in the current case S is the graph scattering transform. Fig. 3 illustrates the generation procedure.

3) Graph and signal generation: The scattering transform can be used as an encoder for generating both the graph and the signal on it. We train two MLP's $\mathbf{D}_1$ and $\tilde{\mathbf{D}}_2$, where both take the Gaussian random variable $\mathbf{Z} = \mathbf{\Phi}(\mathbf{X})$ as input. The
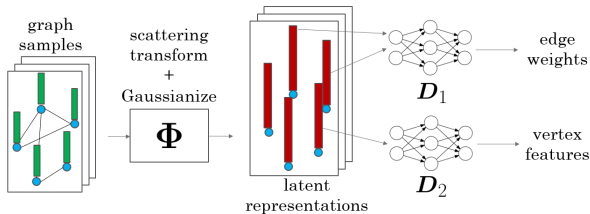
Fig. 4: Sketch of the graph scattering network for graph and signal generation.

network $\boldsymbol{D}_1(\boldsymbol{Z}) = \sigma(\tilde{\boldsymbol{D}}_1(\boldsymbol{Z})\tilde{\boldsymbol{D}}_1(\boldsymbol{Z})^T)$ is used to generate the weight matrix $\boldsymbol{W}$ and the network $\boldsymbol{D}_2$ is used to generate the graph signal $\boldsymbol{X}$. They are trained at the same time, with the reconstruction loss function

$$L(\boldsymbol{D}_1, \boldsymbol{D}_2) = T^{-1} \sum_{t=1}^{T} \Big[\left\| \boldsymbol{W}^{(t)} - \boldsymbol{D}_1(\boldsymbol{\Phi}(\boldsymbol{X}^{(t)})) \right\| + \left\| \boldsymbol{X}^{(t)} - \boldsymbol{D}_2(\boldsymbol{\Phi}(\boldsymbol{X}^{(t)})) \right\| \Big], \tag{4}$$

where $\{(\boldsymbol{X}^{(t)}, \boldsymbol{W}^{(t)})\}_{t=1}^{T}$ are the training data points. The norms can be replaced with cross-entropy losses if one wants the outputs to be categorical, as in the molecule generation task. Fig. 4 illustrates this generation procedure.

IV. EXPERIMENTS

We test our proposed method, whose code is available at https://github.com/dmzou/SCAT, and compare it with other available algorithms using datasets addressing the three different tasks reviewed in §I.

Even though we advocate using the graph scattering network of [1] (due to its robustness to signal manipulation), we also tested our generation algorithm with the graph scattering network of [2]. When addressing link prediction (so a Gaussianization procedure is not applied), we denote by SCAT-S and SCAT-D our proposed generative procedure with the spectral and diffusion networks of [1] and [2] respectively. For the other two applications, we distinguish between the two Gaussianization procedures: whitening and spherization. We use "W" to denote whitening and "N" to denote normalization, i.e., spherization. We use, as above, "S" to denote the spectral graph scattering transform in [1] and "D" to denote the diffusion graph scattering transform in [2]. We denote the four resulting networks by SCAT-SW, SCAT-DW, SCAT-SN and SCAT-DN.

Following [1] we use in all experiments the simple Shannon wavelet and the limiting scale $J = 3$ for SCAT-S, SCAT-SW and SCAT-SN. We similarly use $J = 3$ and choose $t = 3$ (t is the power of multiplying diffusion operator) for SCAT-D, SCAT-DW and SCAT-DN. For link prediction, we take 2-layer scattering as it performs better on the validation set; for the other two tasks, we take 3-layer scattering.

Our method requires moderate memory and GPU and we easily tested it on a PC with 8GB RAM and GTX1060 GPU. Comparison with VGAE on the Pubmed dataset demanded

TABLE I: Characteristics of three citation datasets

Dataset	Vertices	Edges	Classes	Features
Cora	2708	5429	7	1433
Citeseer	3327	4732	6	3703
Pubmed	19717	44338	3	500

more advanced GPU. We thus report results of all experiments on a Linux machine with 64GB RAM and GTX1080Ti GPU.

A. Link prediction for citation data

We predict links for the three citation datasets of [12]: Cora, Citeseer and Pubmed. Each dataset contains information about publications in certain fields and the corresponding citation linkage between these publications. This information can be embedded in a graph in which the publications and citations are represented by vertices and edges, respectively. Although the citation link is directed, we follow the common convention of assuming an undirected and unweighted graph. That is, if any of two papers cite the other, an edge is drawn between them. Table I lists some characteristics of the three datasets.

We use the preprocessing step suggested in [13] and documented in https://github.com/tkipf/gae for all tests. It divides the original edges into 85% training, 5% validating and 10% testing sets.

For SCAT-S and SCAT-D, the dimension of the output of the scattering transform is reduced to 128. The decoder $\tilde{D}$ in both models is taken to be a single dense layer of size 512, activated by the ReLU function. In order to minimize (2), we use the Adam optimizer [29] with a learning rate of 0.001, where we train 1,000 epochs for each run. We record the following two common scores: area under curve (AUC) and average precision (AP). Similarly to [13], we take 10 runs for each setting and record the average and standard deviation. Table II reports our results for SCAT-S and SCAT-D, together with results for GAE and VGAE [13]. We note that SCAT-S improves over the previous results for all the three datasets, while SCAT-D achieves comparable results with GAE and VGAE. The averaged time for the scattering transform of SCAT-S and SCAT-D, applied to Cora/Citeseer/Pubmed, is 2.82s/6.43s/492.90s and 1.77s/4.31s/346.43s respectively. SCAT-S requires more time due to the spectral decomposition, whose average time for the 3 datasets is 1.24s/2.16s/317.43s. Table III reports the training time for each epoch. This table implies that SCAT-S and SCAT-D are much more efficient. This is because only parameters in the decoder need to be updated. Combining the times of the scattering transforms reported above, which are executed once, and the training times for each epoch in Table III, multiplied by the number of epochs, we conclude that the total times of SCAT-S and SCAT-D are more efficient.

B. Signal generation on graphs

We use the Fashion-MNIST dataset [14] for a sanity check of SCAT for the problem of graph signal generation. Any element of this dataset is a 28×28 grayscale pixel image and can be considered as a graph in the following way: the

TABLE II: Results for link prediction using the citation datasets. We report the mean and standard deviation over 10 runs for each setting. All models for the same dataset are trained based on the same training and validating links.

Dataset	Cora		Citeseer		Pubmed	
	AUC (%)	AP (%)	AUC (%)	AP (%)	AUC (%)	AP (%)
SCAT-S	94.48 ± 0.15	94.63 ± 0.17	97.27 ± 0.12	97.57 ± 0.12	97.52 ± 0.03	97.19 ± 0.04
SCAT-D	92.08 ± 0.09	93.05 ± 0.11	92.54 ± 0.14	94.16 ± 0.12	92.73 ± 0.17	93.56 ± 0.09
GAE	91.34 ± 0.52	92.62 ± 0.38	92.37 ± 0.67	93.72 ± 0.58	96.35 ± 0.18	96.53 ± 0.16
VGAE	91.14 ± 0.40	92.16 ± 0.29	92.70 ± 0.76	93.93 ± 0.57	95.68 ± 0.35	95.92 ± 0.32

TABLE III: Time for training an epoch for the citation data

Dataset	SCAT-S	SCAT-D	GAE	VGAE
Cora	8.1ms	8.1ms	209.1ms	206.4ms
Citeseer	8.1ms	8.1ms	298.6ms	302.3ms
Pubmed	64.9ms	64.9ms	7832.6ms	7889.2ms

pixels are the graph vertices, and nearby pixels are connected by graph edges. The edges and weights are formed as in [1]. That is, each pixel represents a vertex and it is connected with its four nearest neighbors with weight e^{-1} and its four nearest diagonal neighbors with weight e^{-2}.

The encoder of this network is a graph scattering transform. Its output dimension is reduced to 256 from $28 \times 28 \times 13 = 10{,}192$. The decoder is an MLP of two hidden layers of size 512. In order to minimize (3), we use the Adam optimizer with a learning rate of 0.001, where we train 2,000 epochs for each run. To avoid mode collapse, we have restricted the dataset to the "boots" category, which contains 5,454 training examples. Sample images from this category are demonstrated in Fig. 6a.

To generate an "image" (or graph signal), we take a sample $z \in \mathbb{R}^N$ from $\mathcal{N}(0, I)$ for SCAT-SW and SCAT-DW, and from $\mathcal{N}(\boldsymbol{\mu}_s, \boldsymbol{\Sigma}_s)$ for SCAT-SN and SCAT-DN, and report the output of the decoder. Figs. 6b-6e illustrate samples generated by these networks, while using the graph associated with a 28×28 pixel image. We report the ℓ_1 reconstruction loss defined in (3) in Table IV. Note that the averaged ℓ_1-norm of the training images is 174.56. We also report the times for scattering and training a single epoch.

Our experiments indicate that training models with spherization converges faster than with whitening. To see this, we plot the reconstruction loss as a function of epoch for the first 500 epochs for the four methods we tested in Fig. 5. It is clear that SCAT-SN and SCAT-DN converge much faster in the first 300 epochs.

The images generated in Figs. 6b-6e are of similar quality for all the scattering methods. Note that some high frequency components, that is, details, of these images are missing. This

TABLE IV: Reconstruction loss and time for scattering transform (for the complete training data) and training an epoch for the Fashion-MNIST dataset

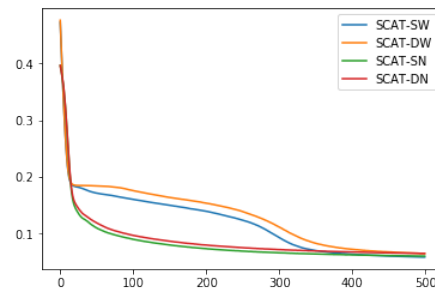|	SCAT-SW	SCAT-DW	SCAT-SN	SCAT-DN
Reconstr. loss	0.0482	0.0532	0.0513	0.0548
Time (scattering)	1,128.6ms	1,484.3ms	1,148.3ms	1,566.0ms
Time (epoch)	39.4ms	39.1ms	30.7ms	30.6ms

Fig. 5: Reconstruction loss with respect to epoch for training SCAT models for the Fashion-MNIST data. No difference is noted after 500 epochs.

phenomena is is common with the Euclidean generative scattering network of [9]. Nevertheless, common graph-type data often do not have high-frequency components. For instance, in the molecular data reviewed in §IV-C, the signals on each vertex just take five different values.

For comparison with other methods, Figs. 6f and 6g illustrate samples generated from graph generative models that combine VAE/GAN with the graph convolutional layers proposed in [5]. Specifically, for what we call VAE-GCN, we construct a graph-based VAE in which the encoder consists of two graph convolutional layers while the decoder is an MLP of two hidden layers. The latent mean and variance both have dimension 256 and the hidden layers of the MLP have dimension 512. For what we call GAN-GCN, we replace the discriminator of a vanilla GAN with two graph convolutional layers and use an MLP with two hidden layers of dimension 512 for the generator.

All three methods are trained using Adam with a learning rate of 0.001. The latent noise for both GAN-GCN and GAN-FCN is of dimension 256. The training time for each epoch for VAE-GCN and GAN-GCN is 3.947s and 1.761s respectively, which is much slower than training SCAT models since parameters from both the generator and the discriminator need to be updated for GAN-GCN, and parameters from both the encoder and the decoder need to be updated for VAE-GCN. Observing Figs. 6f and 6g, we see that samples from VAE-GCN look blurrier than those of SCAT models and also miss high-frequency information. Samples from GAN-GCN are very noisy and still miss high-frequency information and suggest that the model suffers from a severe mode collapse.
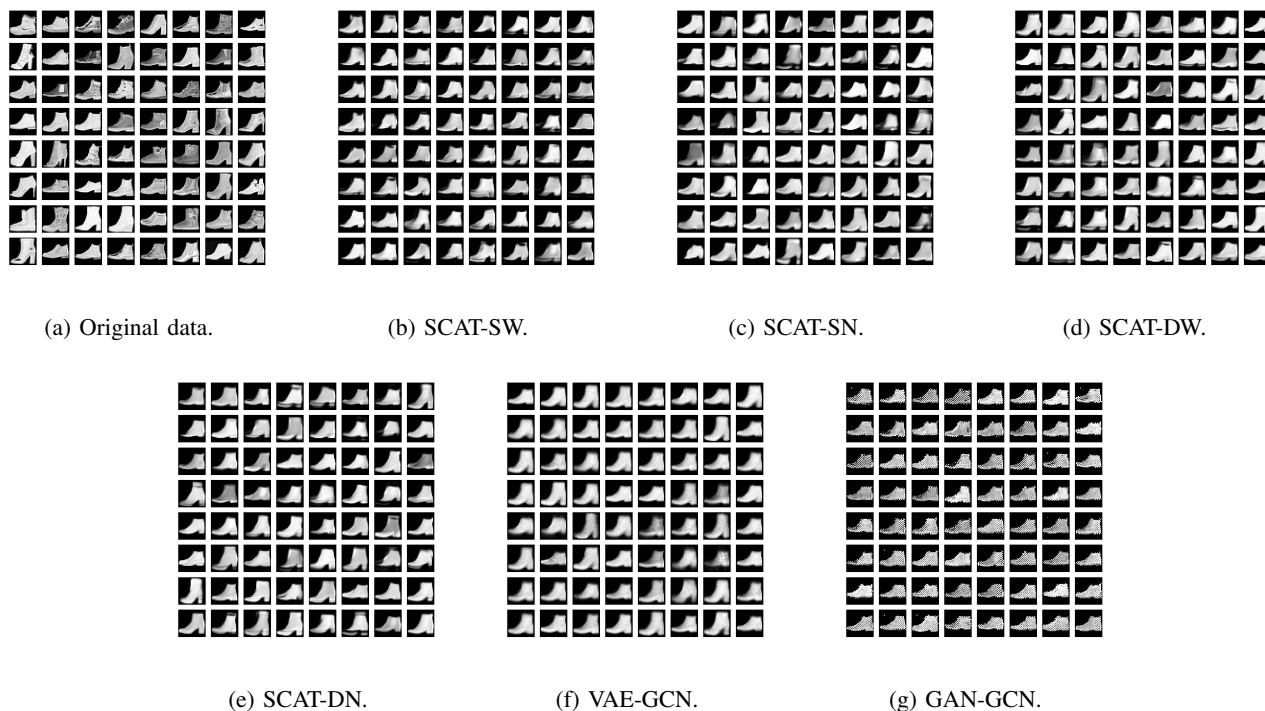
(a) Original data.　　(b) SCAT-SW.　　(c) SCAT-SN.　　(d) SCAT-DW.

(e) SCAT-DN.　　(f) VAE-GCN.　　(g) GAN-GCN.

Fig. 6: Original and generated images for boots of Fashion-MNIST data.

C. Graph generation for molecular data

We test graph and signal generation using the molecular dataset QM9 [15]. This dataset contains 134k molecules made of the following atoms: C, H, O, N, and F. There are two common ways of embedding these kinds of datasets into an interpretable feature space. Kusner et al. [30] treat molecules as "words" by looking at their simplified molecular-input line-entry (SMILE) strings. Graphs are also commonly used to represent molecules, where the graph vertices represent the atoms composing the molecule and the graph edges are the bonds. The vertex signals assign atom types to the vertices. While there are five type of atoms, H is automatically determined by the other atoms and the given chemical bonds. Therefore, only labels of the four heavy atoms (C, O, N or F) need to be assigned. Since the dataset has molecules with at most 9 heavy atoms, we assume graphs with 9 vertices and assign a dummy value for vertices without assignment of a heavy atom. Each heavy atom and the dummy one are assigned a one-hot vector, i.e., a unit coordinate vector in $\mathbb{R}^5$, and each atom is represented by one of these five one-hot vectors.

For graph generation, there is no unique benchmark for checking the quality of the generated graphs. Bojchevski et al. [20] proposed to use graph properties such as the max degree and the number of triangles for graph generation. However, it is often hard to compare these graph properties and there is no motivation for using them for molecule generation. Samanta et al. [31] proposed to check validity (whether a sample is a valid chemical molecule), uniqueness (whether a sample is unique among all generated samples) and novelty

TABLE V: Time for the scattering transform (for the complete training data) and training an epoch for the QM9 dataset

	SCAT-SW	SCAT-DW	SCAT-SN	SCAT-DN
Time (scattering)	137.12s	94.75s	132.01s	95.95s
Time (epoch)	3.68s	3.66s	3.71s	3.69s

(whether a sample is different from any sample in the training data). We use these measures since they are more quantitative and experiments on QM9 by [22] and [23] also report them. They are checked after converting the graphs into SMILE strings using the RDKit package (https://www.rdkit.org/).

The full QM9 dataset is used for training. This choice is the same as that in [22, §5.3]. It is different than [23], in which a small set of molecules is used for training and only molecules with 9 heavy atoms are considered.

As explained in §III-B3, for this application of graph generation, the SCAT decoders for both vertices and edges are MLP's. In our experiments, both of them have three hidden layers of dimension 128, 256, 512, respectively. We take the encoder to be a graph scattering transform, with output dimension reduced to 135 (15 for each vertex). In order to minimize (4), we use the Adam optimizer with a learning rate of 0.001, where we train 300 epochs. Table V reports the computational times of scattering and training on this dataset.

Using SCAT, we generate 10k molecules and record the validity, uniqueness, and novelty in Table VI. For comparison, we also record results reported for GraphVAE in [23] and our test of MolGAN [22] based on the codes available at https://github.com/nicola-decao/MolGAN. For MolGAN, we

TABLE VI: Comparison of graph generation by GraphVAE, MolGAN and SCAT using the QM9 dataset. Values are reported in percentages according to 10k generated samples. Since the GraphVAE code is unavailable, results are copied from the paper and marked with parenthesis.

Algorithm	Valid	Unique	Novel
GraphVAE	(55.7)	(76.0)	(61.6)
GraphVAE (imp)	(56.2)	(42.0)	(75.8)
GraphVAE (no GM)	(81.0)	(24.1)	(61.0)
MolGAN (no RL)	90.4	31.1	97.8
MolGAN (RL Valid)	100.0	0.3	13.6
MolGAN (RL Unique)	99.2	37.1	64.5
MolGAN (RL Novel)	98.5	0.6	100.0
SCAT-SW	65.4	92.7	86.9
SCAT-DW	38.0	98.1	94.2
SCAT-SN	64.9	92.0	85.7
SCAT-DN	47.4	98.3	92.0

perfrom four different tests. The first one (no RL) applies MolGAN without reinforcement learning (RL). The next three (RL Valid, RL Unique and RL Novel) apply RL for validity, uniqueness and novelty, respectively. An RL step is done after each five epochs. For each setting we only report the best result among six choices of the parameter λ (0.01, 0.05, 0.1, 0.25, 0.5, 0.75) and three choices of the dropout rate (0, 0.1, 0.25) used in [22]. Without RL, it takes approximately 11 seconds to train an epoch; with RL for validity or uniqueness, it takes 15-18s to train an epoch; with RL for novelty, it takes 93-122s to train an epoch.

All the SCAT models achieve high scores in uniqueness and novelty. Note that SCAT-DW and SCAT-DN achieve slightly higher uniqueness and novelty than SCAT-SW and SCAT-SN, but they have notably lower score in validity. We remark that it is possible to achieve high validity with the scattering generative models. We were able to achieve 93.9/17.6/98.6 for valid/unique/novel scores if we train SCAT-SW using an MLP with a single hidden layer with dimension 5 for the vertices and 32 for the edges. However, a simpler model leads to severe mode collapse, as implied by the low uniqueness score. We observed that this simple model assigns "carbon" to a lot of vertices, and it has two effects: first, it is easier to construct valid molecules[2]. Second, it causes mode collapse as there is not much variety of molecular decompositions.

We show samples of molecules generated by SCAT-SW with a decoder with three hidden layers in Fig. 7a and samples generated by SCAT-SW with a decoder with one hidden layer in Fig. 7b. It can be seen that many molecules in Fig. 7b are composed only of carbon (and hydrogen); in contrast, there are more oxygen and nitrogen in the samples in Fig. 7a. We believe that in order to explore new design of drugs, it is more important to generate molecules with more variety. Therefore, the results from three-layer decoders are more meaningful.

As a comparison, the three measures reported for GraphVAE are moderate. MolGAN has good validity and novelty scores,

but low uniqueness score, which indicates a mode collapse [22]. This mode collapse is even more severe when either validity or novelty is used for RL, although the model achieves perfect score for validity or novelty, respectively. We remark that MolGAN uses three hidden layers of the same dimensions and we thus believe that the mode collapse is not due to a low-complex structure of the generator.

V. CONCLUSION

We proposed the graph generative scattering network as a generative model for graphs and graph signals. The network applies a prescribed encoder which does not require training and is robust to signal perturbation and graph deformations. Numerical experiments show competitive results for the tasks of link prediction in citation data and molecule generation. Although scattering usually takes time, it is still more efficient to train scattering-based models due to the smaller number of parameters to update in training. We believe the graph generative scattering network has the potential to be used in a wider range of applications on graphs.

We experimented with two possible choices for the scattering transform and two choices for Gaussianization. Overall, the scattering described in [1] achieves better performance than the scattering described in [2] for link prediction and molecule generation. We remark that for molecule generation the scattering of [2] achieves slightly better scores of uniqueness and novelty, however, its validity score is significantly worse. There is no significant difference in the results for the two methods of Gaussianization. However, the spherization-based methods (SCAT-SN and SCAT-DN) converge much faster for graph signal generation.

We used the Fashion-MNIST dataset as a sanity check for the graph signal generation, since we are unaware of a more convincing application for this specific task. In this application, we do not expect graph-based methods to compete with general methods, because graphs only retain partial spatial relationships. Indeed, the resolution of the generated images is not as good as that of the original images. Nevertheless, since the results in the similar discrete tasks of link prediction and molecule generation are competitive, we believe that SCAT also bears promise for graph signal generation when the signals are of low resolution.

ACKNOWLEDGMENT

REFERENCES

[1] D. Zou and G. Lerman, "Graph convolutional neural networks via scattering," *arXiv preprint arXiv:1804.00099*, 2018.
[2] F. Gama, A. Ribeiro, and J. Bruna, "Diffusion scattering transforms on graphs," *arXiv preprint arXiv:1806.08829*, 2018.
[3] M. Henaff, J. Bruna, and Y. LeCun, "Deep convolutional networks on graph-structured data," *arXiv preprint arXiv:1506.05163*, 2015.
[4] M. Defferrard, X. Bresson, and P. Vandergheynst, "Convolutional neural networks on graphs with fast localized spectral filtering," in *Advances in Neural Information Processing Systems*, pp. 3844–3852, 2016.

---

[2]A carbon vertex can have degree at most four while keeping the molecule valid. The largest possible degree is only three for a nitrogen vertex and two for an oxygen vertex.

paper N-20350.pdf

(a) Samples from a generator with three hidden layers.   (b) Samples from a generator with one hidden layer.
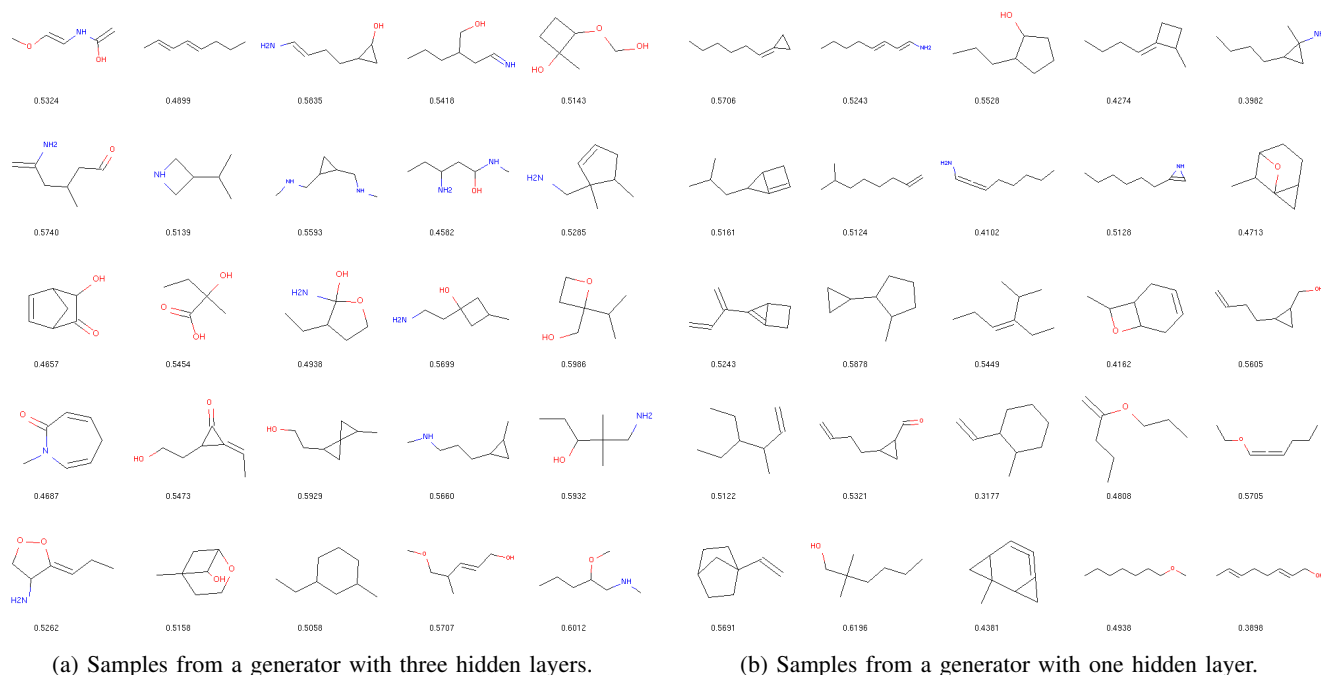
Fig. 7: Samples of generated molecules via SCAT-SW. The number below each molecule is the Quantitative Estimate of Drug-likeness (QED) score [32], which is automatically generated for the figures by RDKit. We are not trying to optimize it.

[5] T. N. Kipf and M. Welling, "Semi-supervised classification with graph convolutional networks," in *International Conference on Learning Representations*, 2017.

[6] Z. Chen, X. Li, and J. Bruna, "Supervised community detection with hierarchical graph neural networks," *arXiv preprint arXiv:1705.08415*, 2017.

[7] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative adversarial nets," in *Advances in neural information processing systems*, pp. 2672–2680, 2014.

[8] D. P. Kingma and M. Welling, "Auto-encoding variational bayes," in *International Conference on Learning Representations*, 2014.

[9] T. Angles and S. Mallat, "Generative networks as inverse problems with scattering transforms," in *International Conference on Learning Representations*, 2018.

[10] D. K. Hammond, P. Vandergheynst, and R. Gribonval, "Wavelets on graphs via spectral graph theory," *Applied and Computational Harmonic Analysis*, vol. 30, no. 2, pp. 129–150, 2011.

[11] R. R. Coifman and M. Maggioni, "Diffusion wavelets," *Applied and Computational Harmonic Analysis*, vol. 21, no. 1, pp. 53–94, 2006.

[12] P. Sen, G. Namata, M. Bilgic, L. Getoor, B. Gallligher, and T. Eliassi-Rad, "Collective classification in network data," *AI magazine*, vol. 29, no. 3, p. 93, 2008.

[13] T. N. Kipf and M. Welling, "Variational graph auto-encoders," in *NIPS Workshop on Bayesian Deep Learning (NIPS-16 BDL)*, 2016.

[14] H. Xiao, K. Rasul, and R. Vollgraf, "Fashion-MNIST: a novel image dataset for benchmarking machine learning algorithms," *arXiv preprint arXiv:1708.07747*, 2017.

[15] R. Ramakrishnan, P. O. Dral, M. Rupp, and O. A. Von Lilienfeld, "Quantum chemistry structures and properties of 134 kilo molecules," *Scientific data*, vol. 1, p. 140022, 2014.

[16] M. Olivecrona, T. Blaschke, O. Engkvist, and H. Chen, "Molecular de-novo design through deep reinforcement learning," *Journal of cheminformatics*, vol. 9, no. 1, p. 48, 2017.

[17] S. Mallat, "Group invariant scattering," *Communications on Pure and Applied Mathematics*, vol. 65, no. 10, pp. 1331–1398, 2012.

[18] J. Bruna and S. Mallat, "Invariant scattering convolution networks," *IEEE transactions on pattern analysis and machine intelligence*, vol. 35, no. 8, pp. 1872–1886, 2013.

[19] J. Bruna, W. Zaremba, A. Szlam, and Y. LeCun, "Spectral networks and locally connected networks on graphs," in *International Conference on Learning Representations*, 2014.

[20] A. Bojchevski, O. Shchur, D. Zügner, and S. Günnemann, "NetGAN: Generating graphs via random walks," in *Proceedings of the 35th International Conference on Machine Learning*, pp. 610–619, 2018.

[21] H. Wang, J. Wang, J. Wang, M. Zhao, W. Zhang, F. Zhang, X. Xie, and M. Guo, "GraphGAN: Graph representation learning with generative adversarial nets," in *AAAI Conference on Artificial Intelligence*, 2018.

[22] N. De Cao and T. Kipf, "MolGAN: An implicit generative model for small molecular graphs," *arXiv preprint arXiv:1805.11973*, 2018.

[23] M. Simonovsky and N. Komodakis, "GraphVAE: Towards generation of small graphs using variational autoencoders," *arXiv preprint arXiv:1802.03480*, 2018.

[24] W. Jin, R. Barzilay, and T. Jaakkola, "Junction tree variational autoencoder for molecular graph generation," in *Proceedings of the 35th International Conference on Machine Learning*, pp. 2323–2332, 2018.

[25] S. S. Chen and R. A. Gopinath, "Gaussianization," in *Advances in neural information processing systems*, pp. 423–429, 2001.

[26] V. Laparra, G. Camps-Valls, and J. Malo, "Iterative gaussianization: from ica to random rotations," *IEEE transactions on neural networks*, vol. 22, no. 4, pp. 537–549, 2011.

[27] P. Bojanowski, A. Joulin, D. Lopez-Pas, and A. Szlam, "Optimizing the latent space of generative networks," in *Proceedings of the 35th International Conference on Machine Learning*, pp. 600–609, 2018.

[28] A. Blum, J. Hopcroft, and R. Kannan, *Foundations of Data Science*. June 2017.

[29] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.

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

[31] B. Samanta, A. De, N. Ganguly, and M. Gomez-Rodriguez, "Designing random graph models using variational autoencoders with applications to chemical design," *arXiv preprint arXiv:1802.05283*, 2018.

[32] G. R. Bickerton, G. V. Paolini, J. Besnard, S. Muresan, and A. L. Hopkins, "Quantifying the chemical beauty of drugs," *Nature chemistry*, vol. 4, no. 2, p. 90, 2012.

paper N-20350.pdf