Knowledge and Information Systems (2021) 63:1305–1337
https://doi.org/10.1007/s10115-021-01553-9

REGULAR PAPER

Deep graph transformation for attributed, directed, and signed networks

Xiaojie Guo[1] · Liang Zhao[2] · Houman Homayoun[3] · Sai Manoj Pudukotai Dinakarrao[4]

Received: 29 January 2020 / Revised: 18 February 2021 / Accepted: 23 February 2021 /
Published online: 3 April 2021

Abstract
Generalized from image and language translation, the goal of graph translation or transformation is to generate a graph of the target domain on the condition of an input graph of the source domain. Existing works are limited to either merely generating the node attributes of graphs with fixed topology or only generating the graph topology without allowing the node attributes to change. They are prevented from simultaneously generating both node and edge attributes due to: (1) difficulty in modeling the iterative, interactive, and asynchronous process of both node and edge translation and (2) difficulty in learning and preserving the inherent consistency between the nodes and edges in generated graphs. A general, end-to-end framework for jointly generating node and edge attributes is needed for real-world problems. In this paper, this generic problem of multi-attributed graph translation is named and a novel framework coherently accommodating both node and edge translations is proposed. The proposed generic edge translation path is also proven to be a generalization of existing topology translation models. Then, in order to discover and preserve the consistency of the generated nodes and edges, a spectral graph regularization based on our nonparametric graph Laplacian is designed. In addition, two extensions of the proposed model are developed for signed and directed graph translation. Lastly, comprehensive experiments on both synthetic and real-world practical datasets demonstrate the power and efficiency of the proposed method.

Keywords Multi-attributed graphs · Deep graph transformation · Graph Laplacian · Graph neural network

1 Introduction

Many structured predictions problems are encountered in the process of "translating" input data (e.g., image or text) into corresponding output data, namely modeling a translation mapping from the source domain to the target domain. Taking the problems in image processing and computer vision as examples, they involve a "translation" from an input image into the target output image. Similarly, language translation [60–62] also involves "translation" problems where sentences (sequences of words) in one language are translated into corresponding sentences in another language. In recent years, rapidly growing attention has been given to

Extended author information available on the last page of the article

this general translation problem, which is critical yet has been inherently intensely difficult. The classic data translation problem typically deals with data under a special topology. As in the examples mentioned above, both grids and sequences are special types of graphs. An image is a type of grid where each pixel is a node and each node has connections to its eight spatial neighbors. Texts are typically considered sequences where each word is a node and an edge exists between two contextual words. However, many real-world problems require working on data with more adjustable structures than girds and sequences. Thus, more powerful translation techniques are needed to handle more general graph-structured data, and such techniques have been proposed and widely applied to many applications, e.g., predicting objects' future state in a system in the physical domain based on the fixed relations (e.g., gravitational forces) among objects [4] and forecasting the average traffic speed of traffic networks [39,64,65]. Though generic graph-structured data is considered in these works, the topology of the graphs from the input domain and target domain are assumed to be the same. Thus, they cannot model or predict the change of the graph topology.

To predict the topology change of the graphs during transformation, a few deep-learning-based graph translation problems have appeared very recently. This problem is important and promising to the domains where the variety of the graph topology is possible and commonplace, such as social networks and cyber-networks. 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 of highly beneficial to accurately predict contact graph in target circumstances regarding situational awareness and resource allocation. There exist some topology translation models [21,55] that generate the graph topology (i.e., edges) in a target domain based on that in source domain. However, they focus on predicting the graph topology without considering the change of the node attributes.

Consequently, existing works can only generate node attributes with fixed topology or generate edge attributes with unchanged node attributes. However, both node and edge attributes need change and to be generated in many real-world problems. In this paper, we address this general problem *multi-attributed graph transformation*, the goal of which is to learn the mapping from the input multi-attributed graph to the target multi-attributed graph via observing a large amount of observed input-target graph pairs. This novel problem has many important real-world applications, such as exploring the mapping from biological structural to functional brain network in the domain of neural science [2], network intervention research [50] in the domain of network security, and circuit simplification where a logical expression is reduced to a logically equivalent expression with fewer operations [31]. For instance, the malware confinement[1] process of the Internet of Things (IoT) is a typical multi-attributed graph translation problem as shown in Fig. 1. The initial status of the IoT is regarded as input, and the goal is to predict the target network, which is ideally the optimal status of the network with modified connections (i.e., edges) and changed devices (i.e., nodes) state that help prevent malware propagation and maintain network throughput. Epidemic controlling is also an example of a multi-attributed graph translation problem. Its goal is to model the joint change process of the initial disease contact network (i.e., multi-attributed edges) and the human health stage (i.e., multi-attributed nodes) after particular interventions.

The multi-attributed graph translation problem is highly complicated, and there is no general framework but only ad hoc methods for a few specific domains. These methods extend

[1] A device infected in an IoT network can propagate to other nodes connected to it, leading to contamination of the whole network, such as in the MiraiBot attack. As such, it is non-trivial to confine the malware to limit the infection and also equally important to maintain overall network connectivity and performance.

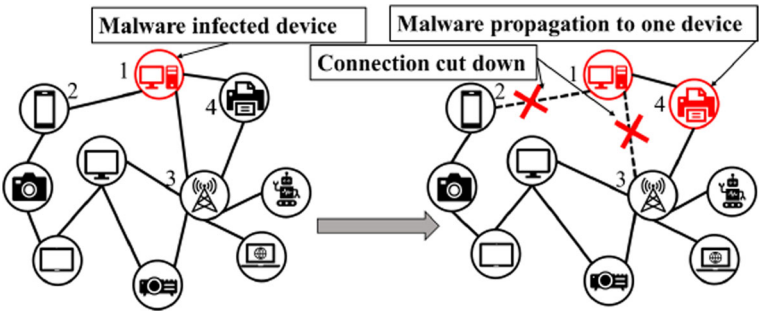


Fig. 1 Given the network at time t (shown in the left graph), malware confinement is conducted to predict the most optimal status at time $t + \gamma$ shown in the right, where Devices 2 and 3 are protected by cutting the links (edges) to the compromised Device 1, while the Device 4 is propagated by malware without cutting link

graph theory into attributed networks transformation [13], such as the graph-based reasoning on the knowledge graph [24], where the mechanistic models and transformation rules are intensively hand-crafted based on the pre-knowledge of the semantic or logic relationship of entities (i.e., nodes) and relations (i.e., edges). The other example is graph morphism, where the relationship between the input and target graphs is already known in advance [5,14]. They typically rely on the network generation principles predefined by human heuristics and prior knowledge, which effectively abstract the high-dimension problems down to a manageable scale. Such methods usually fit well toward the properties that have been covered by the predefined principles, but not on those that have not been covered [29]. However, in many domains, the network properties and transformation principles are largely unknown yet, such as those for explaining the mechanisms of mental diseases in brain networks [1,2] and protein structure folding [58]. As a result, there is a great need for a generic, efficient end-to-end framework for general multi-attributed graph translation problems. By observing a large amount of observed data, such a framework is required to be able to broadly learn translation mapping, solve human bias, and preserve efficient prediction.

In this paper, we aim to solve the generic problem of multi-attributed graph translation, which is difficult for the existing methods to handle due to the following challenges: (1) **The node and edge attribute translations are mutually dependent**. The translation of edge attributes should not only rely on edges, but also the node attributes. For instance in Fig. 1, two links are cut down since their neighboring Device 1 is infected by malware, which illustrates the interaction between nodes and edges. Similarly, translation of node attributes also needs to consider edges, e.g., Device 4 is infected due to its link to Device 1. No existing works can jointly consider and solve all the above affairs. (2) **Iterative and asynchronous changes of node and edge attributes during graph translation**. A series of iterative changes in both edge and node attributes may be included in the multi-attributed graph translation process. For instance, in Fig. 1, the translation could involve several steps due to the iterative malware propagation from one device to the others. The connection to a device may be cut (i.e., edge changes) right after it is infected by malware (i.e., node attribute change). It is very critical, yet hard, to learn these orders and dependencies of how node and edge attributes change during the translation. (3) **Difficulty in discovering and preserving the correct consistency between node attributes and graph spectra**. Even though the generated node and edge attributes are two different outputs from two translation paths, they should be intensively dependent on each other instead of being unrelated. For example, in Fig. 1, the reason why Devices 2 and 3 on the right network are not infected is that they are no longer linked with the compromised Device

1. Since the consistent patterns of node and edge attributes are very complicated and domain-specific, they are difficult to discover and preserve. (4) **Difficulty in the transformation of signed and directed graphs**. In real-world applications, many types of graphs should be considered, such as signed (e.g., brain net correlation network) and directed (e.g., message network). Signed and directed graphs may incur difficulty in calculating the graph spectrum and frequencies during transformation process. This is because the graph Laplacian may not be symmetric and positive semi-definite anymore.

We believe that this paper is the first work that undertakes all the above challenges and proposes a general framework for the multi-attributed graph translation problem. This paper develops a Node-Edge Co-evolving Deep Graph Translator (NEC-DGT) with novel architecture and techniques for conjoint node and edge translation. Multi-block deep neural networks with mutual node and edge translation paths are designed to translate node and edge attributes jointly. To allow the non-synchronicity of changes in node and edge attributes, a skip connection is applied among different blocks. In addition, a novel spectral graph regularization is proposed to discover and preserve the consistent patterns of nodes and edges in generated graphs. Here, the consistent patterns refers to the shared property in both input and target graphs that the generated node and edge attributes in one graph match each other, namely the generated graph should be semantically valid. For example, in protein interaction networks, two proteins may be connected (edge attributes) only when their gene ontology features (node attributes) are same or correlated. It is important to explore and maintain these kinds of patterns between nodes and edges in the generated graph. The contributions of this work are summarized as follows:

- **The design of a novel framework for multi-attributed graph translation** A multi-attributed graph translation problem is formulated for the first time, and NEC-DGT is proposed to solve this problem. The proposed framework is general for various applications where both node and edge attributes need to be translated.
- **The proposal of novel and generic edge translation layers and blocks** A novel edge translation path is proposed to translate the edge attributes from the source domain to the target domain. Our translation path can handle broad multi-attributed edges and nodes and is proven to be a generalization of the existing topology translation methods.
- **The proposal of a spectral-based regularization that enforces consistency of the generated nodes and edges** A novel nonparametric graph Laplacian regularization and a graph frequency regularization are proposed, with the aim of learning and preserving the inherent relationships between the generated nodes and edges.
- **The extension of the proposed model to signed graph translation** A variant of the proposed model called sign-NEC-DGT is designed by proposing a novel signed spectral-based regularization, which is based on a specially designed definition of graph Laplacian for signed networks.
- **The extension of the proposed model to directed graph translation** A variant of the proposed model called di-NEC-DGT is designed by proposing a novel direct spectral-based regularization, which is based on the approximation of the Perron vector.
- **The conducting of comprehensive experiments to validate the effectiveness and efficiency of the proposed model** Comprehensive experiments on four synthetic and four real-world datasets indicate that NEC-DGT is able to generate graphs close to ground-truth target graphs and considerably surpasses other generative methods.

2 Related works

2.1 Traditional graph transformation

Traditional graph transformation target on handling a specified relationship between the input and output graphs, including graph morphism, conceptual graph projecting and reasoning, and graph matching [43,53]. Specifically, graph morphism aims to rewrite the input graph into its isomorphic graph based on predefined transformation rules [5,14]. For many applications, one requires more than graphs labeled over a finite alphabet. Hence, attributed graph transformation methods are proposed to deal with graphs that are attributed with elements of given data types (e.g. integer, string, boolean) [13,35,47]. They can perform computations (e.g., add two integers) of attributes and define guards that restrict the applicability of rules (e.g., apply the rule only if a certain attribute is above some threshold). Conceptual graph projection and reasoning only handles the conceptual graph which is a knowledge representation scheme in formalizing semantic networks. These kinds of graphs are also attributed graphs with two types of nodes: concepts (which represent objects, entities or ideas) and relation nodes, which represent relations between the concepts [10]. Reasoning based on the conceptual graphs can be regarded as a translation problem where the output graph is generated based on some predefined canonical formation rules, such as equivalence, specialization, and generalization. Concept projection in ontology is used to find out whether two ontologies are semantically compatible or similar.

Though the above problem is about translating an input graph into a target graph, these traditional graph transformation requires the pre-knowledge of the relationship between the input and target graph (e.g., isomorphism or semantically compatible), based on which a set of rules and operations (e.g., projection) are hand-crafted. However, in real-world graphs, the relationship between two graphs can be various and complex and is hard to define and observe. And the graphs are also not limited to semantic graph or logical graphs. Thus, to model and learn the relationship between the input and output graphs, we resort to deep learning methods on graphs via observing from large amounts of data. The most significance of the proposed deep learning-based graph transformation model is that given any large amounts of input-target graphs where the relationship and translation rules are unknown, our proposed data-driven model could automatically learn and model this translation process via optimizing an objective function.

2.2 Graph neural networks learning

In recent years, there has been a surge in research focusing on graph neural networks (GNN), which are generally divided into two categories: Graph Recurrent Networks [20,37,51] and Graph Convolutional Networks [8,12,32,33,42,44,45,59]. Graph Recurrent Networks originates from the early works of graph neural networks proposed by Gori et al. [20] and Scarselli et al. [51] based on recursive neural networks. Another line of research is to generalize convolutional neural networks from grids (e.g., images) to generic graphs. Bruna et al. [7] first introduced the spectral graph convolutional neural networks, and then it was extended by Defferrard et al. [12] using fast localized convolutions, which is further approximated for an efficient architecture for a semi-supervised setting [33]. The above-mentioned methods all focus on the general problem of graph representation learning, which aims to embed the graphs into the low-dimension spaces. They provide the fundamental theories and opera-

tions for many downstream tasks such as node classification, graph classification, and graph generation.

2.3 Deep generative models for graph generation

Deep generative models for graph generation aim to learn the distribution of a set of observed graphs via a deep generative model. At present, two classes of methods in this domain are proposed: (1) domain-specific models and (2) generic models. Domain-specific models [11, 30,36] typically take sequence inputs (such as SMILES for representing molecules) and generate other sequences or parse trees from context-free grammars or sub-graphs by utilizing the collection of valid components, while generic graph generation handles general graphs that are not restricted to specific applications, which is more relevant to this paper. Most of the existing GNN-based graph generation for general graphs have been proposed in the last 2 years and are based on variational auto-encoder (VAE) [22,48,52] and generative adversarial nets (GANs) [6], among others [38,63]. Most of these approaches generate nodes and edges sequentially to form a whole graph, leading to the issues of being sensitive to the generation order and very time-consuming for large graphs. Differently, graph recurrent neural network (GraphRNN) [63] builds an autoregressive generative model on these sequences with long short term memory (LSTM) model and has demonstrated its good scalability. However, the above-mentioned models all deal with the unconditional graph generation instead of generating a target graph conditioning on an input graph, as our problem requires in this paper.

2.4 Graph-structured data translation

The existing graph-structured data translation deal with a similar problem of our method in this paper. However, they either deal with the node attributes prediction or generate the graph topology, without considering both. Node attributes prediction aims at predicting the node attributes given the fixed graph topology [4,18,39,64]. Li et al. [39] propose a diffusion convolution recurrent neural network (DCRNN) for traffic forecasting which incorporates both spatial and temporal dependency in the traffic flow. Yu et al. [64] formulated the node attributes prediction problem of graphs based on the complete convolution structures. Graph topology translation considers the change of graph topology from one domain distributions to another. Guo et al. [21] proposed and tackled graph topology translation problem by proposing a generative model consisting of a graph translator with graph convolution and deconvolution layers and a new conditional graph discriminator. Sun et al. [55] proposed a graphRNN-based model which generates a graph's topology based on another graph. Beyond the above two separate tasks, our proposed model can jointly predict the node attributes and topology via modeling the complex interaction between them during the translation process.

3 Problem formulation

This paper focuses on predicting a *target multi-attributed graph* based on an *input multi-attributed graph* by learning the graph translation mapping between them. The following provides the notations and mathematical problem formulation (Table 1).

Define an input graph as $G(\mathcal{V}_0, \mathcal{E}_0, E_0, F_0)$ where $\mathcal{V}_0$ is the set of N nodes, and $\mathcal{E}_0 \subseteq \mathcal{V}_0 \times \mathcal{V}_0$ is the set of M edges. $e_{i,j} \in \mathcal{E}_0$ is an edge connecting nodes $i \in \mathcal{V}_0$ and $j \in \mathcal{V}_0$. $\mathcal{E}_0$ contains

Table 1 Important notations and descriptions

Notations	Descriptions
$G(\mathcal{V}_0, \mathcal{E}_0, E_0, F_0)$	Input graph with node set $\mathcal{V}_0$, edge set $\mathcal{E}_0$, edge attributes tensor E_0 and node attributes matrix F_0
$G(\mathcal{V}', \mathcal{E}', E', F')$	Target graph with node set $\mathcal{V}'$, edge set $\mathcal{E}'$, edge attributes tensor E' and node attributes matrix F'
C	Contextual information vector
N	Number of nodes
M	Number of edges
D	Dimension of node attributes
K	Dimension of edge attributes
c	Dimension of contextual information vector
S	Number of translation blocks

all pairs of nodes while the existence of $e_{i,j}$ is reflected by its attributes. $E_0 \in \mathbb{R}^{N\times N\times K}$ is the edge attributes tensor, where $E_{0,i,j} \in \mathbb{R}^{1\times K}$ denotes the edge attributes of edge $e_{i,j}$ and K is the dimension of edge attributes. $F_0 \in \mathbb{R}^{N\times D}$ refers to the node attribute matrix, where $F_{0,i} \in \mathbb{R}^{1\times D}$ is the node attributes of node i and D is the dimension of the node attributes. Similarly, we define the target graph as $G(\mathcal{V}', \mathcal{E}', E', F')$. Note that the target and input graphs are different both in their node attributes as well as edge attributes. The node set of the input and output graph should be the same, namely, $\mathcal{V}_0 = \mathcal{V}'$. Therefore, multi-attributed graph translation is defined as learning a mapping: $\mathcal{T} : G(\mathcal{V}_0, \mathcal{E}_0, E_0, F_0); C \rightarrow G(\mathcal{V}', \mathcal{E}', E', F')$, where $\mathcal{T}$ is a function composition which consists of L sub-functions as $\mathcal{T} = (f_L \circ f_{L-1} \circ \ldots f_0)(E_0, F_0, C) = f_L(f_{L-1}(\ldots f_0(E_0, F_0, C)))$, indicating that the evolution process of the graph transformation. Each composition is formalized as: $f_l : E_l, F_l; C \rightarrow E_{l+1}, F_{l+1}$, where E_l, F_l are the inputs of the composition f_l and E_{l+1}, F_{l+1} are the outputs. We also have $E_L = E'$ and $F_l = F'$. C is a contextual information vector, which is optional and can be any additional environmental information provided to facilitate or constrain the whole translation process. For example, when considering the transformation of IoT, the attackers information can be utilized as a contextual information. Node and edge attributes refer to a vector or matrix that describe each node and edge, including properties and identification. For the initial input and final outputs in graph transformation problem, node and edge attributes (i.e., F_0, F_L, E_0, and E_L) can have physical meaning. For the immediate input and outputs at each round during evolution process, the node and edge attributes (i.e., F_l and E_l, $l \neq 0, L$) are latent representations.

In the problem of multi-attributed graph transformation, there are several unique interactions need to be considered and modeled for each function composition. At each step, we have $f_l(E_l, F_l, C) = \{g_e(E_l, F_l, C), g_f(E_l, F_l, C)\}$, where $g_e(\cdot)$ and $g_f(\cdot)$ are functions that combine the output of the four mapping functions into the final output. Specifically, $g_e(E_l, F_l, C) = g_e(\mathcal{S}_{e\rightarrow e}(E_l), \mathcal{S}_{f\rightarrow e}(F_l), C)$ and $g_f(E_l, F_l, C) = g_f(\mathcal{S}_{f\rightarrow f}(F_l), \mathcal{S}_{e\rightarrow f}(E_l), C)$, where the functions $\mathcal{S}_{e\rightarrow e}(\cdot), \mathcal{S}_{e\rightarrow f}(\cdot), \mathcal{S}_{f\rightarrow e}(\cdot)$, and $\mathcal{S}_{f\rightarrow f}(\cdot)$ model four types of interactions as shown in Fig. 2. The four types of functions are listed as below:

(1) **Edges-to-edges interaction** The edges-to-edges interaction is modeled by function $\mathcal{S}_{e\rightarrow e} : E_l \rightarrow E_{l+1}$, meaning the mapping from the edge attributes in Round l to the edge attributes in Round $l + 1$. Specifically, the edge attributes $E_{l+1,i,j}$ of an edge $e_{i,j}$ in Round $l + 1$ can be influenced by its incident edges' attributes $E_{l,i,k}$ and $E_{l,k,j}$ in

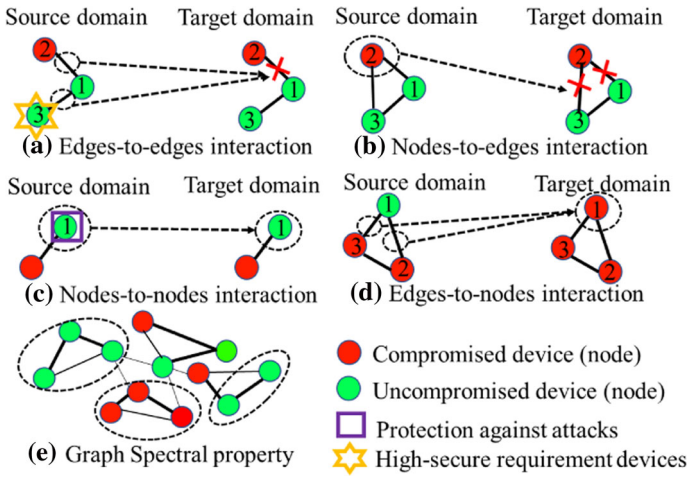


Fig. 2 Five types of interactions during graph translation in the example of malware confinement. Node attributes are indications of malware attacks of IoT devices and edges represent the connections between devices

Round l. For example, in Fig. 2a, if Devices 1 and 3 must be prevented from infection, then the edges between the compromised Device 1 and Device 2 need to be cut, due to the paths among them in input domain.

(2) **Nodes-to-edges interaction** The nodes-to-edges interaction is modeled by function $\mathcal{S}_{f\to e} : F_l \to E_{l+1}$, meaning the mapping from the edge attributes in Round l to the edge attributes in Round $l+1$. Specifically, the attributes $E_{l+1,i,j}$ of edge $e_{i,j}$ in Round $l+1$ can be influenced by its incident nodes' attributes $F_{l,i}$ and $F_{l,j}$ in Round l. As shown in Fig. 2b, if Device 2 is compromised in input domain, then in target domain, only its connections to Devices 1 and 3 need to be removed but the connection between Devices 1 and 3 can be retained because they are not compromised.

(3) **Nodes-to-nodes interaction** The nodes-to-nodes interaction is modeled by function $\mathcal{S}_{f\to f} : F_l \to F_{l+1}$, meaning the mapping from the node attributes in Round l to the node attributes in Round $l+1$. Specifically, For a given node i, its attribute $F_{l,i}$ in Round l input domain may directly influence its attribute $F_{l+1,i}$ in Round $l+1$. As shown in Fig. 2c, Device 3 with effective anti-virus protection (e.g., firewall) may not be easily compromised in target domain.

(4) **Edges-to-nodes interaction** The edges-to-nodes interaction is modeled by function $\mathcal{S}_{e\to f} : E_l \to F_{l+1}$, meaning the mapping from the edge attributes in Round l to the node attributes in Round $l+1$. Specifically, for a given node i, its related edge attributes $E_{l,i,j}$ in Round l may affect its attributes $F_{l+1,i}$ in in Round $l+1$. As shown in Fig. 2d, Device 1 which has more connections with compromised devices in input domain is more likely to be infected in target domain.

(5) **Spectral Graph Property** There exist relationships between nodes and edges in one graph as reflected by the graph spectrum. These relationships are claimed to have some persistent or consistent patterns across input and target domains, which have also been verified in many real-world applications such as brain networks [2]. For example, as shown in Fig. 2e, the devices that are densely connected as a sub-community tend to be in the same node status, which is a shared pattern for relationships between nodes and edges in different domains.

Thus, multi-attributed graph translation should define and model all the above-mentioned functions, including $g_e(\cdot)$, $g_f(\cdot)$, $\mathcal{S}_{e\to e}(\cdot)$, $\mathcal{S}_{e\to f}(\cdot)$, $\mathcal{S}_{f\to e}(\cdot)$, and $\mathcal{S}_{f\to f}(\cdot)$. Traditional graph transformation methods which pre-define simple relationships between the input and target graph (e.g., isomorphism or semantically compatible) and simple rules and operations (e.g., projection) can only cover a limited patterns of transformation, while many real-world transformation patterns are complex. For example, considering the malware confinement case where the nodes refer to IoT devices and the edges reflect the communication links between two devices. The node attributes include the malware-infection status and the properties of that device (i.e., specification and anti-virus software features). A single IoT device (i.e., node) that is compromised has the potential to spread malware infection across the network, eventually compromising the network or even ceasing the network functionality. In contrast, in order to avoid malware spreading as well as maintain the performance of the network, the network connectivity (i.e., graph topology) should be modified through *malware confinement*, thus to change the device status (i.e., node attributes) accordingly. Hence, malware confinement can be considered as predicting the optimal topology as well as the corresponding node and edge attributes of the target graph, where both malware prevention and device performance are maximized.

Thus, our goal is to build a end-to-end framework that can automatically learn and fit the above mentioned functions from massive data by deep neural networks. This is benefited from universal approximation theory such that when the capacity (e.g., number of neurons) in modeling the above functions increases, the approximation errors gradually diminishes. In addition, existing deep learning-based graph transformation also cannot comprehensively handle the above problem due to: (1) Lack of a generic framework to simultaneously characterize and automatically infer all of the above node-edge interactions during translation process. (2) Difficulty in automatically discovering and characterizing the inherent spectral relationship between the nodes and edges in each graph, and ensuring consistent spectral patterns in graphs across input and target domains. (3) All the above interactions could be imposed repeatedly, alternately, and asynchronously during the translation process. It is difficult to discover and characterize such important yet sophisticated process.

4 The proposed method: NEC-DGT

In this section, we propose the Node-Edge Co-evolving Deep Graph Translator (NEC-DGT) to model the multi-attributed graph translation process. First, an introduction of the overall architecture and the loss functions is given. Then, the elaborations of three modules on edge translation, node translation, and graph spectral regularization are presented.

4.1 Overall architecture

4.2 Multi-block asynchronous translation architecture

The proposed NEC-DGT learns the distribution of graphs in the target domain conditioning on the input graphs and contextual information. However, such a translation process from input graph to the final target graph may experience a series of interactions of different types among edges and nodes. Also, such a sophisticated process is hidden and needs to be learned by a sufficiently flexible and powerful model. To address this, we propose the NEC-DGT as shown in Fig. 3. Specifically, the node and edge attributes of input graphs are inputted into the

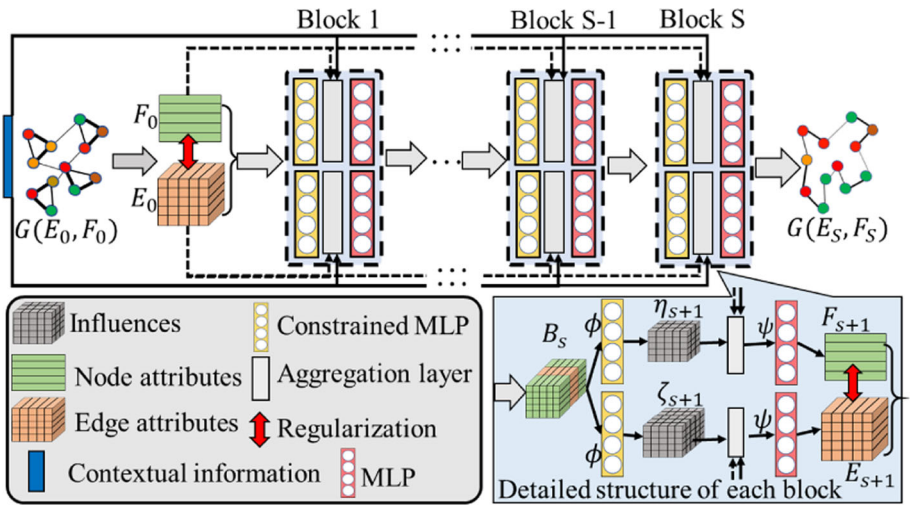


Fig. 3 Proposed NEC-DGT consists of multiple blocks. Each block has edge and node translation paths which are co-evolved and combined by a graph regularization during training process

model and the model output the generated target graphs' node attributes and edge attributes after several blocks. Each block consists of both node and edge translation paths, where an immediate attributed graph will be generated and inputted into the next block. Following this, a sequence of blocks can successfully model the gradual transformation process, as shown in the dotted frame in Fig. 3. The skip-connection architecture (black dotted lines in Fig. 3) implemented across different blocks aims to deal with the asynchrony property of different blocks, which ensures that the final translated results fully utilize various combinations of blocks' information. To train the deep neural network to generate the target graph $G(E', F')$ conditioning on the input graph $G(E_0, F_0)$ and contextual information C, we minimize the loss function as follows:

$$\mathcal{L}_{\mathcal{T}} = \mathcal{L}(\mathcal{T}(G(E_0, F_0), C), G(E', F')) \tag{1}$$

where the nodes set $\mathcal{V}_0$ and $\mathcal{V}'$ as well as edges set $\mathcal{E}_0$ and $\mathcal{E}'$ can be reflected in F_0 and F', as well as E_0 and E'.

4.2.1 Node and edge translation paths

To jointly tackle various interactions among nodes and edges, respective translation paths are proposed for each block. In node translation path (in upper part of detailed structure in Fig. 3), node attributes are generated considering the "nodes-to-nodes" and "edges-to-nodes" interactions. In edge translation path (in lower part of detailed structure in Fig. 3), edge attributes are generated following the "edges-to-edges" and "node-to-edges" interactions.

4.2.2 Spectral graph regularization

To discover and characterize the inherent relationship between nodes and edges of each graph, the frequency-domain properties of the graph is learned, based on which the interactions between node and edge attributes are jointly regularized upon nonparametric graph Laplacian.

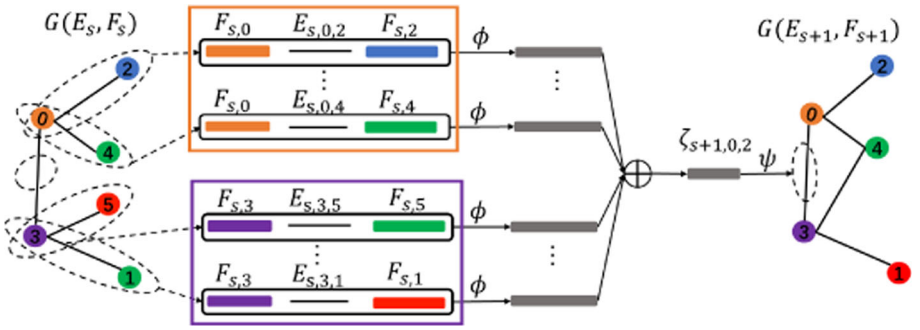


Fig. 4 Details of edge translation path for one edge (i.e., $e_{0,3}$) in a single block

Moreover, to maintain consistent spectral properties throughout the translation process, we enforce the shared patterns among the generated nodes and edges in different blocks by regularizing their relevant parameters in the frequency domain. The regularization of the graphs is formalized as follows:

$$\mathcal{R}(G(E, F)) = \sum_{s=0}^{S} \mathcal{R}_\theta(G(E_s, F_s)) + \mathcal{R}(\theta) \tag{2}$$

where S refers to the number of blocks, and θ refers to the overall parameters in the spectral graph regularization. E_s and F_s refer to the generated edge attributes tensor and node attributes matrix in the sth block. Thus, $G(E_S, F_S)$ is the generated target graph. Then, the final loss function can be summarized as follows:

$$\widetilde{\mathcal{L}} = \mathcal{L}(\mathcal{T}(G(E_0, F_0), C), G(E', F')) + \beta\mathcal{R}(G(E, F)) \tag{3}$$

where β is a nonnegative constant which controls the contribution of the spectral graph regularization in the overall loss function during optimization. The model is trained by minimizing the mean squared error of E_S with E', and F_S with F', enforced by the regularization. Optimization methods (e.g., Stochastic gradient descent (SGD) and Adam) based on back-propagation technique can be utilized to optimize the whole model.

Subsequently, the details of a single translation block are introduced: edge translation path in Sect. 4.3, node translation path in Sect. 4.4 and graph spectral-based regularization in Sect. 4.5.

4.3 Edge translation path

Edge translation path aims to model the nodes-to-edges and edges-to-edges interactions, where edge attributes in the target domain can be influenced by both nodes and edges in the input domain. Therefore, we propose to first jointly embed both node and edge information into influence vectors and then decode it to generate edges attributes. Specifically, the edge translation path of each block contains two functions, *influence-on-edge function* which encodes each pair of edge and node attributes into the influence for generating edges, and the *edge updating function* which aggregates all the influences related to each edge into an integrated influence and decodes this integrated influence to generate each edge' attributes. Figure 4 shows the operation of the two functions in a single block by translating the current input of graph $G(E_s, F_s)$ to output graph $G(E_{s+1}, F_{s+1})$.

4.3.1 Influence-on-edge layers

As shown in Fig. 4, the input graph $G(E_s, F_s)$ is first organized in unit of several pairs of node and edge attributes. For each pair of nodes v and u, we concatenate their edge attributes $E_{s,u,v}$ and their node attributes: $F_{s,u}$ and $F_{s,v}$ as: $B_{s,u,v} = [F_{s,u}, E_{s,u,v}, F_{s,v}]$ (as circled in black rectangles in Fig. 4). Then, $B_{s,u,v} \in \mathbb{R}^{1\times(2D+K)}$ is inputted into the influence-on-edge function: a constrained MLP (Multilayer Perceptron) ϕ which is used to calculate the influence $\phi(B_{s,u,v}) \in \mathbb{R}^{1\times q}$ from the pair of the nodes u and v. q refers to the dimension of the final influence on edges. ϕ for edge translation path is expressed as follows:

$$\begin{aligned} &\phi(X; W_E, b_E) = H_M, \\ &\text{where} \quad H_m = \sigma_m(H_{m-1} * W_E^{(m)} + b_E^{(m)}), m = 0, \ldots, M, H_0 = X \\ &\text{s.t. }, W_{E,1:D}^{(0)} \equiv W_{E,(D+K):(2D+K)}^{(0)}, \end{aligned} \tag{4}$$

where M refers to the number of layers of ϕ, and $W_E^{(m)}$ and $b_E^{(m)}$ are weights and bias for each layer in edge translation path. H_m is the output of the mth layer and input of $(m+1)$th layer. σ_m refers to the activation functions of each layer. For undirected graph, we add a weight constraint $W_{E,1:D}^{(0)} \equiv W_{E,(D+K):(2D+K)}^{(0)}$ to ensure that the influence of $B_{s,u,v}$ is the same as the influence of $B_{s,v,u}$, which means that the first D rows (related to the attributes of node u) and the last D rows (related to the attributes of node v) of $W_E^{(0)}$ are shared. The influence on edges of each pair is computed through the same function with the same weights. Thus, the NEC-DGT can handle various size of graphs.

4.3.2 Edge updating layers

After calculating the influence of each pair of nodes and edge, the next step is to assign each pairs' influences to its related edge to get the integrated influence for each edge (as shown in $\bigoplus$ operation in Fig. 4). This is because each edge is generated depending on both its two related nodes and its incident edges (like the pairs circled in the orange rectangle and purple rectangle related to node 0 and node 3, respectively, in Fig. 4). Here, we define the integrated influence on one edge attribute $E_{s+1,i,j}$ as: $\zeta_{s+1,i,j} \in \mathbb{R}^{1\times q}$, which is computed as follows:

$$\begin{aligned} \zeta_{s+1,i,j} = &\sum\nolimits_{k_1\in N(i)} \phi(B_{s,i,k_1}; W_E, b_E) + \\ &\sum\nolimits_{k_2\in N(j)} \phi(B_{s,k_2,j}; W_E, b_E) \end{aligned} \tag{5}$$

where $N(i)$ refers to the neighbor nodes of node i. Then, the edge attributes $E_{s+1,i,j}$ is generated by $\psi([E_{0,i,j}, \zeta_{s+1,i,j}, C])$, where $E_{0,i,j}$ refers to the input edge attributes of edge $e_{i,j}$. C refers to the contextual information for the translation. The function ψ is implemented by an MLP.

4.3.3 Relationship with other edge convolution networks

Edge convolution network is the most typical method to handle the edge embedding in graphs, which was first introduced as BrainNetCNN [32] and later explored in many studies [21,34, 54]. Our edge translation path is a highly flexible and generic mechanism to handle multi-attributed nodes and edges. Several existing edge convolution layers and their variants can be considered as special cases of our method, as demonstrated in the following theorem.

Theorem 1 *The influence-on-edge function ϕ in edge translation path of NEC-DGT is a generalization of conventional edge convolution neural networks (ECNN).*

Proof Due to $E_{s,(i,j)} \in [1, 0]$, The influence calculated from the effects function in node translation path of NEC-DGT for block s of node i can be summarized as:

$$\bar{I}_{s,i}^{F} = \sum_{j=1}^{d} \phi_{I_F}(E_{s-1,(i,j)}[F_{s-1,i}; 1; F_{s-1,j}]) \tag{6}$$

Then, after the updating function, the node i at stage s can be modeled as: $F_{s,i} = \phi U_F(F_{0,i}; C; \bar{I}_{s,i}^{F})$. $F_{0,i}$ denotes the node attributes from the input graphs.

In ECNN, the message information learned for node i is formalized as:

$$\hat{I}_{s,i}^{F} = \sum_{n=0}^{s-1} \sum_{j=1}^{d} p_{n,(i,j)} F_{0,j} W_n^{s-1} \tag{7}$$

where s denotes the pre-defined number of hop information utilized in ECNN. $W_n \in \mathbb{R}^{d \times F}$ denotes the weights related to nth hop. and F denotes to the dimension of node effects. The $p_{s,(i,j)}$ is the s-hop reachable of two nodes calculated by an iteration: $p_{s+1,(i,j)} = \sum_{k=1}^{d} p_{0,(i,k)} p_{s,(k,j)}$ and $p_{0,(i,j)} = E_{0,(i,j)}$. For ECNN, $F_{s,i} = F_{0,i}$. Due to the iterative nature of n-hop problem, we use Iterative proof to validate that $\bar{I}_{s,i}^{F} = \hat{I}_{s,i}^{F}$ when the parameters of NEC-DGT are trained to meet some requirements.

(1) If $s = 1$, $p_{0,(i,j)} = E_{0,(i,j)}$ and $E_{0,(i,j)} \in [1, 0]$.

For ECNN we got:

$$\hat{I}_{1,i}^{F} = \sum_{j=1}^{d} E_{0,(i,j)} F_{0,j} W_0^0 \tag{8}$$

where $W_0^0 \in \mathbb{R}^{d \times F}$.

For NEC-DGT, since ϕ_{E_O} is implemented by a two-layer MLP. After the first layer, the output can be written as:

$$\bar{I'}_{1,i}^{F} = \sum_{j=1}^{d} E_{0,(i,j)}[F_{0,i}; 1; F_{0,j}]W_1 \tag{9}$$

$$= \sum_{j=1}^{d} E_{0,(i,j)}(F_{0,i} w_{11} + w_{12} + F_{0,j} w_{13}) \tag{10}$$

where $W_1 \in \mathbb{R}^{(2d+1) \times d}$ and can be divided into $w_{11} \in \mathbb{R}^{d \times d}$, $w_{12} \in \mathbb{R}^{1 \times d}$ and $w_{13} \in \mathbb{R}^{d \times d}$. Considering one special situation when w_{11} and w_{12} are learnt as 0 after training, we can get:

$$\bar{I}_{s,i}^{F} = \sum_{j=1}^{d} E_{0,(i,j)} F_{0,j} w_{13} \tag{11}$$

After the second layer of MLP, it can be expressed as:

$$\bar{I}_{s,i}^{F} = \sum_{j=1}^{d} E_{0,(i,j)} F_{0,j} w_{13} W_2 \tag{12}$$

where $W_2 \in \mathbb{R}^{d\times F}$. Thus, if $w_{13}W_2 = W_0^0$, it is proved that $\bar{I}_{1,i}^F = \hat{I}_{1,i}^F$.

(2) Given $\bar{I}_{m,i}^F = \hat{I}_{m,i}^F$ is valid (s=m), we need prove $\bar{I}_{m+1,i}^F = \hat{I}_{m+1,i}^F$, namely to prove:

$$\sum_{j=1}^{d} \phi_{I_F}(E_{m,(i,j)}[F_{m,i}; 1; F_{m,j}]) = \sum_{n=0}^{m}\sum_{j=1}^{d} p_{n,(i,j))} F_{0,j} W_n^m \tag{13}$$

For the right of part of equation 13, we can write it as:

$$right = \sum_{n=1}^{m}\sum_{j=1}^{d} p_{n,(i,j)} F_{0,j} W_n^m + \sum_{j=1}^{d} p_{0,(i,j)} F_{0,j} W_0^m \tag{14}$$

$$= \sum_{n=1}^{m}\sum_{j=1}^{d}\sum_{k=1}^{d} p_{0,(i,k)} p_{n-1,(k,j)} F_{0,j} W_n^m + \sum_{j=1}^{d} p_{0,(i,j)} F_{0,j} W_0^m \tag{15}$$

$$= \sum_{n=1}^{m}\sum_{j=1}^{d}\sum_{k=1}^{d} E_{0,(i,k)} p_{n-1,(k,j)} F_{0,j} W_n^m + \sum_{j=1}^{d} E_{0,(i,j)} F_{0,j} W_0^m \tag{16}$$

For the left part of Equation 13, we can write as:

$$left = \sum_{j=1}^{d} E_{m,(i,j)}[F_{m,i}; 1; F_{m,j}] W_6 \tag{17}$$

$$= \sum_{j=1}^{d} E_{m,(i,j)}(F_{m,i} w_{61} + E_{m,(i,j)} w_{62} + F_{m,j} w_{63}) \tag{18}$$

where $W_6 \in \mathbb{R}^{(2d+1)\times d}$ and can be divided as $w_{61} \in \mathbb{R}^{d\times d}$, $w_{62} \in \mathbb{R}^{1\times d}$ and $w_{63} \in \mathbb{R}^{d\times d}$. When $w_6 1 = \mathbf{0}$ and $w_{62} = \overrightarrow{0}$, we can got:

$$left = \sum_{j=1}^{d} E_{m,(i,j)} F_{m,j} w_{63} \tag{19}$$

$$= \sum_{j=1}^{d} E_{m,(i,j)} \phi_{U_F}(F_{0,j}; \bar{I}_{m,j}^F) w_{63} \tag{20}$$

$$= \sum_{j=1}^{d} E_{m,(i,j)} (F_{0,j} w_{70} + \bar{I}_{m,j}^F w_{71}) w_{63} \tag{21}$$

$$\sum_{j=1}^{d} \phi_{I_F}(E_{m,(i,j)}[F_{m,i}; 1; F_{m,j}]) = \sum_{j=1}^{d} E_{m,(i,j)} (F_{0,j} w_{70} w_{63} + \bar{I}_{m,j}^F w_{71} w_{63}) \tag{22}$$

$$= \sum_{j=1}^{d} E_{m,(i,j)} (F_{0,j} w_{70} w_{63} + \hat{I}_{m,j}^F w_{71} w_{63}) \tag{23}$$

$$= \sum_{j=1}^{d} E_{m,(i,j)} F_{0,j} w_{70} w_{63} + \sum_{n=0}^{m-1}\sum_{j=1}^{d}\sum_{k=1}^{d} E_{m,(i,j)} p_{n,(j,k)} F_{0,k} w_{71} w_{63} W_n^{m-1} \tag{24}$$

Since $E_{m,(i,j)} = E_{0,(i,j)}$, when $w_{70} w_{63} = W_0^m$ and $W_n^{m-1} w_{71} w_{63} = W_{n+1}^m$, it is proved that $\bar{I}_{m+1,i} = \hat{I}_{m+1,i}$. □

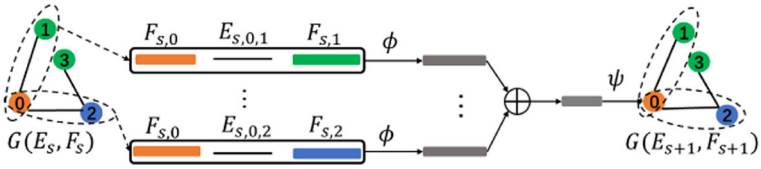


Fig. 5 Details of node translation path for one node (i.e. node 0) in a single block

4.4 Node translation path

Node translation aims to learn the "nodes-to-nodes" and "edges-to-nodes" interactions, where translation of one node's attributes depends on the edge attributes related to this node and its own attributes. The node translation path of each block contains two functions, *influence-on-node function* which learns the influence from each pair of nodes, and *node updating function* which generates the new node attributes by aggregating all the influences from pairs containing this node. Figure 5 shows how to translate a node in a single block.

4.4.1 Influence-on-node layers

As shown in Fig. 5, the input graph $G(E_s, F_s)$ is first organized in the unit of pairs of nodes, where each pair is $B_{s,u,v} \in \mathbb{R}^{1\times(2D+K)}$ which is similar to the edge translation path (as circled in the black rectangle in Fig. 5). Then $B_{s,u,v}$ is inputted into the influence-on-node function, which is implemented by contrained MLP ϕ as Equation (4), to compute the influence $\phi(B_{s,u,v}; W_F, b_F) \in \mathbb{R}^{1\times h}$ to nodes (as shown in the grey bar after ϕ in Fig. 5), where h is the dimension of the influence on nodes.

4.4.2 Node updating layers

After computing the influences of each node pair, the next step is to generate node attributes. For node i, an assignment step is required to aggregate all the influences from pairs containing node i (as shown of $\bigoplus$ operation in Fig. 5). Thus, all the influences for node i are aggregated and input into the updating function, which is implemented by a MLP model ψ to calculate the attributes of node i as: $F_{s+1,i} = \psi([F_{0,i}, \sum_{j\in N(i)} \phi(B_{s,i,j}; W_F, b_F), C])$.

4.5 Graph spectral-based regularization

Based on the edge and node translation path introduced above, we can generate node and edge attributes, respectively. However, since these generated node and edge attributes are predicted separately in different paths, their patterns may not be consistent and harmonic. To ensure the consistency of the edge and node patterns mentioned in Sect. 3, we propose a novel adaptive regularization based on nonparametric graph Laplacian, and a graph frequency regularization.

4.5.1 Non-parametric graph Laplacian regularization

First, we recall the property of the multi-attributed graphs where node information can be smoothed over the graph via some form of explicit graph-based regularization, namely by the well-known graph Laplacian regularization term [33]:

$$F_s^{(d)^T} L_s^{(k)} F_s^{(d)} = \sum_{i,j \in \mathcal{V}} E_{s,i,j}^{(k)} \left\| F_{s,i}^{(d)} - F_{s,j}^{(d)} \right\|^2, \tag{25}$$

where $F_s^{(d)} \in \mathbb{R}^{N \times 1}$ is the node attribute vector for the dth node attribute and $E_s^{(k)} \in \mathbb{R}^{N \times N}$ is the edge attribute matrix for kth attribute generated in the sth block. $L_s^{(k)} = D_s^{(k)} - E_s^{(k)}$ denotes the graph Laplacian for the kth edge attributes matrix. The degree matrix $D_s^{(k)} \in \mathbb{R}^{N \times N}$ is computed as: $D_{s,i,i}^{(k)} = \sum_{j \in N(i)} E_{s,i,j}^{(k)}$.

However, the above traditional graph Laplacian can only impose an absolute smoothness regularization over all the nodes by forcing the neighbor nodes to have similar attribute values, which is often over-restrictive for many situations such as in signed networks and teleconnections. In the real world, the correlation among the nodes is much more complicated than purely "smoothness" but should be a mixed pattern of different types of relations. To address this, we propose an end-to-end framework of nonparametric graph Laplacian which can automatically learn such node correlation patterns inherent in specific types of graphs, with rigorous foundations on spectral graph theory. In essence, we propose the nonparametric graph Laplacian based on the parameter θ as: $g_\theta(\hat{L}_s^{(k)})$. $\hat{L}_s^{(k)}$ is the normalized Laplacian computed as $\hat{L}_s^{(k)} = D_s^{(k)^{-\frac{1}{2}}} L_s^{(k)} D_s^{(k)^{-\frac{1}{2}}}$ and can be diagonalized by the Fourier basis $U_s^{(k)} \in \mathbb{R}^{N \times N}$, such that $\hat{L}_s^{(k)} = U_s^{(k)} \Lambda_s^{(k)} U_s^{(k)^T}$ where $\Lambda_s^{(k)} \in \mathbb{R}^{N \times N}$ is a diagonal matrix storing the graph frequencies. For example, $\Lambda_{s,1}^{(k)}$ is the frequency value of the first Fourier basis $U_{s,1}^{(k)}$. Then, we got $g_\theta(\hat{L}_s^{(k)}) = g_\theta(U_s^{(k)} \Lambda_s^{(k)} U_s^{(k)^T}) = U_s^{(k)} g_\theta(\Lambda_s^{(k)}) U_s^{(k)^T}$. Therefore, we have the regularization as follows:

$$\mathcal{R}_\theta(G(E_s, F_s)) = \sum_{k=1}^{K} \sum_{d=1}^{D} F_s^{(d)^T} U_s^{(k)} g_\theta(\Lambda_s^{(k)}) U_s^{(k)^T} F_s^{(d)} \tag{26}$$

where $g_\theta(\Lambda_s^{(k)})$ is a nonparametric Laplacian eigenvalues that will be introduced subsequently.

4.5.2 Scalable approximation

$g_\theta(\Lambda_s^{(k)})$ is a nonparametric vector whose parameters are all free; It can be defined as: $g_\theta(\Lambda_s^{(k)}) = diag(\theta_s^{(k)})$, where the parameter $\theta_s^{(k)} \in \mathbb{R}^N$ is a vector of Fourier coefficients for a graph. However, optimizing the parametric eigenvalues has the learning complexity of $O(N)$, the dimensionality of the graphs, which is not scalable for large graphs. To reduce the learning complexity of $O(N)$ to $O(\mathbf{1})$, we propose approximating $g_\theta(\Lambda_s^{(k)})$ by a normalized truncated expansion in terms of Chebyshev polynomials [25]. The Chebyshev polynomial $T_p(x)$ of order p may be computed by the stable recurrence relation $T_p(x) = 2xT_{p-1}(x) - T_{p-2}(x)$ with $T_1 = 1$ and $T_2 = x$. The eigenvalues of the approximated Laplacian filter can thus be parametric as the truncated expansion:

$$g_\theta(\Lambda_s^{(k)}) = \sum_{p=1}^{P} \theta_{s,p}^{(k)} T_p(\widetilde{\Lambda}_s^{(k)}) / \sum_{p=1}^{P} \theta_{s,p}^{(k)} \tag{27}$$

for P orders, where $T_p(\widetilde{\Lambda}_s^{(k)}) \in \mathbb{R}^{N \times N}$ is the Chebyshev polynomial of order p evaluated at $\widetilde{\Lambda}_s^{(k)} = 2\Lambda_s^{(k)} / \Lambda_{s,max}^{(k)} - I$, a diagonal matrix of scaled eigenvalues that lie in $[-1, 1]$. The $\Lambda_{s,max}$ refers to the largest element in $\Lambda_s^{(k)}$. $\theta \in \mathbb{R}^{S \times P \times K}$ denotes the parameter tensor for all S blocks. $\theta_{s,p}^{(k)}$ is the pth element of Chebyshev coefficients vector $\theta_s^{(k)} \in \mathbb{R}^P$ for the kth edge attribute. Each $\theta_{s,p}^{(k)}$ is normalized by dividing the sum of all the coefficients in $\theta_s^{(k)}$ to

avoid the situation where $\theta_s^{(k)}$ is trained as zero. Thus, the laplacian computation can then be written as $g_\theta(\widetilde{L}_s^{(k)}) = \sum_{p=1}^{P} \theta_{s,p}^{(k)} T_p(\widetilde{L}_s^{(k)}) / \sum_{p=1}^{P} \theta_{s,p}^{(k)}$, where $T_p(\widetilde{L}_s^{(k)}) \in \mathbb{R}^{N\times N}$ is the Chebyshev polynomial of order p evaluated at the scaled Laplacian $\widetilde{L}_s^{(k)} = 2\hat{L}_s^{(k)} / \Lambda_{s,max}^{(k)} - I$. For efficient computation, we further approximate $\Lambda_{s,max}^{(k)} \approx 1.5$, as we can expect that the neural network parameters θ will adapt to this change in scale during training.

4.5.3 Graph frequency regularization

To ensure that the spectral graph patterns are consistent throughout the translation process across different blocks, we utilize a graph frequency regularization to not only maintain the similarity but also allow the exclusive properties of each block's patterns to be reserved to some degree. Specifically, regarding all the frequency pattern basis of form $\widetilde{L}$, some are important in modeling the relationships between nodes and graphs while some are not, resulting in the sparsity pattern of θ. Thus, inspired by the multi-task learning, we learn the consistent sparsity pattern of θ_s by using the $L_{2,1}$ norm as regularization:

$$\mathcal{R}(\theta) = \sum_{k=1}^{K} \sum_{s=1}^{S} \sqrt{\sum_{p=1}^{P} {\theta_{s,p}^{(k)}}^2} \tag{28}$$

4.6 Extensions of graph spectral-based regularization

In this section, we extend our framework of multi-attributed graph translation into broadly signed graph and directed graphs, which makes our model more general to various graph types. The node and edge translation paths do not need to change since the trainable parameters can be adaptive in dealing with various edge and node attributes. However, apart from the two translation paths, the spectral-based regularization term cannot be utilized without change for signed or directed graphs. This is incurred by several technical challenges: the Laplacian matrix for both sign and directed graph are different from the undirected graphs. The way to calculate the Laplacian matrix which should be symmetric and positive semi-definite needs to be customized for both signed and direct graphs. Thus, the way to approximate the Laplacian needs also changes accordingly. Thus, we explore how to fit our model into the signed and directed graph translation, respectively, by proposing the signed spectral-based graph regularization and directed spectral-based graph regularization.

4.6.1 Extension to signed graphs

Signed graphs are defined as the weighted graphs in which negative and positive entries are allowed, the intuition being that a negative weight indicates dissimilarity or distance. Thus, for a signed graph, the degree matrix may contain zero or negative entries (thus $D_s^{(k)^{-\frac{1}{2}}}$ may not exist), and the Laplacian $L_s^{(k)}$ may no longer be positive semi-definite. However, the spectral-based graph regularization requires the Laplacian to be positive semi-definite for representing the relations between the node and edge attributes.

Thus, we adopt a definition of the signed graph Laplacian [17]. First, the degree matrix is computed as $\bar{D}_{s,i,i}^{(k)} = \sum_{j\in N(i)} \left| E_{s,i,j}^{(k)} \right|$, and then the normalized version of the Laplacian for the signed graph is $L_s^{(k)} = I - \bar{D}_s^{(k)^{-\frac{1}{2}}} E_s^{(k)} \bar{D}_s^{(k)^{-\frac{1}{2}}}$. In this way, the signed graph Laplacian can be positive semi-definite and decomposed with eigenvalues and eigenvectors. Thus, we

can fit our proposed nonparametric regularization and scalable approximation into the signed graph as

$$\mathcal{R}_\theta(G(E_s, F_s)) = \sum_{k=1}^{K} \sum_{d=1}^{D} F_s^{(d)T} \sum_{p=1}^{P} \theta_{s,p}^{(k)} T_p(\widetilde{L}_s^{(k)}) F_s^{(d)} / \sum_{p=1}^{P} \theta_{s,p}^{(k)}, \quad (29)$$

where $\widetilde{L}_s^{(k)} = 2L_s^{(k)}/\Lambda_{s,max}^{(k)} - I$. Thus, the signed spectral-based graph regularization can enjoy a constant time complexity $O(1)$.

4.6.2 Extension to directed graphs

For directed graphs, the major problem is that the edge attribute matrix is not symmetric, and the way to calculate the Laplacian matrix described above cannot semantically enforce the relations between edges and nodes. Here, we calculate the Laplacian of the directed graph as a Hermitian matrix by using the transition probability matrix [9].

First, we use $E_{s,i,j}$ to denote the sth edge attributes of the edge from the ith node to the jth node. Thus, for a weighted directed graph with edge weights $E_{s,i,j} > 0$, each element of a general transition probability matrix P_s regarding the sth attribute can be defined as

$$P_{s,i,j} = \frac{E_{s,i,j}}{\sum_k E_{s,i,k}}. \quad (30)$$

The Perron–Frobenius theorem [27] states that an irreducible matrix with nonnegative entries has a unique (left) eigenvector with all entries positive. Let ρ_s denote the eigenvalue of the positive eigenvector of P_s. Namely, the transition probability matrix P_s of a strongly connected directed graph has a unique left eigenvector ϕ_s with $\phi_{s,i} > 0$ for all nodes i, and $\phi_s P_s = \rho_s \phi_s$. We can normalize and choose ϕ_s to satisfy $\sum_i \phi_{s,i} = 1$. Now, the ϕ_s is called the Perron vector of P_s. Then, the Laplacian of a directed graph is defined by

$$L_s = I - \frac{\Phi_s^{\frac{1}{2}} P_s \Phi_s^{\frac{-1}{2}} + \Phi_s^{\frac{-1}{2}} P_s^* \Phi_s^{\frac{1}{2}}}{2}, \quad (31)$$

where Φ_s is a diagonal matrix with entries $\Phi_{s,i,i} = \phi_{s,i}$, and P_s^* denotes the conjugated transpose of P_s. Next, we use the nonparametric approximation to define the spectral-based regularization. As mentioned in Sect. 4.5, we need to approximate the eigenvalues $g_\theta(\Lambda_s)$ of the original graph Laplacian. This means the decomposition of P_s to calculate Φ_s and the decomposition of L_s to calculate U_s (U_s is the eigenvector for L_s, as mentioned in Sect. 4.5) are both needed. However, it will lead to intensive computation, especially for large graphs. Thus, to avoid decomposition, the nonparametric method is used to approximate $g_\theta(L_s)$ for directed graphs as

$$g_\theta(L_s) = g_\theta(I - \frac{\Phi_s^{\frac{1}{2}} P_s \Phi_s^{\frac{-1}{2}} + \Phi_s^{\frac{-1}{2}} P_s^* \Phi_s^{\frac{1}{2}}}{2}) \quad (32)$$

$$= (I - \frac{g_\theta(\Phi_s)^{\frac{1}{2}} P_s g_\theta(\Phi_s)^{\frac{-1}{2}} + g_\theta(\Phi_s)^{\frac{-1}{2}} P_s^* g_\theta(\Phi_s)^{\frac{1}{2}}}{2}), \quad (33)$$

It is easy to find that approximating L_s is equal to approximating Φ_s. It should be noted that this operation also reduces the computational effort that is spent on getting Φ_s by decomposing the transition matrix P_s, because otherwise we need to use a polynomial-time algorithm to calculate the exact Φ_s computationally since there is no closed form solution for Φ_s [40]. Now, we approximate Φ_s by learning $g_\theta(\Phi_s) = diag(\theta_1, \ldots, \theta_N)$ with the complexity $O(N)$.

4.7 Complexity analysis

The proposed NEC-DGT requires $O(N^2)$ operations in time complexity and $O(N^2)$ space complexity in terms of number of nodes in the graph. It is more scalable than most of the graph generation methods. For example, GraphVAE [52] requires $O(N^4)$ operations in the worst case and Li et al [38] uses graph neural networks to perform a form of message passing with $O(MN^2)$ operations to generate a graph.

5 Experiments

In this section, we present both the quantitative and qualitative experiment results on NEC-DGT as well as the comparison models. All experiments are conducted on a 64-bit machine with Nvidia GPU (GTX 1070, 1683 MHz, 8 GB GDDR5). The model is trained by Adam optimization algorithm.[2]

5.1 Experimental setup

5.1.1 Datasets

We performed experiments on four synthetic datasets and four real-world datasets with different graph sizes and characteristics. All the dataset contain input-target pairs.

Synthetic dataset Four datasets are generated based on different types of graphs and translation rules. The input graphs of the first three datasets (named as Syn-I, Syn-II, and Syn-III) are Erdos-Renyi (E-R) graphs generated by the Erdos Renyi model [15] with the edge probability of 0.2 and graph size of 20, 40, and 60, respectively. The target graph topology is the 2-hop connection of the input graph, where each edge in the target graph refers to the 2-hop reachability in the input graph (e.g., if node i is 2-hop reachable to node j in the input graph, then they are connected in the target graph). The input graphs of the fourth dataset (named as Syn-IV) are Barabási-Albert (B-A) graphs generated by the Barabási-Albert model [3] with 20 nodes, where each node is connected to 1 existing node. In Syn-IV, topology of target graph is the 3-hop connection of the input graph. For all the four datasets, the edge attributes $E_{s,i,j} \in [0, 1]$ denotes the existence of the edge. For both input and target graphs, the node attributes are continuous values computed following the polynomial function: $f(x) : y = ax^2 + bx + c(a = 0, b = 1, c = 5)$, where x is the node degree and $f(x)$ is the node attribute. Each dataset is divided into two subsets, each of which has 250 pairs of graphs. Validation is conducted where one subset is used for training and another for testing, and then exchange them for another validation. The average result of the two validations is regarded as the final result.

Malware confinement dataset Malware dataset are used for measuring the performance of NEC-DGT for malware confinement prediction. There are three sets of IoT nodes at different amount (20, 40 and 60) encompassing temperature sensors connected with Intel ATLASEDGE Board and Beagle Boards (BeagleBone Blue), communicating via Bluetooth protocol. Benign and malware activities are executed on these devices to generate the initial attacked networks as the input graphs. Benign activities include MiBench [23] and SPEC2006 [26], Linux system programs, and word processor. The nodes represent devices

[2] The code of the model and additional experiment results are available at:https://github.com/xguo7/NEC-DGT.

and node attribute is a binary value referring to whether the device is compromised or not. Edge represents the connection of two devices and the edge attribute is a continuous value reflecting the distance of two devices. The real target graphs are generated by the classical malware confinement methods: stochastic controlling with malware detection [46,49,50]. We collected 334 pairs of input and target graphs with different contextual parameters (infection rate, recovery rate, and decay rate) for each of the three datasets. Each dataset is divided into two subsets: one has 200 pairs and another has 134 pairs. The validation is conducted in the same way as the synthetic dataset.

Molecule reaction dataset We apply our NEC-DGT to one of the fundamental problems in organic chemistry, thus predicting the product (target graph) of chemical reaction given the reactant (input graph). Each molecular graph consists of atoms as nodes and bond as edges. The input molecule graph has multiple connected components since there are multiple molecules comprising the reactants. The reactions used for training are atom-mapped so that each atom in the product graph has a unique corresponding atom in the reactants. We used reactions from USPTO granted patents, collected by Lowe [41]. We obtained a set of 5,000 reactions (reactant-product pair) and divided them into 2,500 and 2,500 for training and testing. Atom (node) features include its elemental identity, degree of connectivity, number of attached hydrogen atoms, implicit valence, and aromaticity. Bond (edge) features include bond type (single, double, triple, or aromatic), and whether it is connected.

Real-world HCP dataset The human connectome project (HCP) dataset is used for evaluating the signed graph translation model. Brain network prediction, such as prediction of functional connectivity based on structural connectivity, is a very critical task in neuroscience. The goal is to learn the mapping from the resting-state functional connectivity into task-specific functional connectivity in the human brain. In this dataset, the source and the target graphs respectively reflect the structural connectivity (SC) and the functional connectivity (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 HCP dataset [56]. By following the preprocessing procedure in [57], the SC data is constructed by applying probabilistic tracking on the diffusion MRI data using the Probtrackx tool from the FMRIB Software Library [28] with 68 predefined regions of interest (ROIs). Then, the edge attributes of FC are defined as the Pearson's correlation between two ROIs blood oxygen level-dependent time obtained from the resting-state functional MRI data. The node attributes refer to the index of each node by a one-hot vector. Since the edges of FC can be either positive or negative, signed graph translation model is needed to handle this task. In total, 823 pairs of SC and FC samples are used and fivefold cross-validation is performed.

Online breast cancer community dataset We adopt the dataset from Gao [19] for validation of the proposed di-NEC-DGT. The data is collected through the Breast Cancer Community,[3] which is one of the largest online forums designed for patients to share information related to breast cancer. The forum data collected for this study covers an 8-year period from the beginning of 2010 to the end of 2017. There are 80 sub-forums, such as "Not Diagnosed But Worried" and "Breast Reconstruction". The user sub-forum activity transition is defined as being when the users posted new topics or replied to existing topics and the time window was set as one month. After removing common words and stop words, 59 top-frequency keywords from the forum content construct the feature vectors for the sub-forums. Also, each user may come across different health stages. The health stages consists of "Dx," "Chemotherapy," "Targeted," "Hormonal," "Radiation," "Surgery,". Each transition graph of a user reflects the stage of the user, and it is obvious that the graph can change

[3] https://community.breastcancer.org/.

as the user transfers to another health stage since the things that concern them change. Our goal is to predict how the user transitions across subforums when their health stage changes from the current to the next, given the current transition graph. We randomly selected 70% of users who provided their health stage history for training, another 10% for validation, and the remaining 20% for testing. The predicted transition graphs were validated against the real transition graphs in the target stage. We treat "Dx" as the initial input stage and treat the others as the target stages.

5.1.2 Comparison methods

Since there is no existing method handling the multi-attributed graph translation problem, NEC-DGT is compared with two categories of methods: (1) graph topology generation methods, and (2) graph node attributes prediction methods.

Graph topology generation methods (1) GraphRNN [63] is a recent graph generation method based on sequential generation with LSTM model; (2) Graph Variational Auto-encoder (GraphVAE) [52] is a VAE-based graph generation method for small graphs; (3) Graph Translation-Generative Adversarial Networks (GT-GAN) [21] is a new graph topology translation method based on graph generative adversarial network.

Node attributes prediction methods (1) Interaction Network (IN) [4] is a node state updating network considering the interaction of neighboring nodes; (2) DCRNN [39] is a node attribute prediction network for tranffic flow prediction; (3) Spatio-Temporal Graph Convolutional Networks (STGCN) [64] is a node attribute prediction model for traffic speed forecast.

Furthermore, to validate the effectiveness of the graph spectral-based regularization, we conduct a comparison model (named as NR-DGT) which has the same architecture of NEC-DGT but without the graph regularization.

5.1.3 Evaluation metrics

A set of metrics are used to measure the similarity between the generated and real target graphs in terms of node and edge attributes. To measure the attributes which are Boolean values, the Acc (accuracy) is utilized to evaluate the ratio of nodes or edges that are correctly predicted among all the nodes or possible node pairs. To measure the attributes which are continuous values, MSE (mean squared error), R2 (coefficient of determination score), Pearson and Spearman correlation are computed between attributes of generated and real target graphs. $N- < metric >$ represents metrics evaluated on node attributes and $E- < metric >$ represents metrics evaluated on edge attributes.

5.2 Performance

5.2.1 Metric-based evaluation for synthetic graphs

For synthetic datasets, we compare the generated and real target graphs on various metrics and visualize the patterns captured in the generated graphs. Table 2 summarizes the effectiveness comparison for four synthetic datasets. The node attributes are continuous values evaluated by N-MSE, N-R2, N-P, and N-SP. The edge attributes are binary values evaluated by the accuracy of the correctly predicted edges. The results in Table 2 demonstrate that the proposed NEC-DGT outperforms other methods in both node and edge attributes prediction and is the

Table 2 Evaluation of generated target graphs for synthetic dataset (N for node attributes, E for edge attributes, P for Pearson correlation, SP for Spearman correlation and Acc for accuracy)

dataset	Method	N-MSE	N-R2	N-P	N-Sp	Method	E-Acc
Syn-I	IN	5.97	0.06	0.48	0.44	GraphRNN	0.6212
	DCRNN	51.36	0.12	0.44	0.45	GraphVAE	0.6591
	STGCN	15.44	0.19	0.42	0.56	GT-GAN	0.7039
	NR-DGT	2.13	**0.87**	0.90	0.89	NR-DGT	0.7017
	NEC-DGT	**1.98**	0.76	**0.93**	**0.91**	NEC-DGT	**0.7129**
Syn-II	IN	**1.36**	0.85	0.77	0.87	GraphRNN	0.5621
	DCRNN	71.07	0.11	0.39	0.37	GraphVAE	0.4639
	STGCN	33.11	0.21	0.15	0.15	GT-GAN	0.7005
	NR-DGT	1.43	0.91	0.94	**0.97**	NR-DGT	0.7016
	NEC-DGT	1.91	**0.93**	**0.97**	**0.97**	NEC-DGT	**0.7203**
Syn-III	IN	35.46	0.31	0.59	0.56	GraphRNN	0.4528
	DCRNN	263.23	0.09	0.41	0.39	GraphVAE	0.3702
	STGCN	43.34	0.22	0.48	0.47	GT-GAN	0.5770
	NR-DGT	5.90	0.90	0.94	0.92	NR-DGT	0.6259
	NEC-DGT	**4.56**	**0.93**	**0.97**	**0.96**	NEC-DGT	**0.6588**
Syn-IV	IN	4.63	0.10	0.53	0.51	GraphRNN	0.5172
	DCRNN	63.03	0.12	0.22	0.16	GraphVAE	0.3001
	STGCN	6.52	0.08	0.11	0.10	GT-GAN	0.8052
	NR-DGT	4.49	0.12	0.55	0.54	NR-DGT	0.6704
	NEC-DGT	**1.86**	**0.73**	**0.93**	**0.89**	NEC-DGT	**0.8437**

only method to handle both. Specifically, in terms of node attributes, the proposed NEC-DGT get smaller N-MSE value than all the node attributes prediction methods by 85%, 71%, 95% and 95% on average for four dataset, respectively. Also, NEC-DGT outperforms the other methods by 46%, 36%, 44% and 58% on average for four dataset, respectively, on N-R2, N-P, and N-SP. This is because all the node prediction methods only consider a fixed graph topology, while NEC-DGT allows the edges to vary. In terms of edges, the proposed NEC-DGT get the highest E-ACC than all the other graph generation methods. It also has higher E-ACC than graph topology translation method: GT-GAN by 7% on average since NEC-DGT considers both edge and node attributes in learning the translation mapping while GT-GAN only considers edges. The proposed NEC-DGT outperforms the NR-DTG by around 3% on average in terms of all metrics, which demonstrates the effectiveness of the graph spectral-based regularization.

5.2.2 Evaluation of the learned translation mapping for synthetic graphs

To evaluate whether the inherent relationship between node and edge (reflected by node degree) attributes is learned and maintained by NEC-DGT, we draw the distributions of the node attribute versus node degree of each node in the generated graphs to visualize their relationship. For comparison, a ground-truth correlation is drawn according to the predefined rule of generating the dataset, namely each node's degree and attribute follows the function $y = x + 5$. Figure 6 shows four example distributions of nodes in terms of node attributes

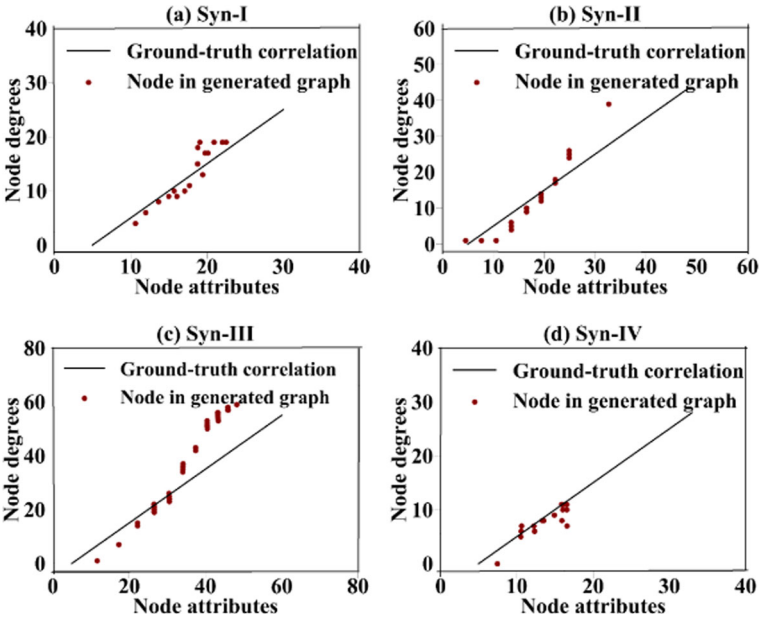


Fig. 6 Relation visualizations between node attributes and node degrees for samples from four synthetic graphs

and degree with the black line as ground-truth. As shown in Fig. 6, the nodes are located closely on the ground-truth, especially for the syn-I and syn-IV, where around 85% nodes are correctly located. This is largely because the proposed graph spectral-based regularization successfully discovers the patterns: the densely connected nodes all tend to have large node attributes and in reverse.

To explore the performance of the proposed methods in dealing with graphs with different densities, 7 subsets of input–output synthetic graph pairs are generated for validations, where the graph density of input graphs ranges from 0.1 to 0.9. Figure 7 plots the relationship between graph density and the different evaluation metrics. As shown in Fig. 7, when the graph density ranges from 0.2 to 0.8, the proposed model is robustness in handling graphs without varying densities, namely both dense or sparse graphs. However, when the density of input graph increases to 0.9, namely the density of target graph is almost 1, it is difficult for the model to learn a valid pattern from a fully connected graph.

5.2.3 Metric-based evaluation for malware datasets

Table 3 shows the evaluation of NEC-DGT by comparing the generated and real target graphs. For malware graphs, the node attributes are evaluated by N-ACC by calculating the percentage of nodes whose attributes are correctly predicted in all nodes. The edge attributes are continuous value evaluated by E-MSE, E-R2 and E-P. We also use E-Acc to evaluate the correct existence of edges among all pairs of nodes. The results in Table 3 demonstrates that NEC-DGT performs the best for all the three datasets. In terms of E-Acc, the graph generation methods (GraphRNN and GraphVAE) cannot handle the graph translation work and got low E-Acc of around 0.6 at Mal-I,Mal-II, and 0.8 at Mal-III. GT-GAN achieves high E-ACC, but its E-MSE is about 2 folds larger than that of the proposed NEC-DGT on average. NEC-DGT successfully handle the translation tasks with high E-Acc above 0.9, and

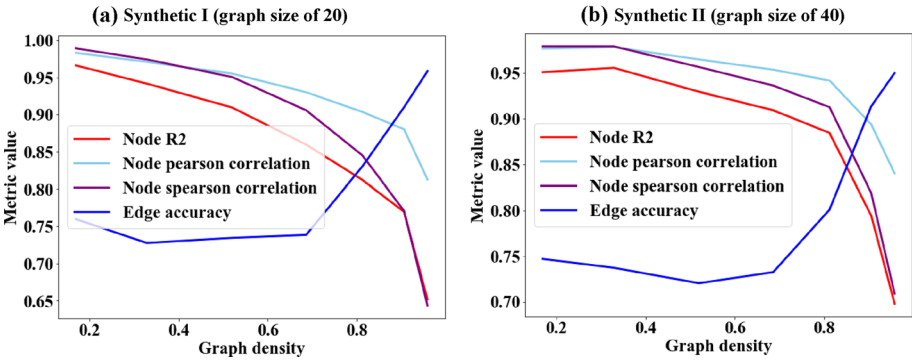


Fig. 7 Transformation performance when dealing with input graphs with different densities regarding two synthetic dataset: **a** the synthetic-I dataset and **b** the synthetic-II dataset

Table 3 Evaluation of generated target graphs for Malware dataset (N for node attributes, E for edge attributes, P for Pearson correlation, SP for Spearman correlation and Acc for accuracy)

Method	E-Acc	E-MSE	E-R2	E-P	Method	N-Acc
Malware-I						
GraphRNN	0.6107	1831.43	0.52	0.00	IN	0.8786
GraphVAE	0.5064	2453.61	0.00	0.04	DCRNN	0.8786
GT-GAN	0.6300	1718.02	0.42	0.11	STGCN	0.9232
NR-DGT	0.9107	668.57	**0.82**	**0.91**	NR-DGT	0.9108
NEC-DGT	**0.9218**	**239.79**	0.78	**0.91**	NEC-DGT	**0.9295**
Malware-II						
GraphRNN	0.7054	1950.46	0.44	0.29	IN	0.8828
GraphVAE	0.6060	2410.57	0.73	0.16	DCRNN	0.8790
GT-GAN	0.9033	462.73	0.13	0.81	STGCN	0.9330
NR-DGT	0.9117	448.48	0.68	0.83	NR-DGT	0.8853
NEC-DGT	**0.9380**	**244.40**	**0.81**	**0.91**	NEC-DGT	**0.9340**
Malware-III						
GraphRNN	0.8397	1775.58	0.16	0.23	IN	0.8738
GraphVAE	0.8119	2109.64	0.39	0.32	DCRNN	0.8738
GT-GAN	0.9453	550.30	0.63	0.80	STGCN	**0.9375**
NR-DGT	0.9543	341.10	0.76	0.88	NR-DGT	0.8773
NEC-DGT	**0.9604**	**273.67**	**0.81**	**0.90**	NEC-DGT	0.9002

the smallest E-MSE. In terms of N-Acc, NEC-DGT outperforms other methods by around 5% on the first two datasets. In summary, the proposed NEC-DGT cannot only jointly predict the node and edges attributes, but also performs the best in most of metrics. The superiority of NEC-DGT over the NR-DGT in terms of E-MSE demonstrates that the graph spectral-based regularization indeed improve modeling translation mapping.

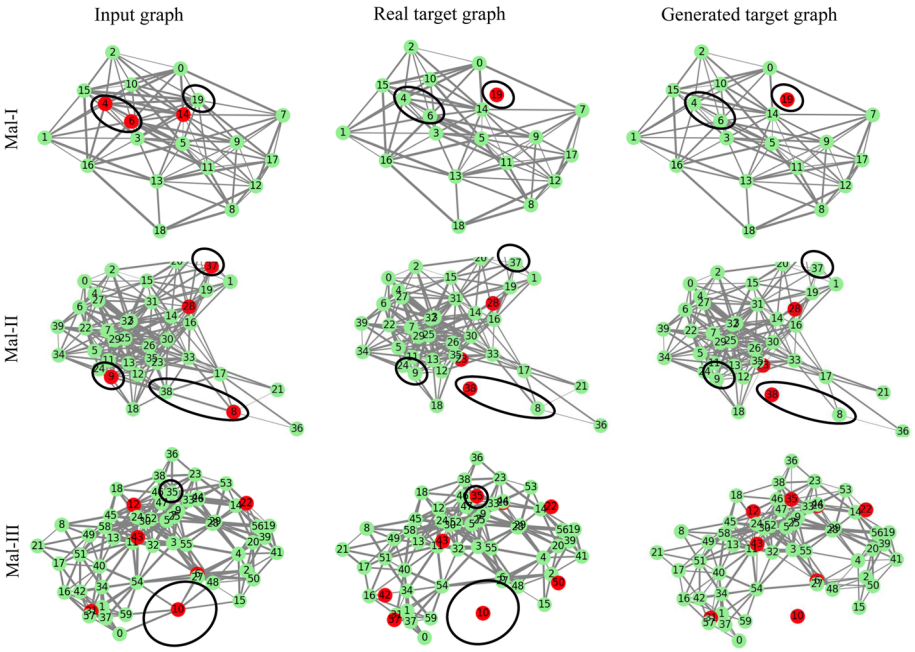


Fig. 8 Cases of Malware translation by NEC-DGT

5.2.4 Case study for Malware dataset

Figure 8 investigates three cases of input, real target and generated target graph by NEC-DGT. The green nodes refer to the uncompromised devices, while the red nodes refer to the compromised devices. The width of each edge reflects the distance between two devices. In the first case, both in generated and real target graphs, Devices 4 and 6 are restored to normal, while Device 19 get attacked and is isolated from the other devices. It validates that our NEC-DGT successfully finds the rules of translating nodes and performs like the true confinement process. In the second case, Device 8 propagates the malware to Device 38, which is also modeled by NEC-DGT in generated graphs. In addition, the NEC-DGT not only correctly predicts the nodes attributes, but also discovers the change in edge attributes, e.g., in the third case, most of the connections of compromised Device 10 were cut both in generated and real target graphs.

5.2.5 Metric-based evaluation for molecule reaction datasets

In this task, the NEC-DGT is compared to the Weisfeiler–Lehman Difference Network (WLDN) [29], which is a graph learning model specially for reaction prediction. Table 4 shows the performance of our NEC-DGT on the reaction dataset on five metrics, which are the same with the synthetic datasets. The proposed NEC-DGT outperforms both the translation model GT-GAN and the WLDN by 5% on average. Though the atoms do not change during reaction, we evaluate the capacity of our NEC-DGT to copy the input node features. As shown in Table 4, The NEC-DGT get the smallest N-MSE and get higher N-R2 than other comparison methods by around 18%. This shows that our NEC-DGT can deal with

Table 4 Evaluation of generated target graphs for molecule dataset: N for node attributes, E for edge attributes

Method	N-MSE	N-R2	N-P	N-Sp	Method	E-Acc
IN	0.0805	0.46	0.13	0.12	GT-GAN	0.8687
STGCN	0.0006	0.98	**0.99**	0.97	WLDN	0.9667
NR-DGT	0.0008	0.97	**0.99**	**0.99**	NR-DGT	0.9918
NEC-DGT	**0.0004**	**0.99**	**0.99**	**0.99**	NEC-DGT	**0.9925**

Table 5 Pearson correlation between the predicted graph and empirical graph on HCP datasets (Res for Resting, Emo for Emotion, Gam for Gambling, Re for Relational)

Method	Res	Emo	Gam	Lang	Motor	Re	Social	WM
Ganlan2008	0.41	0.42	0.44	0.44	0.45	0.44	0.45	0.45
Abdelnour2018	0.40	0.41	0.43	0.43	0.44	0.43	0.44	0.45
Sign-NEC-DGT(NR)	0.42	**0.43**	0.44	**0.45**	**0.46**	**0.45**	0.45	**0.46**
Sign-NEC-DGT	**0.43**	**0.43**	**0.46**	**0.45**	0.45	**0.45**	**0.46**	**0.46**

a wide range of real-world applications, whether the edges and nodes need change or keep stable.

5.2.6 Metric-based evaluation for HCP datasets

We consider two classic brain network prediction methods that use SC to FC [2,16] as the comparison methods for this experiment. Abdelnour et al. [2] considered the graph spectral transformation kernels by assuming that SC and FC share the identical eigenvectors on their Laplacians. Another method directly considers the graph translation between SC and FC. The goal is to predict the FC given the SC when the brain is doing different tasks.

Table 5 shows the Pearson coefficient by comparing the predicted graphs with the empirical target graphs. Our method achieves the highest Pearson coefficient on seven out of eight datasets with superiority of 3.8%, and the highest average Pearson correlation. Specifically, the proposed sign-NEC-DGT outperformed the comparison methods by 5.8% on average in the resting task and 5.4% in Gambling tasks. Also, the proposed sign-NEC-DGT has a 1.3% higher Pearson coefficient than the one without regularization, which demonstrates the effectiveness of the proposed signed graph spectral regularization. This superiority is mainly because the proposed sign-NEC-DGT has the most freedom to learn different parameters for dealing with positive and negative edge attributes.

5.2.7 Case study for HCP dataset

Figure 9 plots two subjects: (1) structural connectivity (i.e., the adjacent matrix of the source graph shown on the left column), (2) empirical functional connectivity (i.e., the adjacent matrix of the target graph shown on the middle column), and (3) predicted functional connectivity (i.e., the adjacent matrix of the target graph shown on the right column). As shown in Fig. 9, the predicted FC using Subject 87s SC is very close to the same subject's empirical FC. On the other hand, the predicted FC using Subject 71s SC is different from Subject 87s empirical FC, although Subject 87s SC is very similar to Subject 71s SC. This is because

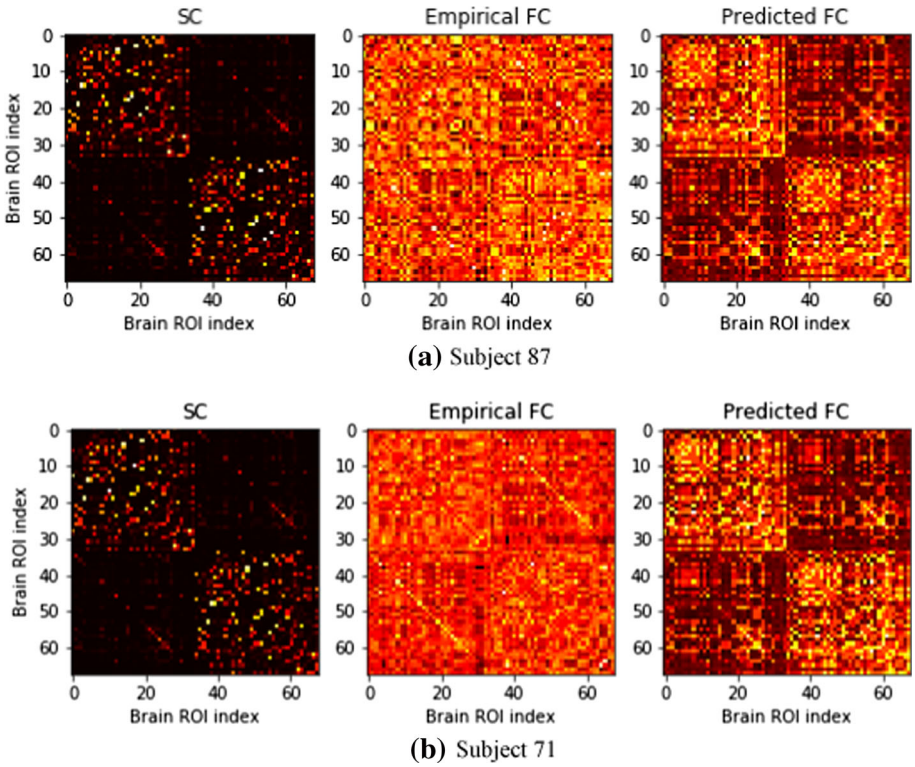


Fig. 9 Cases of FC prediction by sign-NEC-DGT

SC reflects the human brains anatomical neural network, which has relatively fewer individual differences among human beings. Unlike SC, the FC used in this dataset reflects the Pearson correlations between two time series (i.e., Blood Oxygen Level Dependent (BOLD) signal) of different brain regions of interest (ROIs), when the subject is instructed under the resting-state. Practically, it is difficult to control these subjects brain activities, which causes the empirical FC to be very noisy such that it may affect the performance of all prediction methods.

5.2.8 Metric-based evaluation for breast cancer community dataset

In this task, the goal is to predict the direct transition network of a user at one health stage given the transition network at another health stage (Dx). Thus, there are five tasks relating to five different target stages. Due to the requirement of handling the directed graphs, we use two comparison methods: GT-GAN [21] and IN [4]. Table 6 shows the metric-based evaluation results for this task.

As shown in Table 6, the proposed di-NEC-DGT is the only method that not only can handle the node attribute prediction but also can predict the edge attributes for the directed graph. More importantly, it achieves a better performance than the comparison methods for all the different tasks. Specifically, for node prediction tasks, the proposed di-NEC-DGT achieves the a better performance than IN on four metrics on average by about 47.9% in the Chemotherapy task, 44.8% in the Radiation task, 45.4% in the Hormonal task, 45.3% in the

Table 6 Evaluation of generated target graphs for breast cancer dataset: N for node attributes, E for edge attributes, Chem for Chemotherapy, Ra for Radiation, Hor for Hormonal, Sur for Surgery, Tar for Targeted

Task	Method	N-MSE	N-R2	N-P	N-Acc	Method	E-Acc
Chem	IN	0.0666	0.31	0.72	0.9167	GT-GAN	**0.9993**
	di-NEC-DGT(NR)	0.0068	0.93	**0.98**	0.9991	di-NEC-DGT(NR)	0.9988
	di-NEC-DGT	**0.0060**	**0.94**	**0.98**	**0.9995**	di-NEC-DGT	0.9988
Ra	IN	0.0667	0.31	0.72	0.9167	GT-GAN	0.9953
	di-NEC-DGT(NR)	0.0076	**0.92**	0.97	**0.9989**	di-NEC-DGT(NR)	0.9981
	di-NEC-DGT	**0.0073**	**0.92**	**0.98**	0.9985	di-NEC-DGT	**0.9982**
Hor	IN	0.0661	0.32	0.73	0.9177	GT-GAN	0.9945
	di-NEC-DGT(NR)	0.0078	0.92	0.97	0.9983	di-NEC-DGT(NR)	**0.9973**
	di-NEC-DGT	**0.0061**	**0.94**	**0.98**	**0.9996**	dir-NEC-DGT	**0.9973**
Sur	IN	0.0661	0.32	0.73	0.9175	GT-GAN	**0.9981**
	di-NEC-DGT(NR)	0.0066	0.93	**0.98**	**0.9992**	dir-NEC-DGT(NR)	**0.9981**
	di-NEC-DGT	**0.0064**	**0.94**	**0.98**	**0.9992**	dir-NEC-DGT	**0.9981**
Tar	IN	0.0671	0.31	0.71	0.9158	GT-GAN	0.9986
	di-NEC-DGT(NR)	0.0205	0.79	0.90	0.9866	di-NEC-DGT(NR)	**0.9989**
	di-NEC-DGT	**0.0073**	**0.92**	**0.98**	**0.9989**	di-NEC-DGT	**0.9989**

Surgery task, and 47.1% in the Targeted task. The success of the proposed di-NEC-DGT lies mainly on its outgoing and incoming edge parameters and the customized approximation of the direct spectral Laplacian matrix. In addition, for some specific tasks where the edge and node may have many more relations, the spectral regularization term plays a very important role for good prediction. For example, in the Targeted task, di-NEC-DGT has a 17.5% better metric score on average than the same model without regularization.

6 Conclusion and future work

This paper focuses on a new problem: multi-attributed graph translation. To achieve this, we propose a novel NEC-DGT consisting of several blocks which translates a multi-attributed input graph to a target graph. To jointly tackle the different types of interactions among nodes and edges, node and edge translation paths are proposed in each block and the graph spectral-based regularization is proposed to preserve the consistent spectral property of graphs. Extensive experiments have been conducted on the synthetic and real-world datasets. As the extension of the undirected NEC-DGT, this paper also proposed the sign-NEC-DGT for the signed graphs and di-NEC-DGT for the directed graphs. Experiment results show that our NEC-DGT can discover the ground-truth translation rules and significantly outperform comparison methods in terms effectiveness.

This paper provides a further step of research for graph transformation problems in more general scenarios. We are excited about the prospect of introducing the interpretability of the translation process for explainable deep graph transformation in future work, where each semantic factor in formatting the target graph can be captured and controlled by each latent dimension learned during the translation process. We further hope to inspire future work to think beyond the plane graphs for three-dimensional (3D) graph transformation, where the

geometry of graphs (i.e., the spatial location of nodes) also plays a crucial role in influencing the properties of the graphs, such as molecules and air transport networks.

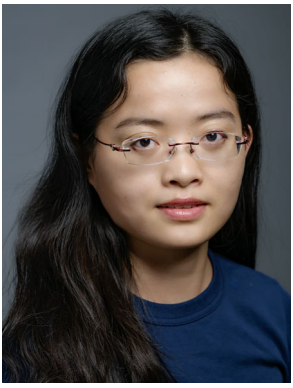
Acknowledgements This work was supported by the National Science Foundation (NSF) Grant Nos. 1755850, 1841520, 2007716, 2007976, 1942594, 1907805, a Jeffress Memorial Trust Award, Amazon Research Award, NVIDIA GPU Grant, and Design Knowledge Company (subcontract number: 10827.002.120.04).

References

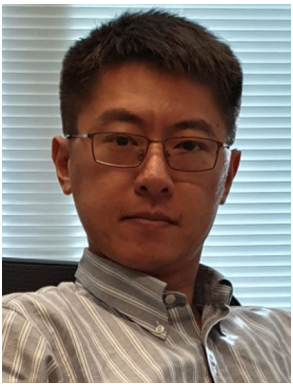
1. Abdelnour F, Voss HU, Raj A (2014) Network diffusion accurately models the relationship between structural and functional brain connectivity networks. Neuroimage 90:335–347
2. Abdelnour F, Dayan M et al (2018) Functional brain connectivity is predictable from anatomic network's Laplacian eigen-structure. Neuroimage 172:728–739
3. Barabási AL et al (1999) Emergence of scaling in random networks. Science 286(5439):509–512
4. Battaglia P, Pascanu R, Lai M, Rezende DJ et al (2016) Interaction networks for learning about objects, relations and physics. In: Advances in neural information processing systems, pp 4502–4510
5. Bézivin J, Heckel R (2005) 04101 abstracts collection—language engineering for model-driven software development. In: Dagstuhl seminar proceedings, Schloss Dagstuhl-Leibniz-Zentrum für Informatik
6. Bojchevski A, Shchur O, Zügner D, Günnemann S (2018) Netgan: generating graphs via random walks. In: International conference on machine learning, vol 80, pp 610–619
7. Bruna J, Zaremba W, Szlam A, LeCun Y (2013) Spectral networks and locally connected networks on graphs. arXiv preprint arXiv:1312.6203
8. Cao S, Lu W, Xu Q (2016) Deep neural networks for learning graph representations. In: AAAI Conference on artificial intelligence, pp 1145–1152
9. Chung F (2005) Laplacians and the Cheeger inequality for directed graphs. Ann Comb 9(1):1–19
10. Corbett D (2004) Interoperability of ontologies using conceptual graph theory. In: International conference on conceptual structures. Springer, Berlin, Heidelberg, pp 375–387
11. Dai H, Tian Y, Dai B, Skiena S, Song L (2018) Syntax-directed variational autoencoder for structured data. In: International conference on learning representations
12. Defferrard M, Bresson X, Vandergheynst P (2016) Convolutional neural networks on graphs with fast localized spectral filtering. In: Advances in neural information processing systems, pp 3844–3852
13. Ehrig H, Prange U, Taentzer G (2004) Fundamental theory for typed attributed graph transformation. In: International conference on graph transformation. Springer, pp 161–177
14. Ehrig H, Ehrig K, Prange U, Taentzer G (2006) Fundamentals of algebraic graph transformation, Monographs in theoretical computer science. an EATCS series
15. Erdős P, Rényi A (1960) On the evolution of random graphs. Publ Math Inst Hung Acad Sci 5(1):17–60
16. Galán RF (2008) On how network architecture determines the dominant patterns of spontaneous neural activity. PLoS ONE 3(5):e2148
17. Gallier J (2016) Spectral theory of unsigned and signed graphs. applications to graph clustering: a survey. arXiv preprint arXiv:1601.04692
18. Gao Y, Guo X, Zhao L (2018) Local event forecasting and synthesis using unpaired deep graph translations. In: ACM SIGSPATIAL workshop on analytics for local events and news, p 5
19. Gao Y, Wu L, Homayoun H, Zhao L (2019) Dyngraph2seq: dynamic-graph-to-sequence interpretable learning for health stage prediction in online health forums. In: International conference on data mining, pp 1042–1047
20. Gori M, Monfardini G, Scarselli F (2005) A new model for learning in graph domains. IEEE International joint conference on neural networks. IEEE, vol 2, pp 729–734
21. Guo X, Wu L, Zhao L (2018) Deep graph translation. arXiv preprint arXiv:1805.09980
22. Guo X, Zhao L, Qin Z, Wu L, Shehu A, Ye Y (2020) Interpretable deep graph generation with node-edge co-disentanglement. In: Proceedings of the 26th ACM SIGKDD international conference on knowledge discovery & data mining, pp 1697–1707
23. Guthaus MR, Ringenberg JS, Ernst D, Austin TM, Mudge T, Brown RB (2001) Mibench: a free, commercially representative embedded benchmark suite. In: IEEE international workshop on workload characterization, pp 3–14
24. Haase C, Ishtiaq S, Ouaknine J, Parkinson MJ (2013) Seloger: a tool for graph-based reasoning in separation logic. In: International conference on computer aided verification. Springer, pp 790–795
25. Hammond DK, Vandergheynst P, Gribonval R (2011) Wavelets on graphs via spectral graph theory. Appl Comput Harm Anal 30(2):129–150

26. Henning JL (2006) Spec cpu2006 benchmark descriptions. ACM SIGARCH Comput Archit News 34(4):1–17
27. Horn RA, Johnson CR (2012) Matrix analysis. Cambridge University Press, Cambridge
28. Jenkinson M, Beckmann CF, Behrens TE, Woolrich MW, Smith SM (2012) FSL. Neuroimage 62(2):782–790
29. Jin W, Coley C, Barzilay R, Jaakkola T (2017) Predicting organic reaction outcomes with Weisfeiler–Lehman network. In: Advances in neural information processing systems, pp 2607–2616
30. Jin W, Barzilay R, Jaakkola T (2018) Junction tree variational autoencoder for molecular graph generation. In: International conference on machine learning, vol 80, pp 2328–2337
31. Kaluza MCDP, Amizadeh S, Yu R (2018) A neural framework for learning DAG to DAG translation. In: Workshop on neural information processing systems
32. Kawahara J, Brown CJ, Miller SP, Booth BG, Chau V, Grunau RE, Zwicker JG, Hamarneh G (2017) BrainNetCNN: convolutional neural networks for brain networks; towards predicting neurodevelopment. Neuroimage 146:1038–1049
33. Kipf TN, Welling M (2016) Semi-supervised classification with graph convolutional networks. In: International conference on learning representations
34. Kivilcim BB, Ertugrul IO et al (2018) Modeling brain networks with artificial neural networks. In: Graphs in biomedical image analysis and integrating medical imaging and non-imaging modalities, pp 43–53
35. König B, Kozioura V (2008) Towards the verification of attributed graph transformation systems. In: International conference on graph transformation. Springer, pp 305–320
36. Kusner MJ, Paige B, Hernández-Lobato JM (2017) Grammar variational autoencoder. In: International conference on machine learning, vol 70, pp 1945–1954
37. Li Y, Tarlow D, Brockschmidt M, Zemel R (2016) Gated graph sequence neural networks. In: International conference on learning representations
38. Li Y, Vinyals O, Dyer C, Pascanu R, Battaglia P (2018) Learning deep generative models of graphs. arXiv preprint arXiv:1803.03324
39. Li Y, Yu R, Shahabi C, Liu Y (2018) Diffusion convolutional recurrent neural network: data-driven traffic forecasting. In: International conference on learning representations
40. López CM (1997) Chip firing and the Tutte polynomial. Ann Comb 1(1):253–259
41. Lowe DM (2012) Extraction of chemical structures and reactions from the literature. Doctoral dissertation, University of Cambridge
42. Mousavi SF, Safayani M, Mirzaei A, Bahonar H (2017) Hierarchical graph embedding in vector space by graph pyramid. Pattern Recognit 61:245–254
43. Mugnier ML, Chein M (1992) Conceptual graphs: fundamental notions. Revue dintelligence artificielle 6(4):365–406
44. Niepert M, Ahmed M, Kutzkov K (2016) Learning convolutional neural networks for graphs. In: International conference on machine learning, vol 48, pp 2014–2023
45. Nikolentzos G, Meladianos P, Tixier AJP, Skianis K, Vazirgiannis M (2018) Kernel graph convolutional neural networks. In: International conference on artificial neural networks, pp 22–32
46. Sai PD, Manoj HS (2019) Lightweight node-level malware detection and network-level malware confinement in IoT networks. In: ACM/EDAA/IEEE design automation and test in Europe (DATE)
47. Plump D, Steinert S (2004) Towards graph programs for graph algorithms. In: International conference on graph transformation. Springer, pp 128–143
48. Samanta B, De A, Ganguly N, Gomez-Rodriguez M (2018) Designing random graph models using variational autoencoders with applications to chemical design. arXiv preprint arXiv:1802.05283
49. Sayadi H et al (2019) 2SMaRT: a two-stage machine learning-based approach for run-time specialized hardware-assisted malware detection. In: ACM/EDAA/IEEE design automation and test in Europe (DATE)
50. Sayadi H, Patel N et al (2018) Ensemble learning for hardware-based malware detection: a comprehensive analysis and classification. In: ACM/EDAA/IEEE design automation conference
51. Scarselli F, Gori M, Tsoi AC, Hagenbuchner M, Monfardini G (2008) The graph neural network model. IEEE Trans Neural Netw 20(1):61–80
52. Simonovsky M, Komodakis N (2018) Graphvae: towards generation of small graphs using variational autoencoders. In: International conference on artificial neural networks, pp 412–422
53. Smith BM (2002) A dual graph translation of a problem in life. In: International conference on principles and practice of constraint programming. Springer, pp 402–414
54. Sturmfels P, Rutherford S, Angstadt M, Peterson M, Sripada CS, Wiens J (2018) A domain guided CNN architecture for predicting age from structural brain images. In: Machine learning for healthcare conference, vol 85, pp 295–311

55. Sun M, Li P (2019) Graph to graph: a topology aware approach for graph structures learning and generation. In: International conference on artificial intelligence and statistics, pp 2946–2955
56. Van Essen DC, Smith SM, Barch DM, Behrens TE, Yacoub E, Ugurbil K, Consortium WMH et al (2013) The Wu-Minn human connectome project: an overview. Neuroimage 80:62–79
57. Wang P, Kong R, Kong X, Liégeois R, Orban C, Deco G, van den Heuvel MP, Yeo BT (2019) Inversion of a large-scale circuit model reveals a cortical hierarchy in the dynamic resting human brain. Sci Adv 5(1):eaat7854
58. Wang S, Sun S, Li Z, Zhang R, Xu J (2017) Accurate de novo prediction of protein contact map by ultra-deep learning model. PLoS Comput Biol 13(1):e1005324
59. Wu L, Yen IEH, Zhang Z, Xu K, Zhao L, Peng X, Xia Y, Aggarwal C (2019) Scalable global alignment graph kernel using random features: from node embedding to graph embedding. In: ACM SIGKDD international conference on knowledge discovery & data mining, pp 1418–1428
60. Xu K, Wu L, Wang Z, Feng Y, Sheinin V (2018) SQL-to-text generation with graph-to-sequence model. In: Conference on empirical methods in natural language processing, pp 931–936
61. Xu K, Wu L, Wang Z, Feng Y, Witbrock M, Sheinin V (2018) Graph2seq: graph to sequence learning with attention-based neural networks. arXiv preprint arXiv:1804.00823
62. Xu K, Wu L, Wang Z, Yu M, Chen L, Sheinin V (2018) Exploiting rich syntactic information for semantic parsing with graph-to-sequence model. In: Conference on empirical methods in natural language processing, pp 918–924
63. You J, Ying R, Ren X, Hamilton WL, Leskovec J (2018) Graphrnn: generating realistic graphs with deep auto-regressive models. In: International conference on machine learning, vol 80, pp 5694–5703
64. Yu B, Yin H, Zhu Z (2018) Spatio-temporal graph convolutional networks: a deep learning framework for traffic forecasting. In: International joint conference on artificial intelligence, pp 3634–3640
65. Zhao L (2020) Event prediction in big data era: a systematic survey. arXiv preprint arXiv:2007.09815



Xiaojie Guo is a Ph.D. candidate in the Department of Information Science and Technology at GMU. She is supervised by Dr. Liang Zhao. She received her M.S. degree and B.E. degree in Control Engineering from Soochow University, China. Her research interests include data mining, artificial intelligence, and machine learning, with special interests in deep learning on graphs, deep graph transformation, deep graph generation as well as disentangled representation learning. She received the Best Paper Award from ICDM in 2019.



Liang Zhao is an Assistant Professor at the Department of Computer Science at Emory University. He received the Ph.D. degree from Virginia Tech, USA. His research interests include data mining, artificial intelligence, and machine learning, with special interests in spatiotemporal and network data mining, deep learning on graphs, nonconvex optimization, and interpretable machine learning. He has published over 80 papers in top-tier conferences and journals such as KDD, ICDM, TKDE, Proceedings of the IEEE, TKDD, TSAS, IJCAI, AAAI, WWW, CIKM, SIGSPATIAL, and SDM. He won NSF CAREER Award, Amazon Research Award, Jeffress Trust Award in 2019, Outstanding Doctoral Student in the Department of Computer Science at Virginia Tech in 2017.

Houman Homayoun is an Associate Professor in the Department of Electrical and Computer Engineering at University of California, Davis. He is the director of National Science Foundation Center for Hardware and Embedded Systems Security and Trust (CHEST). Houman conduct research in hardware security and trust, data-intensive computing and heterogeneous computing, where he has published more than 100 technical papers and directed over 8M in research funding from NSF, DARPA, AFRL, NIST and various industrial sponsors. He received several best paper awards and nominations in various conferences including GLSVLSI 2016, ICDM 2019, and ICCAD 2019, and 2020. He served as Member of Advisory Committee, Cybersecurity Research and Technology Commercialization working group in the Commonwealth of Virginia in 2018. Since 2017 he has been serving as an Associate Editor of IEEE Transactions on VLSI. He was the technical program co-chair of GLSVLSI 2018 and the general chair of 2019 conference.

Sai Manoj Pudukotai Dinakarrao is an assistant professor at George Mason University (GMU). Prior joining to GMU as an assistant professor, he was research assistant professor and post-doctoral research fellow at GMU. He received his Ph.D. in Electrical and Electronics Engineering from Nanyang Technological University, Singapore in 2015. He received his Masters in Information Technology from International Institute of Information Technology Bangalore (IIITB), Bangalore, India in 2012. His research interests include on-chip hardware security, neuromorphic computing, adversarial machine learning, self-aware SoC design, image processing and time-series analysis. His works received best paper awards and nominations in top-tier conferences. His recent works have won best paper award in ICCE 2020, ICDM 2019 and were nominated for best paper award in DATE 2018, IGSC 2020, AsianHOST 2020. He is the recipient of the "A. Richard Newton Young Research Fellow" award in Design Automation Conference, 2013.

Authors and Affiliations

Xiaojie Guo[1] · Liang Zhao[2] · Houman Homayoun[3] · Sai Manoj Pudukotai Dinakarrao[4]

✉ Liang Zhao
liang.zhao@emory.edu,lzhao9@gmu.edu

Xiaojie Guo
xguo7@gmu.edu

Houman Homayoun
hhomayoun@ucdavis.edu

Sai Manoj Pudukotai Dinakarrao
spudukot@gmu.edu

1 Department of Information Science and Technology, George Mason University, Fairfax, USA

2 Department of Computer Science, Emory University, Atlanta, Georgia

3 Department of Electrical and Computer Engineering, University of California, Davis, Davis, USA

4 Department of Electrical and Computer Engineering, George Mason Univeristy, Fairfax, USA