Deep Graph Translation

Xiaojie Guo, Lingfei Wu, *Member, IEEE*, and Liang Zhao, *Senior Member, IEEE*

***Abstract*— Deep generative models for graphs have recently achieved great successes in modeling and generating graphs for studying networks in biology, engineering, and social sciences. However, they are typically unconditioned generative models that have no control over the target graphs given a source graph. In this article, we propose a novel graph-translation-generative-adversarial-nets (GT-GAN) model that transforms the source graphs into their target output graphs. GT-GAN consists of a graph translator equipped with innovative graph convolution and deconvolution layers to learn the translation mapping considering both global and local features. A new conditional graph discriminator is proposed to classify the target graphs by conditioning on source graphs while training. Extensive experiments on multiple synthetic and real-world datasets in the domain of cybernetworks, the Internet of Things, and neuroscience demonstrate that the proposed GT-GAN model significantly outperforms other baseline methods in terms of both effectiveness and scalability. For instance, GT-GAN outperforms the classical state-of-the-art (SOTA) methods in functional connectivity (FC) prediction of brain networks by at least 32.5%.**

***Index Terms*— Deep graph generation, deep graph translation (DGT), generative adversarial networks, graph neural network (GNN).**

I. Introduction

IN RECENT years, deep learning on graphs (DLG) has seen a surge of interest, especially for graph representation and recognition tasks, such as node-level classification [1]–[5] and graph-level classification [6]–[8]. Because of the successes in graph neural networks (GNNs), researchers have recently started to explore the use of deep generative models for graph synthesis on practical applications, such as designing new chemical molecular structures [9], [10]. This has led to many of the recent advances in deep graph generative models, some of which are domain-dependent models [11], [12] for generating graphs with physical constraints, while others consider the generation of generic graphs [13]–[15].

In many applications, it is crucial to guide the graph generation process by conditioning on an input graph, which can be cast as a graph translation learning problem—translating the graph in the source domain to the graph in the target domain. Such graph translation can be highly desirable for applications such as molecule optimization and rare event forecasting, where rare and abnormal graph patterns (e.g., traffic congestion and terrorism events) can be inferred prior to their occurrence even without historical data on the abnormal patterns for this specific graph (e.g., a road network or a human contact network). For example, in social networks where nodes represent people and edges represent their contacts, their contacting graphs vary dramatically across different circumstances. For instance, when people are organizing a riot, their contact graph is expected to become denser and have several special "hubs" (e.g., key players). Therefore, it is highly beneficial to accurately predict contact graphs in target circumstances regarding situational awareness and resource allocation.

This type of new problem is formulated as *deep graph translation* (DGT), which aims at translating the graph in the source domain into the distribution of corresponding output graphs in the target domain based on deep GNNs [16]. This problem is analogical to image-to-image translation in image processing [17] and language translation in natural language processing (NLP) [18]. Unfortunately, similar to the consensus that the existing image or text generation methods are not appropriate to be directly applied to graph generation [9], existing image or text translation methods cannot be directly applied to DGT problems.

Few works have been proposed in the domain of graph translation, however, with their own limitations. Some works propose to only handle some specific tasks without generality, such as molecule reaction prediction or molecule optimization in the domain of biology [19], [20]. Others utilize the sequential generating technique that has difficulty in preserving the global probability of graphs due to the lack of long-term dependency in the sequence [21]. Thus, there are still critical challenges that hurdle the further scientific exploration of the DGT domain.

1) *Difficulty in Jointly Learning Both Local and Global Information for Translation:* One needs to learn the translation mapping not only in the local information (i.e., neighborhood pattern of each node) but also in the global property of the whole graph (e.g., node degree distribution or graph density).
2) *Difficulty in Transferring the Extracted Patterns of Multiple Levels From the Input Graph Into the Generated Target Graph:* In the DGT process, multiple levels' graph information or latent features can be learned from the input graph and have different contributions

Manuscript received December 1, 2020; revised November 18, 2021; accepted January 12, 2022. This work was supported in part by the National Science Foundation (NSF) under Grant 1755850, Grant 1841520, Grant 2007716, Grant 2007976, Grant 1942594, and Grant 1907805, a Jeffress Memorial Trust Award, Amazon Research Award, NVIDIA GPU Grant; in part by the Design Knowledge Company under Contract 10827.002.120.04; and in part by CIFellowship (2021CIF-Emory-05). *(Corresponding author: Liang Zhao.)*

Xiaojie Guo and Lingfei Wu are with JD.COM Silicon Valley Research Center, Mountain View, CA 94043 USA (e-mail: xguo7@gmu.edu; lwu@email.wm.edu).

Liang Zhao is with Emory University, Atlanta, GA 22043 USA (e-mail: liang.zhao@emory.edu).

Color versions of one or more figures in this article are available at https://doi.org/10.1109/TNNLS.2022.3144670.

Digital Object Identifier 10.1109/TNNLS.2022.3144670

on generating the target graph. How to utilize this hierarchical information in generating the target graph is a problem.

To address the aforementioned challenges, we present novel neural network architecture: graph-translation-generative-adversarial-nets (GT-GAN). We first propose a conditional graph GAN architecture that consists of an encoder–decoder translator and a conditional graph discriminator (CGD) to learn the conditional distribution for graph translation. In addition, a novel graph U-net translator with graph skips is proposed to guarantee the model to learn multiple levels' information in the graph encoder and select valuable ones to generate the target graphs in the graph decoder. To corporate with the U-net structure, a set of mirrored graph convolution and deconvolution layers is designed, including both the edge and the node convolution and deconvolution. Furthermore, the proposed graph convolution and deconvolution layers are able to jointly embed the local and global information. Finally, GT-GAN is scalable with at most quadratic computation and memory consumption in terms of the number of nodes in a graph, making it suitable for at least modest-scale graphs. We highlight our main contributions as follows.

1) We develop a generic framework, which consists of a novel graph translator and CGD for learning a conditional distribution of target graphs given the input graphs.
2) We propose a novel graph encoder consisting of "edge convolution" layers and "node convolution" layers that can embed both local and global information from multiple levels.
3) We propose a novel graph decoder consisting of the "edge deconvolution" and "node deconvolution" layers, which hierarchically decodes the node representations first into the edge representations and then generates the final target graph. The graph skip-connection is also utilized to map the learned multiple-levels' information to the target graphs.
4) We conduct extensive experiments on both synthetic and real-world datasets on five performance metrics to demonstrate the effectiveness and efficiency of the proposed model.

II. Related Works

A. Graph Neural Networks (GNNs)

The recent surge of research into GNN can be generally divided into two categories: graph recurrent networks and graph convolutional networks. Graph recurrent networks originate from early work [22], [23] and are based on recursive neural networks that have been extended by modern deep learning techniques, such as gated recurrent units [1]. The other category, graph convolutional networks, originates from spectral graph convolutional neural networks (GCNs) [24], which were extended by using fast localized convolutions [25] and further approximated by an efficient architecture for a semisupervised setting [2]. Self-attention mechanism and subgraph-level information were also explored later to further improve the representation power of learned node embeddings [3], [26]–[28]. GNNs are mostly utilized for first learning the latent representation of the nodes or graphs and then apply the learned representation to many downstream tasks, such as node classification, graph classification, and link prediction.

B. Graph Generative Models

Most of the existing graph generation methods for general graphs have been proposed in the last two years and are based on variational autoencoders (VAE) [9], [14], generative adversarial nets (GANs) [29], and others [10], [13]. The current deep graph generation methods can be categorized into two main branches.

1) *Sequential Generating [10], [13]:* This generates the nodes and edges in a sequential way, one after another. Sequential generating performs the local decisions made in the preceding one in an efficient way with a time complexity of only $O(N)$, but it has difficulty in preserving the long-term dependency. Thus, some global properties (e.g., scale-free property) of the graph are hard to include.
2) *One-Shot Generating [9], [14]:* This refers to building a probabilistic graph model based on the matrix representation that can generate all nodes and edges in one shot. One-shot generating methods have the capacity of modeling the global property of a graph by generating and refining the whole graph (i.e., nodes and edges) synchronously through several iterations, but most of them are limited to small graphs (i.e., the size of node set is less than 20) since the time complexity is not less than $O(N^4)$. In this article, we adopt the one-shot generation process considering that the global information and the local information are both critical to be captured in the graph translation task. Furthermore, our one-shot generation process is validated to enjoy less complexity [i.e., $O(N^2)$] compared to the existing one-shot generation methods.

C. Graph-Based Translation Methods

A variety of graph-to-sequence models [30] have been proposed to cope with different tasks, including machine translation [31], [32], semantic parsing [33]–[35], question generation [36], and health status prediction [37], [38]. The sequence-to-graph algorithms are generally popular with those working on NLP methods, including generating Abstract Meaning Representation (AMR) structures [39] and dependency graphs [40], [41]. A few very recent attempts have been made to develop graph-to-graph translation models. Jin *et al.* [19] proposed a domain-specific graph translation model to deal with the molecular optimization task through junction-tree-VAE. Do *et al.* [20] dealt with the chemical reaction product prediction problem by predicting the products based on the input graph of reactants. Sun and Li [21] proposed a recurrent neural network (RNN)-based model for encoding and decoding the directed acyclic graph (converted from regular graphs), which can be viewed as a contemporary work to our work. Guo *et al.* [42] proposed the node edge co-disentanglement (NEC)-DGT model to deal with the transformation between multiattributed graphs, which, however, is trained for prediction tasks instead of learning a distribution. The above

existing methods are either domain-specific methods [15], [20] or cannot handle the graph translation within a fixed set of nodes, which is the most desirable task in many real-world applications, which is the most desirable task in many real-world applications.

III. Overall Architecture of GT-GAN

In this section, we first formulate the DGT problem. We then propose the GT-GAN model for graph translation and discuss each component in detail in Sections III-B–III-D.

A. Problem Formulation for Deep Graph Translation

Our goal is to learn an end-to-end translation mapping from an *source graph* to a *target graph*. Let a source graph $G_X = (\mathcal{V}, A, S)$ such that $\mathcal{V}$ is the set of N nodes and $A \in \mathbb{R}^{N\times N}$ is an adjacency matrix (binary or weighted), where G_X can be weighted or unweighted, directed or undirected. Let $S \in \mathbb{R}^{N\times F}$ be a node feature matrix with each row representing a node feature vector S_i. $A_{i,j} \in A$ denotes the corresponding weight of the edge between node v_i and v_j. Similarly, we define a *target graph* $G_Y = (\mathcal{V}', \mathcal{E}', A', S')$ that shares the same node sets and node features with G_X but with different topology and connection weights. Formally, *graph translation* is learning a translator from a source graph $G_X \in \mathcal{G}_X$ with a random noise U to generate a target graph $G_Y \in \mathcal{G}_Y$, where $\mathcal{G}_X$ and $\mathcal{G}_Y$ denote the domains of the source and target graphs, respectively. The translation mapping is denoted as $\mathcal{T} : U, G_X \to G_Y$.

It is worth noting that our aim is to learn a conditional distribution of the target graphs given a source graph, which is cast as a conditional graph generation problem instead of a predict task. The source graph can be mapped into many diverse target graphs that may have different topologies yet follow the same distribution.

1) Proposed GT-GAN Framework: Fig. 1 shows our proposed generic GAN framework for graph translation that consists of a graph translator $\mathcal{T}$ and a CGD $\mathcal{D}$. In this figure, we assume that the node feature has only one dimension for simplicity. Since our task is to train a conditional generator with "one-to-many mapping" instead of a deterministic one, the noise U is introduced by the dropout function [43] in each convolution and deconvolution[1] layer, as shown (in green lines) in Fig. 1. Our graph translator $\mathcal{T}$ is trained to produce target graphs that cannot be distinguished from "real" ones by our CGD $\mathcal{D}$. Specifically, the generated target graph $G_{Y'} = \mathcal{T}(G_X, U)$ cannot be distinguished from the real one, G_Y, based on the current source graph G_X. $\mathcal{T}$ and $\mathcal{D}$ undergo an adversarial training process based on source and target graphs by solving the following loss function:

$$\mathcal{L}(\mathcal{T}, \mathcal{D}) = \mathbb{E}_{G_X,G_Y}[\log \mathcal{D}(G_Y|G_X)] + \mathbb{E}_{G_X,U}[\log(1 - \mathcal{D}(\mathcal{T}(G_X, U)|G_X))] \quad (1)$$

where $\mathcal{T}$ tries to minimize this objective, while an adversarial $\mathcal{D}$ tries to maximize it, i.e., $\mathcal{T}^* = \arg\min_{\mathcal{T}} \max_{\mathcal{D}} \mathcal{L}(\mathcal{T}, \mathcal{D})$.

[1]"Deconvolution," here, is actually "transposed convolution" instead of a mathematical operation that reverses the effect of convolution.

We mix the GAN loss with the L1 loss to enforce sparsity similarity, which is also found to be useful in the image translation problem [17]

$$\mathcal{L}_{l1}(\mathcal{T}) = \mathbb{E}_{A,A',U}\left[\left\|A' - T(G_X, U)\right\|_1\right] \quad (2)$$

where $T(G_X, U)$ refers to the adjacent matrix of the generated graph. The training process is a tradeoff between $\mathcal{L}_{l1}$ and $\mathcal{L}(\mathcal{T}, \mathcal{D})$, which jointly enforces $\mathcal{T}(G_X, U)$ and G_Y to follow a similar, but not necessarily identical, topological pattern. Specifically, $\mathcal{L}_{l1}$ makes $\mathcal{T}(G_X, U)$ share the same rough outline of sparsity pattern as G_Y, while $\mathcal{L}(\mathcal{T}, \mathcal{D})$ allows $\mathcal{T}(G_X, U)$ to vary to some degree. Thus, the optimal objective $\mathcal{T}^*$ of the translator, which generates graphs that are as "real" as possible, is defined as

$$\mathcal{T}^* = \arg\min_{\mathcal{T}} \max_{\mathcal{D}} \mathcal{L}(\mathcal{T}, \mathcal{D}) + \lambda \mathcal{L}_{l1}(\mathcal{T}). \quad (3)$$

The graph translator $\mathcal{T}$ is an encoder–decoder architecture, where we propose a new graph encoder to obtain the node representations of the source graph and propose the graph deconvolution with skips to generate the target graph, as shown in Fig. 1, which we elaborate on in the following sections.

B. Graph Encoder

The graph encoder aims to learn the representations of nodes based on the node features and graph topology of the source graph. One of the crucial challenges is to learn both local and global information in graph embedding. For instance, when learning translation between two scale-free graphs, one needs to translate both the local information (i.e., neighborhood patterns) and the scale-free property (i.e., node degree distributions of the whole graph) from a source graph to a target graph.

Thus, we first propose "edge convolution" layers to learn the edge representations for all pairs of nodes. The aim is to explore a group of hidden relations from the topology of the source graph, which may include both the n-hop connection relations and those that can deliver structural similarity among nodes. Then, the "node convolution" layer is used to embed each node's representation by aggregating all the other nodes' information based on the learned edge representations. In this way, each node is finally affected by not only its neighboring nodes (local information) but also all the other nodes in the graph (global information), by varying degrees. Fig. 2 illustrates the details of these matrix operations involving graph convolution.

1) Edge Convolution: In each "edge convolution" layer, each edge's hidden representations are computed by its adjacent edges from the last layer. In the directed graph, each edge has a source node and a target node. Thus, there is a pair of learnable parametric vectors ϕ and ψ as convolution filters for two directions to convolute the adjacent edges or edge representations of each edge, as shown in Fig. 2(a). The edge representation $E^{l+1}_{i,j}$ of the $(l+1)$th layer is learned by all the outgoing edge representations of node v_i and all the incoming edge representations of node v_j

$$E^{l+1}_{i,j} = \sigma\left(\sum_{k_1=1}^{N} E^{l}_{i,k_1}\phi^{l}_{k_1}\right) + \sigma\left(\sum_{k_2=1}^{N} E^{l}_{k_2,j}\psi^{l}_{k_2}\right) \quad (4)$$

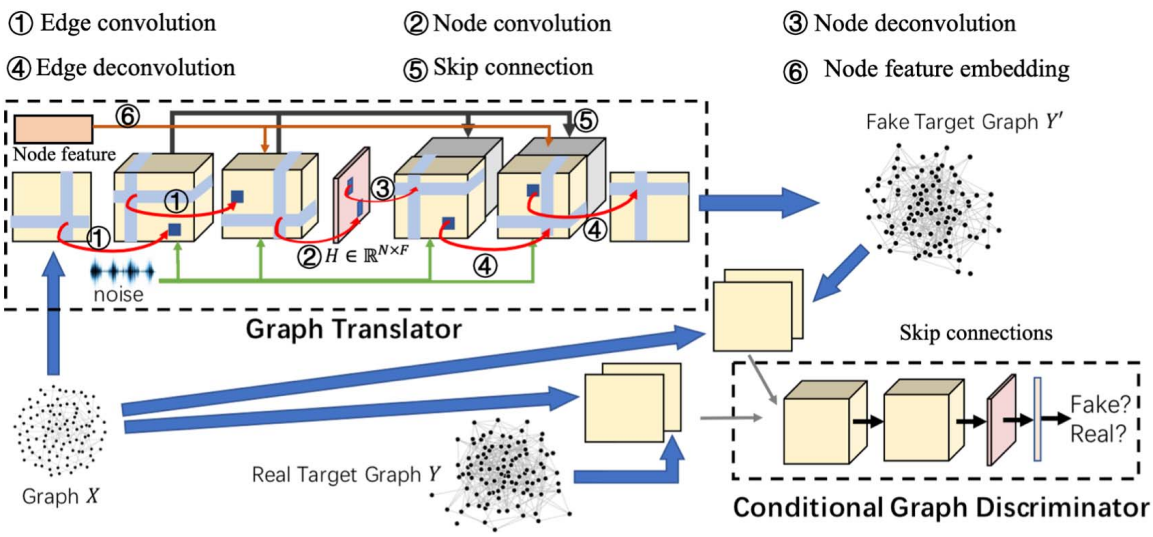


Fig. 1. GT-GANs consisting of a graph translator and a CGD. Novel graph encoder and decoder are designed for the graph translation problem.

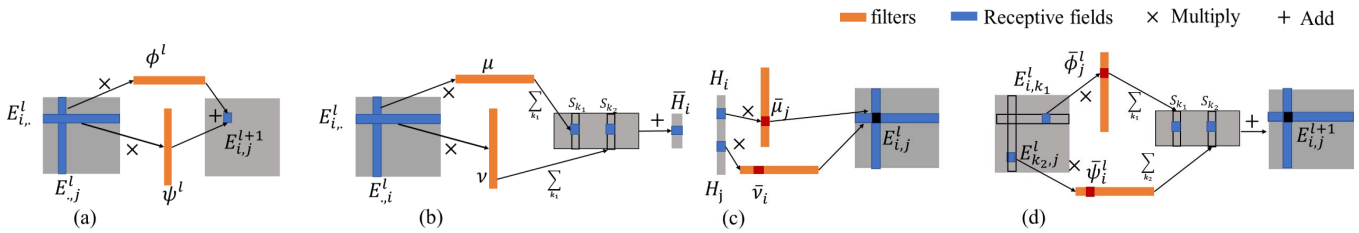


Fig. 2. Matrix operations for graph convolution and graph deconvolution. In convolution operations, we need to utilize row filter to convolute "incoming" edges and column filter for "outgoing" edges. However, in deconvolution operations, we have to utilize the transposed filters, namely, the column filter to decode for "incoming" edges and row filter to decode for "outgoing" edges. (a) Edge convolution. (b) Node convolution. (c) Node deconvolution. (d) Edge deconvolution.

where $E^0_{i,j} \equiv A_{i,j}$, $\phi^l \in \mathbb{R}^{N\times 1}$ refers to the filter vector to be learned, and $\phi^l_{k_1}$ refers to the k_1th entry in ϕ^l that is related to node v_{k_1}. The edge convolution here does not involve the node features since it aims to only encode the structure information.

2) Node Convolution: After learning the various edge representations through several edge convolution layers, one "node convolution" layer is used to learn each node's representations by aggregating all the other nodes based on the learn edge representations and their node features. There are two vectors of the covolution filter μ and ν for two directions, as shown by the rectangles in orange in Fig. 2(b). Then, the node representation $\bar{H}_i \in \mathbb{R}^{1\times F}$ for node v_i is computed as

$$\bar{H}_i = \sigma\left(\sum_{k_1=1}^{N} E^{l+1}_{i,k_1}\mu_{k_1}S_{k_1}\right) + \sigma\left(\sum_{k_2=1}^{N} E^{l+1}_{k_2,i}\nu_{k_2}S_{k_2}\right) \quad (5)$$

where $\bar{H}_i \in \mathbb{R}^{1\times F}$ and $\mu, \nu \in \mathbb{R}^{N\times 1}$ refers to the filter vectors for the two directions to be learned, and μ_{k_1} refers to the element of μ that is related to node v_{k_1}. $\bar{H}_i$ is then flattened and transformed into a node representation vector $H_i \in \mathbb{R}^{1\times C}$ by a fully connected layer. C is the length of the node representation. It is worth noting that node features are involved in the node convolution since it aims to encode both node and structure information into node embedding.

C. Graph Decoder

The decoder aims to generate the edges of the target graph by taking the extracted latent information from the source graph. To model the complex dependency inside the target graph and transfer the information of multiple levels that are learned from multiple layers in the encoder, we propose a graph U-Net[2] consisting of graph skips and dedicated graph deconvolution layers. The graph deconvolution decodes the single node (or edge) information to yield its incoming and outgoing adjacent edges as a mirrored graph convolution process. In addition, several skips are implemented to map the learned information of each layer in the encoder to mirror the corresponding layers in the decoder. The proposed graph deconvolution technique incorporates both "node deconvolution" and "edge deconvolution" layers.

1) Node Deconvolution: First, one "node deconvolution" layer is used to generate the edge representations of the target graph mentioned above based on the learned latent node representations. As shown in Fig. 2(c), "node deconvolution" is a reversed process of "node" convolution. It is assumed that each node can influence its potential edges to other nodes.

[2]Gao *et al.* [44] proposed a graph embedding method with the similar name "graph U-Net," while it uses the input graph topology all the way from the pooling part to the unpooling part without new graph generation.

Suppose that the node deconvolution is at the $(l-1)$th layer; then, its output, namely, the edge representations $E^l_{i,j}$ between node v_i and node v_j, can be computed as follows:

$$E^l_{i,j} = \sum_{m=1}^{C} \left(\sigma\left(H^m_i \bar{\mu}_j\right) + \sigma\left(H^m_j \bar{v}_i\right)\right) \tag{6}$$

where $\sigma(H^m_i \bar{\mu}_j)$ means the deconvolution contribution of node v_i to its edge representations with node v_j, which is made by the mth entry of its node representations, and $\bar{\mu}_j$ represents one entry of the deconvolution filter vector $\bar{\mu} \in \mathbb{R}^{N\times 1}$ that is related to node v_j. Here, the node features are not involved in the node deconvolution because the direct inputs of the decoder are already node embedding, which contains the node feature information.

2) Edge Deconvolution: We can now recursively apply our proposed "edge deconvolution" layer to decode all the latent edge representations from the upper layer back to those of the lower layer. As a reversed way of doing "edge" convolution, the edge representation for each pair of nodes in the lth layer can make a contribution to generating its adjacent edges' representations in the $(l+1)$th layer, as shown in Fig. 2(d). Thus, the relation $E^l_{i,j}$ between node v_i and node v_j in the $(l+1)$th layer is computed as follows:

$$E^{l+1}_{i,j} = \sigma\left(\bar{\phi}^l_j \sum_{k_1=1}^{N} E^l_{i,k_1} S_{k_1}\right) + \sigma\left(\bar{\psi}^l_i \sum_{k_2=1}^{N} E^l_{k_2,j} S_{k_1}\right) \tag{7}$$

where $\bar{\phi}^l_j \sum_{k_1=1}^{N} E^l_{i,k_1} S_{k_1}$ can be interpreted as the decoded contribution of node v_i to its edge representations with node v_j, and $\bar{\phi}^l_j$ refers to the element of deconvolution filter vector that is related to node v_j. The output of the last "edge" deconvolution layer denotes the probability of the existence of an edge in the target graph. All the symbols σ refers to the activation functions. Node features are used in the edge convolution since each edge generation in the target graphs will be influenced by both the adjacent edges and related nodes of the source graphs.

It is worth noting that both edge and node convolution layers can easily add more pairs of filter vectors for enhancing the capacity of the encoder to learn more latent patterns. They can also be adapted to undirected graphs, where only one direction's convolution filters are needed. To enlarge the power of the convolution and deconvolution filters to capture more complex patterns, the above-learned filter parameters are customized to each node and edge and, thus, not invariant to different node permutations. However, it can be easily adapted to be invariant to node permutation by making the elements vectors (e.g., μ, ν, ϕ, and ψ) the same. It is suggested to select the appropriate version based on the real-world cases to make full use of the proposed method."

3) Skips for Graph Deconvolution: Based on the graph deconvolution above, it is possible to utilize skips to link the extracted edge latent representations of each layer in the graph encoder with those in the graph decoder. Specifically, the output of the lth "edge deconvolution" layer in the decoder is concatenated with the output of the lth "edge convolution" layer in the encoder to form joint two channels of feature maps, which are then input into the $(l+1)$th deconvolution layer.

It is worth noting that one key factor for effective translation is the design of a symmetrical encoder–decoder pair, which allows skip-connections to directly translate different level's edge information in the format of edge representation at each layer. In addition, this edge representation can be utilized for the following node embedding. The existing edge-related GNNs cannot meet both requirements. There are two main categories: 1) one aims to do the node representation learning, which considers the edge information when aggregating and updating the node embedding, such as GCN [2] and graph attention network (GAT) [3]; however, they cannot directly generate the edge latent embeddings/representation and 2) one category aims to do the edge representation learning for link prediction based on two nodes' hidden representations from GNNs as input [45]. Thus, they cannot utilize the edge latent representation for node representation learning.

D. Conditional Graph Discriminator

The graph discriminator must distinguish between the "translated" target graph and the "real" ones based on the source graphs, as this helps to train the generator in an adversarial way. Technically, this requires the discriminator to accept two graphs simultaneously as inputs (a real target graph and a source graph or a generated graph and a source graph) and classify the two graphs as either related or not. Thus, we propose a CGD that leverages the same graph convolution layers in the encoder for the graph classification, as shown in Fig. 1. Specifically, the source and target graphs are both ingested by the CGD and stacked into an $N \times N \times 2$ tensor, which can be considered a two-channel input. After obtaining the node representations, the graph-level embedding is computed by summing these node embeddings. Finally, a softmax layer is implemented to distinguish the input graph pair from the real graph or generated graph.

E. Computational Complexity Analysis

The graph encoder and decoder share the same time complexity. Without loss of generality, we assume the number of nodes in the graph as N. The total complexity of GT-GAN (i.e., the dense graph) is now $O(N^2)$ for all the "edge convolutions," "node convolutions," and fully connected layers in CGD. Similarly, the total memory consumption for GT-GAN is also $O(N^2)$. In practice, many source graphs are likely to be sparse; thus, using sparse matrix-vector operations [10] can further reduce the computational and memory costs of the first layer to $O(N)$. Compared to the existing works [9], [14] that can only scale to the small size of graphs (up to $|V| = 20$) and often have $O(N^3)$ or even $O(N^4)$ computational costs, our GT-GAN is able to provide a scalable [i.e., $O(N^2)$] algorithm for general graphs.

IV. Experiments

This section reports the results of extensive experiments and the ablation studies that are carried out to test the performance

of GT-GAN on two synthetic and two real-world datasets. All experiments were conducted on a 64-bit machine with an NVIDIA GPU (GTX 1070, 1683 MHz, and 8-GB graphics double data rate (GDDR)5). The code and data utilized are available at https://github.com/anonymous1025/Deep-Graph-Translation-.

A. Datasets

1) Synthetic Datasets: Synthetic datasets contain a group of pairs of scale-free graphs. Each source graph is first generated as a directed scale-free network, whose degree distribution follows the power-law property [46]. To generate the target graph, a node will be selected as the target node with a probability proportional to its in-degree, which will be linked to a new source node with the probability of p_1. Similarly, a node will be selected as a source node with probability proportional to its out-degree, which will be linked to a new target node with a probability of p_2.[3] Thus, both the source and target graphs are scale-free graphs. Each group has five subsets with different graph sizes (number of nodes): 10, 20, 50, 100, and 150. Each subset consists of 5000 source-target graph pairs: 2500 pairs were used for training and the remaining 2500 for testing.

2) User Authentication Dataset: This dataset includes the authentication activities of 97 users on their accessible computers and servers in an enterprise computer network [47]. Each user account generates a log file recording the computer accessing history, which could be formulated as a directed weighted graph called an authentication graph, where nodes represent computers and the directed edges' weights represent authentication activities with certain frequencies. The goal of this application was to synthesize users' future malicious authentication graphs given the normal one. There are 78 pairs of graphs (malicious and normal behavior) of graph size 50 and 315 pairs of graphs of graph size 300 from 97 users as two training sets.

3) Internet-of-Things (IoT) Dataset: This application focused on IoT network malware confinement prediction, namely, predicting optimal network operation given a compromised one [42]. There are three subsets of graph pairs with different sizes (20, 40, and 60), where the nodes represent devices and node attribute is a binary value referring to whether the device is compromised or not, and the edges represent the connections of two devices, the edge attributes are continuous values reflecting the distances of two devices. The real target graphs are generated by the classical malware confinement methods: stochastic controlling with malware detection. There are 334 pairs of source (compromised IoT) and target graphs (optimal IoT) in each subset.

4) Real-World HCP Dataset: Brain network prediction, such as the prediction of functional connectivity (FC) based on structural connectivity (SC), is a very critical task in neuroscience. The goal is to learn the mapping from the resting-state FC into task-specific FC in the human brain. In this dataset, the source and the target graphs, respectively, reflect the SC and the FC of the same subject's brain network. In particular, both types of connectivity are processed from the magnetic resonance imaging (MRI) data obtained from the healthcare provider data (HCP) dataset [48]. The node attributes refer to the index of each node by a one-hot vector. FC data are collected regarding five different human brain states, namely, resting, emotion, gambling, language, and motoring. Thus, there are five subsets for both full normalized and partial regularized datasets. In total, each subset has 823 pairs of SC and FC samples.

[3]In our experiment, p_1 and p_2 are chosen as 0.41 and 0.54, which are default values of scale-free graph generator in a python package named NetworkX.

B. Comparison Methods

We compare our GT-GAN against four state-of-the-art (SOTA) graph generation methods: GraphRNN [10], GraphVAE [9], GraphGMG [13], and RandomVAE [14]. The baseline model called S-Generator is the part of our full model GT-GAN that essentially is a graph translator without discriminator. We propose this S-Generator model in order to evaluate the necessity of building the proposed GT-GAN in a generative instead of a regression. Due to the deficiency of the existing graph translation model, all the comparison methods are for graph generation methods and, thus, were trained on the target graphs without conditioning on the source graphs due to the models' inherent capability limitations. The datasets were assigned to each comparison model for the experiment based on their scalability in terms of graph size.

C. Evaluation Results on Synthetic Datasets

1) Statistics-Based Evaluation: To evaluate the similarity between the generated and real target graphs, we selected four performance metrics: 1) one metric is the distance between a generated and real graph in terms of Closeness centrality (C-dist) [49] and another is similarity score based on the graph kernels of the Weisfeiler–Lehman kernel (wl-sim) [50] and 2) two metrics are used to evaluate the node degree distribution correlation between the generated and real target graphs by Jensen–Shannon distances (JS) and the Wasserstein distances (WDs). Ideally, high-quality generated graphs should be diverse and similar, but not identical. Thus, uniqueness is utilized to capture the diversity of generated graphs. To calculate the uniqueness of a generated graph, the generated graphs that are subgraph isomorphic to some other generated graphs are first removed. The percentage of graphs remaining after this operation is defined as uniqueness [10], [51]. For example, if the model generates 100 graphs, all of which are identical, the uniqueness is $1/100 = 1\%$.

As shown in Table I (left), our GT-GAN consistently outperforms all other baselines by a large margin, especially when the graph size becomes large (i.e., it outperforms the other methods by 34.6% when the size is 150). Compared to the typical graph generation models, the proposed GT-GAN is benefited in conditioning on the source graph, which provides much more necessary feature information and controllable signals for the required pattern of the target graphs. S-Generator is generally the second best method in terms of these six evaluation metrics, highlighting the effectiveness of our proposed graph encoder and decoder. The S-Generator and GT-GAN have both achieved 100% uniqueness in the scale-free datasets

TABLE I

STATISTICS-BASED EVALUATION RESULTS FOR THE SCALE-FREE GRAPHS (*Uniq.* IS SHORT FOR UNIQUENESS)

Size	Methods	JS	WD	C-dist	wl-sim	uniq.
10	GraphRNN	0.47	1.64	0.5319	0.2470	0.45
	GraphVAE	0.67	2.85	0.6664	0.3723	0.42
	GraphGMG	0.43	1.69	0.4763	0.3701	0.73
	S-Generator	**0.35**	0.80	0.2465	0.4185	**1.00**
	GT-GAN	**0.35**	**0.77**	**0.2379**	**0.4195**	**1.00**
100	GraphRNN	0.48	0.90	0.6519	0.2713	0.65
	S-Generator	**0.14**	**0.30**	**0.1501**	0.3522	**1.00**
	GT-GAN	0.15	0.31	0.2087	**0.4078**	**1.00**
150	GraphRNN	0.42	0.95	0.6266	0.2891	0.76
	S-Generator	0.08	0.29	**0.1101**	0.3493	**1.00**
	GT-GAN	**0.07**	**0.27**	0.2105	**0.3926**	**1.00**

with graph size from 10 to 100, while, for typical graph generation methods, such as GraphGMG and GraphRNN, the largest uniqueness is around 73% and 75%, respectively. This advantage comes from the randomness introduced into the generation process and the fact that GT-GAN generates the graphs conditioning on the source graphs that are naturally different from each other.

2) Classifier-Based Evaluation: To further evaluate the generated graphs, we propose a novel classifier-based evaluation schema. We assume that the generated target graphs should share the same underlying properties as the real target graphs, and thus, the classifiers that are trained with generated graphs should also succeed in classifying the real target graphs. Based on this assumption, we train two classifiers: one is trained by the source and the generated target graphs (Classifier 1), and another is trained by the source and the real target graphs (Classifier 2). Then, both two classifiers are tested to distinguish the real source and target graphs in the testing set. Here, Classifier 2 is regarded as the 'Gold Standard' that acts as the "best-possible performer" and is used to judge how "real" the graphs that GT-GAN generate are. A detailed description of classifier-based evaluation is provided in the Appendix. In this experiment, a classifier proposed by Nikolentzo *et al.* [52] is used as the base model for training a graph classifier for both the proposed method, the comparison method, and the gold standard. Table II shows the average results of graph classifiers: precision, recall, area under Roc (AUC), and F1-measure for different methods. The "gold standard" is a classifier trained on the real source and target graphs, which is used to compare with those trained on the graphs generated by different models. For small graphs (e.g., fewer than 10), the power-law property of scale-free networks is less obvious compared to larger size graphs, which may explain why the tasks on smaller scale-free graphs are more difficult. However, when the size of graphs increases, GT-GAN becomes closer to the performance of the "gold standard" with average differences of 10% and 9% on F1 accordingly on two subdatasets, and it significantly outperforms the other methods by large margins up to 51% and 19% on F1, respectively.

TABLE II

CLASSIFIER-BASED EVALUATION RESULTS FOR THE SCALE-FREE GRAPHS

Size	Method	P	R	AUC	F1
10	GraphRNN	0.31	0.11	0.49	0.16
	GraphVAE	0.75	0.23	**0.65**	0.35
	GraphGMG	0.42	0.12	0.49	0.18
	S-Generator	0.46	**0.83**	0.43	0.59
	GT-GAN	**1.00**	0.50	0.52	**0.67**
	Gold Standard	*1.00*	*0.74*	*0.82*	*0.77*
100	GraphRNN	0.61	0.65	0.67	0.60
	S-Generator	0.50	**1.00**	0.50	0.67
	GT-GAN	**0.72**	0.69	**0.68**	**0.70**
	Gold Standard	*0.99*	*0.61*	*0.81*	*0.75*
150	GraphRNN	0.73	**0.92**	0.92	0.81
	S-Generator	0.90	0.50	0.50	0.67
	GT-GAN	**0.94**	0.79	**0.96**	**0.86**
	Gold Standard	*0.99*	*0.93*	*0.96*	*0.95*

TABLE III

CLASSIFIER-BASED RESULTS FOR USER AUTHENTICATION DATASETS

Size	Method	P	R	AUC	F1-score
50	GraphRNN	0.34	0.36	0.50	0.36
	S-Generator	0.72	0.61	0.74	0.66
	GT-GAN	**0.79**	**0.68**	**0.78**	**0.73**
	Gold Standard	*0.97*	*0.97*	*0.97*	*0.97*
300	S-Generator	0.77	0.58	0.62	0.66
	GT-GAN	**0.84**	**0.66**	**0.79**	**0.74**
	Gold Standard	*0.98*	*0.96*	*0.97*	*0.97*

TABLE IV

STATISTICS-BASED EVALUATION FOR USER AUTHENTICATION DATASETS (**UNIQ.** IS SHORT FOR UNIQUENESS)

Size	Method	C-dist	wl-sim	uniq.
50	GraphRNN	0.7456	0.6265	0.46
	S-Generator	0.4414	0.9137	**1.00**
	GT-GAN	**0.1924**	**0.9439**	0.83
300	S-Generator	0.0702	0.9846	**1.00**
	GT-GAN	**0.0681**	**0.9864**	**1.00**

TABLE V

RESULTS FOR THE IOT DATASETS (*Uniq.* IS SHORT FOR UNIQUENESS)

Size	Method	R2	Acc(%)	C-dist	wl-sim	uniq.
20	GraphRNN	0.16	83.97	0.6913	0.3334	0.56
	GraphVAE	0.39	81.19	**0.3283**	0.3471	0.76
	GT-GAN	**0.67**	**92.00**	0.4653	**0.3536**	**1.00**
40	GraphRNN	0.44	70.54	0.6908	0.3333	0.76
	GraphVAE	**0.73**	66.60	0.4683	0.2603	0.69
	GT-GAN	0.69	**93.94**	**0.3378**	**0.3758**	**1.00**
60	GraphRNN	0.52	61.07	0.6914	0.3333	**1.00**
	GraphVAE	0.00	50.64	0.3896	0.3510	**1.00**
	GT-GAN	**0.62**	**94.63**	**0.3051**	**0.3899**	**1.00**

D. Evaluation Results on User Authentication Datasets

1) Classifier-Based Evaluation: As shown in Table III, classifiers trained by the graphs generated by GT-GAN can classify normal and hacked behaviors effectively with AUC above 0.78, which is well above the 0.5 obtained using a random model. GT-GAN significantly outperforms other methods by

TABLE VI

PEARSON CORRELATION BETWEEN THE PREDICTED GRAPH AND THE EMPIRICAL GRAPH ON HCP DATASETS (RES FOR RESTING, EMO FOR EMOTION, GAM FOR GAMBLING, AND LANG FOR LANGUAGE)

	Full Normalized Network-matrices					L2 Regularized Network-matrices				
Method	Res	Emo	Gam	Lang	Motor	Res	Emo	Gam	Lang	Motor
Ganlan et al. [53]	0.23	0.14	0.14	0.14	0.15	0.41	0.42	0.44	0.44	0.45
Abdelnour et al. [54]	0.23	0.14	0.14	0.15	0.15	0.41	0.42	0.43	0.44	0.44
Meier et al. [55]	0.26	0.16	0.15	0.16	0.16	0.43	0.44	0.46	0.46	0.47
Abdelnour et al. [56]	0.23	0.14	0.14	0.15	0.15	0.40	0.41	0.43	0.43	0.44
Guo et al. [42]	0.13	0.34	0.04	0.17	0.14	0.43	0.43	0.46	0.45	0.45
GT-GAN	**0.45**	**0.34**	**0.34**	**0.35**	**0.36**	**0.66**	**0.62**	**0.63**	**0.64**	**0.63**

around 25%, 16%, 24.5%, and 22.1%, respectively, on the four metrics: precision (P), recall (R), AUC, and F1-score for the trained classifier. GT-GAN performs consistently better than other methods when the graph size increases from 50 to 300.

2) Statistics-Based Evaluation: In addition, GT-GAN clearly outperformed the S-Generator in a statistics-based evaluation setting. For example, there are three direct evaluation metrics (i.e., En-dist, C-dist, and wl-sim) mentioned above, which are also tested, and the results can be found in Table IV. The proposed GT-GAN has the best performance in all three aspects both in small scale graphs and large scale graphs. Specifically, for the graphs with size 50, GT-GAN significantly outperforms S-Generator by around 2.2%, 31.2%, and 3.2%, respectively, on the three metrics. For the graphs with size 300, GT-GAN significantly outperforms S-Generator by around 35.2%, 16.7%, and 0.2%, respectively, on the three metrics. This confirms that using a translator alone to learn a deterministic output given a source graph is not sufficient to capture the generic distribution of the target graphs. In addition, the S-Generator and GT-GAN have both achieved 100% uniqueness in the user authentication datasets with graph size 300, while, when the graph becomes smaller (e.g., graph size of 50), the uniqueness of GT-GAN becomes 0.83 due to the simplicity of the graph structure, which is still 37% higher than GraphRNN.

E. Evaluation Results on IoT Dataset

Table V compared the performance of GT-GAN and other comparison methods for the IoT dataset by examining the edges of the generated and real target graphs for four metrics: coefficient of determination score (R2) and accuracy (ACC) for the correct existence of edges among all the pairs of nodes, as well as two statistic-based metrics mentioned above. The results show that GT-GAN performed almost the best for all three subsets. Due to the L1-loss required to maintain topology pattern similarity, GT-GAN outperformed the comparison methods with around 8%, 26%, and 40% superiorities in ACC for the three subsets with the sizes of 20, 40, and 60, respectively. As the graph size increases, the properties of the graphs are clearer, and the superiority of the proposed GT-GAN model is more obvious. For example, the proposed GT-GAN model outperforms the GraphRNN and GraphVAE by 67.6% and 21.5%, on average, on the metrics of C-dist and wl-sim, respectively, when the graph size is 40, and by 75.4% and 12.2% when the graph size is 60. The GT-GAN has achieved 100% uniqueness in IoT datasets with graph size from 20 to 60, which outperforms the typical graph generation methods, such as GraphVAE and GraphRNN, by around 29.8% and 22%, respectively. This shows the superiority of GT-GAN in generating a diverse range of graphs.

F. Evaluation Results for HCP Datasets

We consider four classic brain network prediction methods and one graph-based method that uses SC to predict FC [53], [56] as the comparison methods for this experiment in the domain of brain network science. Abdelnour *et al.* [56] considered the graph spectral transformation kernels by assuming that SC and FC share the identical eigenvectors on their Laplacians. Another two methods directly consider the graph translation between SC and FC. The goal is to predict the FC given the SC when the brain is doing different tasks. The Pearson coefficient was used as metric by comparing the predicted graphs with the empirical target graphs, which is widely used in the domain of brain network [57], [58]. As shown in Table VI, the proposed GT-GAN achieves the highest Pearson coefficient on all the five subdatasets regarding both the full correlation and L2 correlation with superiority of about 52.38% and 32.5% averagely. The great superiority of the proposed GT-GAN over the typical methods in the domain of brain networks validates the power of deep learning-based models in handling complex real-world applications.

G. Model Ablation Study

To further validate the superiority of the proposed graph convolution and deconvolution layers, an ablation experiment was conducted. The graph encoder was replaced by GCN [2], deep convolution neural network (DCNN) [59], and Graph U-NET [44]. The graph decoder was replaced by the decoder in variational graph auto-encoder (VGAE) [60]. There were thus three method combinations for comparison.

Table VII shows parts of the results in the ablation study of the proposed encoder and decoder on part of the scale-free graphs with size 50 (Scale-III) and user authentication with graph size 50 (Auth). The encoder of GT-GAN outperformed both the GCN- and DCNN-based encoders by a large margin on these datasets, especially for the real-world datasets, where the edges of the graphs can have a very complex meaning. For example, on Auth-I, GT-GAN performed 41% and 38% better,

TABLE VII

ABLATION STUDY OF GRAPH ENCODER AND DECODER IN GT-GAN (SCALE-III DENOTES TO THE SCALE FREE DATASETS WITH GRAPH SIZE OF 50; AUTH-I DENOTES TO THE USER AUTHENTICATION DATASET WITH GRAPH SIZE OF 50; AND IOT-III DONOTES TO THE IOT DATASET WITH GRAPH SIZE OF 60)

Dataset	Method	JS	WD	C-dist	wl-sim
Scale-III	GCN+decoder	0.18	18.84	0.6751	0.4031
	DCNN+decoder	0.65	0.77	0.6745	0.4032
	Unet+decoder	0.69	5.77	0.6496	0.4040
	Encoder+VGAE	0.31	43.78	0.2559	0.4003
	GT-GAN	**0.15**	**0.31**	**0.2087**	**0.4078**
		P	F1	C-dist	wl-sim
Auth-I	GCN+decoder	0.31	0.33	0.7494	0.6632
	DCNN+decoder	0.59	0.57	0.3349	0.6851
	Unet+decoder	0.41	0.49	0.6859	0.9239
	Encoder+VGAE	0.49	0.47	0.3129	0.6111
	GT-GAN	**0.79**	**0.73**	**0.1924**	**0.9439**
		R2	Acc(%)	C-dist	wl-sim
IoT-III	GCN+decoder	0.46	92.69	0.4349	0.3304
	DCNN+decoder	0.52	93.26	0.3217	0.3292
	Unet+decoder	0.45	92.46	**0.2771**	0.3310
	Encoder+VGAE	0.12	88.14	0.4876	0.3333
	GT-GAN	**0.62**	**94.63**	0.3051	**0.3899**

on average, than other encoders in terms of precision and F1-scores, respectively. Similar results can also be found for the proposed decoder, which demonstrates that the proposed decoder in GT-GAN was both effective and irreplaceable for the graph generation.

V. CONCLUSION AND FUTURE WORKS

This article focuses on a new problem: DGT. To deal with it, we propose a novel GT-GAN model that translates a source graph to a target graph. To learn both global and local mappings between graphs, a new graph encoder–decoder model is proposed while preserving the graph patterns in various scales. Extensive experiments were conducted on synthetic and real-world datasets to compare with the SOTA graph generation models. Experimental results show that our GT-GAN can discover the ground-truth translation rules and significantly outperforms other baselines in terms of both effectiveness and scalability.

This article opens a thread of research for DGT in many practical applications. As a new topic, some critical problem regarding the DGT still needs to be explored. We suggest that researchers in social sciences and machine learning investigate questions, such as the following.

1) *Intepretability of DGT:* The interpretability and explainability enhancements of the model could fill the gaps among data scientists and domain experts, leading to an additional profound understanding of the underlying mechanism of the graph transformation.
2) *Scalable DGT:* Translation for large-scale graphs (i.e., containing millions of nodes) could expand the applicability of DGT to a broader class of problems, such as applications on social networks and traffic networks.

REFERENCES

[1] Y. Li, R. Zemel, M. Brockschmidt, and D. Tarlow, "Gated graph sequence neural networks," in *Proc. ICLR*, Apr. 2016.
[2] T. N. Kipf and M. Welling, "Semi-supervised classification with graph convolutional networks," in *Proc. Int. Conf. Learn. Represent. (ICLR)*, 2017, pp. 1–14.
[3] P. Veličković, G. Cucurull, A. Casanova, A. Romero, P. Liò, and Y. Bengio, "Graph attention networks," in *Proc. Int. Conf. Learn. Represent.*, 2018, pp. 1–12.
[4] J. Gilmer, S. S. Schoenholz, P. F. Riley, O. Vinyals, and G. E. Dahl, "Neural message passing for quantum chemistry," in *Proc. 34th Int. Conf. Mach. Learn.*, vol. 70, 2017, pp. 1263–1272.
[5] W. Hamilton, Z. Ying, and J. Leskovec, "Inductive representation learning on large graphs," in *Proc. Adv. Neural Inf. Process. Syst.*, 2017, pp. 1025–1035.
[6] M. Niepert, M. Ahmed, and K. Kutzkov, "Learning convolutional neural networks for graphs," in *Proc. Int. Conf. Mach. Learn. (ICML)*, 2016, pp. 2014–2023.
[7] J. Atwood, S. Pal, D. Towsley, and A. Swami, "Sparse diffusion-convolutional neural networks," in *Proc. Adv. Neural Inf. Process. Syst.*, 2016.
[8] B. Wu, Y. Liu, B. Lang, and L. Huang, "DGCNN: Disordered graph convolutional neural network based on the Gaussian mixture model," *Neurocomputing*, vol. 321, pp. 346–356, Dec. 2018.
[9] M. Simonovsky and N. Komodakis, "GraphVAE: Towards generation of small graphs using variational autoencoders," in *Proc. Int. Conf. Artif. Neural Netw.*, Cham, Switzerland: Springer, 2018, pp. 412–422.
[10] J. You, R. Ying, X. Ren, W. Hamilton, and J. Leskovec, "GraphRNN: Generating realistic graphs with deep auto-regressive models," in *Proc. Int. Conf. Mach. Learn.*, 2018, pp. 5694–5703.
[11] M. J. Kusner, B. Paige, and J. M. Hernández-Lobato, "Grammar variational autoencoder," in *Proc. Int. Conf. Mach. Learn.*, 2017, pp. 1945–1954.
[12] H. Dai, Y. Tian, B. Dai, S. Skiena, and L. Song, "Syntax-directed variational autoencoder for structured data," in *Proc. Int. Conf. Learn. Represent.*, Feb. 2018.
[13] Y. Li, O. Vinyals, C. Dyer, R. Pascanu, and P. Battaglia, "Learning deep generative models of graphs," 2018, *arXiv:1803.03324*.
[14] B. Samanta, A. De, G. Jana, P. K. Chattaraj, N. Ganguly, and M. Gomez-Rodriguez, "NeVAE: A deep generative model for molecular graphs," 2018, *arXiv:1802.05283*.
[15] W. Jin, R. Barzilay, and T. Jaakkola, "Junction tree variational autoencoder for molecular graph generation," in *Proc. Int. Conf. Mach. Learn.*, 2018, pp. 2323–2332.
[16] X. Guo, S. Wang, and L. Zhao, *Graph Neural Networks: Graph Transformation*. Singapore: Springer, 2022, pp. 251–275, doi: 10.1007/978-981-16-6054-2_12.
[17] P. Isola, J.-Y. Zhu, T. Zhou, and A. A. Efros, "Image-to-image translation with conditional adversarial networks," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jul. 2017, pp. 1125–1134.
[18] Z. Yang, W. Chen, F. Wang, and B. Xu, "Improving neural machine translation with conditional sequence generative adversarial nets," in *Proc. Conf. North Amer. Chapter Assoc. Comput. Linguistics, Hum. Lang. Technol., (Long Papers)*, vol. 1, 2018, pp. 1346–1355.
[19] W. Jin, K. Yang, R. Barzilay, and T. Jaakkola, "Learning multimodal graph-to-graph translation for molecule optimization," in *Proc. Int. Conf. Learn. Represent.*, 2018.
[20] K. Do, T. Tran, and S. Venkatesh, "Graph transformation policy network for chemical reaction prediction," in *Proc. 25th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, Jul. 2019, pp. 750–760.
[21] M. Sun and P. Li, "Graph to graph: A topology aware approach for graph structures learning and generation," in *Proc. 22nd Int. Conf. Artif. Intell. Statist.*, 2019, pp. 2946–2955.
[22] M. Gori, G. Monfardini, and F. Scarselli, "A new model for learning in graph domains," in *Proc. IEEE Int. Joint Conf. Neural Netw.*, vol. 2, Jul. 2005, pp. 729–734.
[23] F. Scarselli, M. Gori, A. C. Tsoi, M. Hagenbuchner, and G. Monfardini, "The graph neural network model," *IEEE Trans. Neural Netw.*, vol. 20, no. 1, pp. 61–80, Dec. 2009.
[24] J. Bruna, W. Zaremba, A. Szlam, and Y. Lecun, "Spectral networks and locally connected networks on graphs," in *Proc. Int. Conf. Learn. Represent. (ICLR), CBLS*, Apr. 2014, pp. 1–14.
[25] M. Defferrard, X. Bresson, and P. Vandergheynst, "Convolutional neural networks on graphs with fast localized spectral filtering," in *Proc. Adv. Neural Inf. Process. Syst.*, 2016, pp. 3844–3852.
[26] P. Veličković, W. Fedus, W. L. Hamilton, P. Liò, Y. Bengio, and R. D. Hjelm, "Deep graph infomax," in *Proc. Int. Conf. Learn. Represent.*, 2018, pp. 1–46.

[27] Y. Bai *et al.*, "Unsupervised inductive graph-level representation learning via graph-graph proximity," in *Proc. 28th Int. Joint Conf. Artif. Intell.*, Aug. 2019, pp. 1988–1994.

[28] Y. Chen, L. Wu, and M. J. Zaki, "Iterative deep graph learning for graph neural networks: Better and robust node embeddings," in *Proc. 34th Annu. Conf. Neural Inf. Process. Syst. (NeurIPS)*, 2020, pp. 19314–19326.

[29] A. Bojchevski, O. Shchur, D. Zügner, and S. Günnemann, "NetGAN: Generating graphs via random walks," in *Proc. Int. Conf. Mach. Learn.*, 2018, pp. 610–619.

[30] L. Wu *et al.*, "Graph neural networks for natural language processing: A survey," 2021, *arXiv:2106.06090*.

[31] D. Beck, G. Haffari, and T. Cohn, "Graph-to-sequence learning using gated graph neural networks," in *Proc. 56th Annu. Meeting Assoc. Comput. Linguistics (Long Papers)*, vol. 1, 2018, pp. 273–283.

[32] J. Bastings, I. Titov, W. Aziz, D. Marcheggiani, and K. Simaan, "Graph convolutional encoders for syntax-aware neural machine translation," in *Proc. Conf. Empirical Methods Natural Lang. Process.*, 2017, pp. 1957–1967.

[33] K. Xu, L. Wu, Z. Wang, Y. Feng, M. Witbrock, and V. Sheinin, "Graph2Seq: Graph to sequence learning with attention-based neural networks," 2018, *arXiv:1804.00823*.

[34] K. Xu, L. Wu, Z. Wang, M. Yu, L. Chen, and V. Sheinin, "Exploiting rich syntactic information for semantic parsing with graph-to-sequence model," in *Proc. Conf. Empirical Methods Natural Lang. Process.*, 2018, pp. 918–924.

[35] S. Li, L. Wu, S. Feng, F. Xu, F. Xu, and S. Zhong, "Graph-to-tree neural networks for learning structured input-output translation with applications to semantic parsing and math word problem," 2020, *arXiv:2004.13781*.

[36] Y. Chen, L. Wu, and M. J. Zaki, "Reinforcement learning based graph-to-sequence model for natural question generation," in *Proc. Int. Conf. Learn. Represent.*, 2019, pp. 1–17.

[37] Y. Gao, L. Wu, H. Homayoun, and L. Zhao, "DynGraph2Seq: Dynamic-graph-to-sequence interpretable learning for health stage prediction in online health forums," in *Proc. IEEE Int. Conf. Data Mining (ICDM)*, Nov. 2019, pp. 1042–1047.

[38] L. Zhao, "Event prediction in the big data era: A systematic survey," *ACM Comput. Surv.*, vol. 54, no. 5, pp. 1–37, 2021.

[39] X. Peng, D. Gildea, and G. Satta, "AMR parsing with cache transition systems," in *Proc. 32nd AAAI Conf. Artif. Intell.*, 2018, pp. 1–8.

[40] D. Gildea, G. Satta, and X. Peng, "Cache transition systems for graph parsing," *Comput. Linguistics*, vol. 44, no. 1, pp. 85–118, Mar. 2018.

[41] Y. Wang, W. Che, J. Guo, and T. Liu, "A neural transition-based approach for semantic dependency graph parsing," in *Proc. 32nd AAAI Conf. Artif. Intell.*, 2018, pp. 5561–5568.

[42] X. Guo, L. Zhao, C. Nowzari, S. Rafatirad, H. Homayoun, and S. M. P. Dinakarrao, "Deep multi-attributed graph translation with node-edge co-evolution," in *Proc. IEEE Int. Conf. Data Mining (ICDM)*, Nov. 2019, pp. 250–259.

[43] M. L. Seltzer, D. Yu, and Y. Wang, "An investigation of deep neural networks for noise robust speech recognition," in *Proc. IEEE Int. Conf. Acoust., Speech Signal Process.*, May 2013, pp. 7398–7402.

[44] H. Gao and S. Ji, "Graph U-Nets," in *Proc. 36th Int. Conf. Mach. Learn.*, vol. 97, 2019, pp. 2083–2092.

[45] M. Zhang and Y. Chen, "Link prediction based on graph neural networks," in *Proc. NIPS*, vol. 31, 2018, pp. 5165–5175.

[46] B. Bollobás, C. Borgs, J. Chayes, and O. Riordan, "Directed scale-free graphs," in *Proc. 40th Annu. ACM-SIAM Symp. Discrete Algorithms*, 2003, pp. 132–139.

[47] A. D. Kent, *Comprehensive, Multi-Source Cyber-Security Events*. Los Alamos, NM, USA: Los Alamos National Laboratory, 2015.

[48] D. C. Van Essen *et al.*, "The WU-minn human connectome project: An overview," *NeuroImage*, vol. 80, pp. 62–79, Oct. 2013.

[49] L. C. Freeman, "Centrality in social networks conceptual clarification," *Soc. Netw.*, vol. 1, no. 3, pp. 215–239, Jan. 1978.

[50] N. Shervashidze, P. Schweitzer, E. J. van Leeuwen, K. Mehlhorn, and K. M. Borgwardt, "Weisfeiler-lehman graph kernels," *J. Mach. Learn. Res.*, vol. 12, pp. 2539–2561, Sep. 2011.

[51] X. Guo and L. Zhao, "A systematic survey on deep generative models for graph generation," 2020, *arXiv:2007.06686*.

[52] G. Nikolentzos, P. Meladianos, A. J.-P. Tixier, K. Skianis, and M. Vazirgiannis, "Kernel graph convolutional neural networks," in *Proc. Int. Conf. Artif. Neural Netw.*, Cham, Switzerland: Springer, 2018, pp. 22–32.

[53] R. F. Galán, "On how network architecture determines the dominant patterns of spontaneous neural activity," *PLoS ONE*, vol. 3, no. 5, May 2008, Art. no. e2148.

[54] F. Abdelnour, H. U. Voss, and A. Raj, "Network diffusion accurately models the relationship between structural and functional brain connectivity networks," *NeuroImage*, vol. 90, pp. 335–347, Apr. 2014.

[55] J. Meier *et al.*, "A mapping between structural and functional brain networks," *Brain Connectivity*, vol. 6, no. 4, pp. 298–311, May 2016.

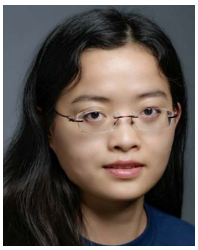
[56] F. Abdelnour, M. Dayan, O. Devinsky, T. Thesen, and A. Raj, "Functional brain connectivity is predictable from anatomic network's Laplacian Eigen-structure," *NeuroImage*, vol. 172, pp. 728–739, 2018.

[57] S. M. Smith *et al.*, "Methods for network modelling from high quality rfMRI data," in *Proc. OHBM Annu. Meeting*, 2014, p. 1718.

[58] SteveSmith. *Fslnets*. Accessed: May 2018. [Online]. Available: https://fsl.fmrib.ox.ac.U.K./fsl/fslwiki/FSLNets

[59] J. Atwood and D. Towsley, "Diffusion-convolutional neural networks," in *Proc. Adv. Neural Inf. Process. Syst.*, 2016, pp. 1993–2001.

[60] T. N. Kipf and M. Welling, "Variational graph auto-encoders," 2016, *arXiv:1611.07308*.



Xiaojie Guo received the Ph.D. degree in information technology from George Mason University, Fairfax, VA, USA, in 2021.

She is a Research Scientist with the JD.COM Silicon Valley Research Center, Mountain View, CA, USA. Her research interests include data mining, artificial intelligence, and machine learning, with special interests in deep learning on graphs (DLG), deep graph transformation, deep graph generation, and disentangled representation learning. She has published over 21 articles in top-tier conferences and journals, such as Conference on Knowledge Discovery and Data Mining (KDD), IEEE International Conference on Data Mining (ICDM), ICLR, NeurIPS, Web Conference (WWW), AAAI Conference on Artificial Intelligence (AAAI), the PROCEEDINGS OF THE IEEE, and CIKM.

Dr. Guo received the Best Paper Award from ICDM in 2019.

Lingfei Wu (Member, IEEE) received the Ph.D. degree in computer science from the College of William and Mary, Williamsburg, VA, USA, in 2016.

He is a Principal Scientist with the JD.COM Silicon Valley Research Center, Mountain View, CA, USA, leading a team of 30+ machine learning/natural language processing (NLP) scientists and software engineers to build intelligent e-commerce personalization systems. Previously, he was a Research Staff Member with the IBM Thomas J. Watson Research Center and has published more than 90 top-ranked conference and journal papers and is a Co-Inventor of more than 40 filed US patents.

Dr. Wu was a recipient of the Best Paper Award and the Best Student Paper Award of several conferences such as IEEE ICC'19, AAAI Conference on Artificial Intelligence (AAAI) workshop on DLGMA'20, and Conference on Knowledge Discovery and Data Mining (KDD) workshop on deep learning on graphs (DLG)'19. In addition, he has served as an Associate Editor for the IEEE TRANSACTIONS ON NEURAL NETWORKS AND LEARNING SYSTEMS, *ACM Transactions on Knowledge Discovery from Data*, and the *International Journal of Intelligent Systems*.

Liang Zhao (Senior Member, IEEE) is an Assistant Professor with the Department of Computer Science, Emory University, Atlanta, GA, USA. His research interests include data mining, artificial intelligence, and machine learning, with special interests in spatiotemporal and network data mining, and deep learning on graphs (DLG). He has published over articles in top-tier conferences and journals, such as Conference on Knowledge Discovery and Data Mining (KDD), IEEE International Conference on Data Mining (ICDM), IEEE TRANSACTIONS ON KNOWLEDGE AND DATA ENGINEERING (TKDE), PROCEEDINGS OF THE IEEE, *ACM Transactions on Knowledge Discovery from Data* (TKDD), *ACM Transactions on Spatial Algorithms and Systems* (TSAS), International Joint Conference on Artificial Intelligence (IJCAI), AAAI Conference on Artificial Intelligence (AAAI), Web Conference (WWW), CIKM, SIGSPATIAL, and SDM.

Dr. Zhao received the NSF CAREER Award, the Amazon Research Award, the Jeffress Trust Award in 2019, and the Outstanding Doctoral Student in the Department of Computer Science at Virginia Tech in 2017.