Infinitely Deep Graph Transformation Networks

Lei Zhang*, Qisheng Zhang*, Zhiqian Chen†, Yanshen Sun*, Chang-Tien Lu* and Liang Zhao‡
*Department of Computer Science, Virginia Tech
Email: {zhanglei, qishengz19, yansh93, ctlu}@vt.edu
†Department of Computer Science and Engineering, Mississippi State University
Email: zchen@cse.msstate.edu
†Department of Computer Science, Emory University
Email: liang.zhao@emory.edu

2023 IEEE International Conference on Data Mining (ICDM) | 979-8-3503-0788-7/23/$31.00 ©2023 IEEE | DOI: 10.1109/ICDM58522.2023.00087

***Abstract*—This work develops a node-edge co-evolution model for attributed graph transformation, where both the node and edge attributes undergo changes due to complex interactions. Due to two fundamental obstacles, learning and approximating attributed graph transformation have not been thoroughly explored: 1) the difficulty of jointly considering four types of atomic interactions including nodes-to-edges, nodes-to-nodes, edges-to-nodes, and edges-to-edges interactions. 2) the difficulty of capturing iterative long-range interactions between nodes and edges. To solve these issues, we offer a novel and scalable equilibrium model, NEC$^\infty$, with node-edge message passing and edge-node message passing. Additionally, we propose an efficient optimization algorithm that is based on implicit gradient theorem and includes a theoretical analysis of NEC$^\infty$. The effectiveness and efficiency of the proposed model have been demonstrated through extensive experiments on synthetic and real-world data sets.**

***Index Terms*—GNN, IGNN, Implicit model, Graph translation, Graph transformation, Attributed graph**

I. Introduction

Graphs have been used as universal representations of relational or interactive components in many problem domains such as social networks, physics, chemistry, and urban computing. To model and learn from such data, GNNs were proposed to generate meaningful node representations by simultaneously considering the edge attributes and node attributes. In the most recent years, a new problem, attributed graph transformation, has been proposed as a more generic task than the most existing problem settings [1]–[3]. Different from most graph-based problem settings, two graphs instead of one are involved in this problem including an input graph and a target graph. The goal of the attributed graph transformation problem is to learn and approximate the mapping from the input attributed graph to the target attributed graph, where both node attributes and edge attributes could change in the transformation.

The attributed graph transformation problem covers a wide range of real-world tasks including the ones that cannot be formulated under the regular graph neural network and node embedding settings. For example, the process of malware confinement and propagation is a typical attributed graph transformation learning problem [4]. Given the initial state of an IoT (Internet of Things) system with node and edge attributes, it is desired to predict its final state in a muti-attributed graph where both node attributes and edge attributes are changed due to malware propagation and malware epidemic control processes. The chemical reaction prediction problem is also an attributed graph transformation problem since both node (atom) attributes and edge (bond) attributes are changed from the input graph (reactant) to the target graph (product) [5].

The attributed graph transformation problem poses significant challenges and existing methods are not sufficient to fully address them. Recent advancements in deep learning have led to the development of graph-related techniques that can address specific aspects of the problem, but not all of them. Neural ordinary differential equations have been used for learning dynamics in systems with fine-grained dynamic graph data [6]–[8]. However, our problem differs from this line of work as we are only given a single snapshot (the input graph) and are required to predict the final state by an end-to-end learning and underlying process instead of being given prior knowledge of it. Some deep graph translation and one-shot graph generation methods explicitly model node-edge, edge-node, node-node, and edge-edge interactions [9]–[11]. However, they still rely on and computationally only afford a predefined and limited number of GNN layers and cannot capture long-range dependencies that suffice the inference of equilibrium of target graphs.

To address these issues, we propose an infinite-depth node-edge co-evolving (NEC$^\infty$) model for solving the graph transformation problem. Our model can jointly consider both node and edge interactions simultaneously, and has the ability to learn unknown graph transaction mappings with complex iterative interactions. Specifically, we model all node-edge interactions with an equilibrium model that update node and edge representations iteratively until a guaranteed equilibrium status. In this way, the final representations are stable fixed-point solutions for both node and edge dynamics in the model.

The contribution of this work are summarized as follows:

1) The development of a new framework for attributed graph transformation. We propose the first deep equilibrium model with the node-edge co-evolution message passing mechanism to tackle the attributed graph transformation task where both node and edge attributes can change after transformation.

2) The theoretical analysis for the deep equilibrium model with the node-edge co-evolution message passing. We derive its well-posedness condition to ensure the existence of a unique fixed-point solution.

3) The proposal of an efficient implicit function theorem-

979-8-3503-0788-7/23/$31.00 ©2023 IEEE
DOI 10.1109/ICDM58522.2023.00087

based optimization algorithm. It does require layer-by-layer backpropagation and has constant memory requirements regardless of the effective depth of the network.

4) The conduct of extensive experiments to validate the effectiveness and efficiency of the proposed model. The results show that NEC$^\infty$ can effectively capture long-range dependencies and outperform the state-of-the-art models on both synthetic and real-world datasets.

II. Related Work

A. Deep Equilibrium Models on Graph Data

Deep equilibrium model is one type of implicit model. Different from traditional explicit models (e.g., CNNs, RNNs), implicit layers/models define a layer in terms of satisfying some joint conditions of the input and output. Specifically, deep equilibrium models are defined for satisfying equilibrium equations and yielding fixed-point solutions [12]. Several works [12]–[14] show the potential advantages of deep equilibrium models on many applications, e.g., language modeling, image classification, and semantic segmentation. Just until recently, deep equilibrium models have been applied to graph data. IGNN [15] learns node representations by solving an equilibrium equation with respect to node attributes. Another work of efficient infinite-depth graph neural networks (EIGNN) derives a closed-form solution to inverse jacobian in IGNN with matrix decomposition, avoiding numeric loss and non-convergence issues in iterative solvers [16]. They all focus on improving the efficiency of optimizing the weights in an equilibrium equation for a vanilla GCN but fail to extend the model to more complicated backbone models.

B. Deep Graph Transformation Learning

Early approaches in this field were tailored to specific applications and domains. For instance, methods were proposed for molecular optimization, and protein structure generation [17], [18]. Similarly, Do et al. focused on the task of chemical reaction prediction by utilizing the explicit connectivity among nodes [19]. These approaches can be classified as one-shot generation methods which refer to building probabilistic graph models based on the matrix representation that can generate all nodes and edges in one shot [11]. Guo et al. proposed the first generic framework for graph-to-graph transformation learning, which is based on four different interaction paths [4]. More recent work has also aimed to address the attributed graph transformation problem, but with a focus on learning a distribution rather than making predictions [10].

III. Problem Formulation

Definition III.1 (Attributed Graph Transformation)**.** *An attributed graph is defined as $G = (\mathcal{V}, \mathcal{E})$, where $\mathcal{V}$ is a set of $|\mathcal{V}|$ nodes and $\mathcal{E}$ represents edges. $F \in \mathbb{R}^{D\times|\mathcal{V}|}$ is the node attributes tensor, where D is the dimension of node attributes. $E \in \mathbb{R}^{K\times|\mathcal{E}|}$ is the edge attributes tensor, where K is the dimension of edge attributes. The problem of attributed graph transformation is defined as learning a mapping $\mathcal{T}$ from an input graph G_0 to a target graph G_T: $\mathcal{T} : G_0(\mathcal{V}_0, \mathcal{E}_0, E_0, F_0) \rightarrow G_T(\mathcal{V}', \mathcal{E}', E', F')$.*

TABLE I: 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 tensor 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 tensor F'
$\|\mathcal{V}\|$	Number of nodes
$\|\mathcal{E}\|$	Number of edges
B	Node-edge incidence matrix
H_v	node embedding, $H_v \in \mathbb{R}^{p\times\|\mathcal{V}\|}$
H_e	edge embedding, $H_e \in \mathbb{R}^{q\times\|\mathcal{E}\|}$
W_{ve}	weight matrix for the node to edge message passing
W_{ev}	weight matrix for the edge to node message passing
$(\square)^{\|\cdot\|}$	element-wise absolute value of a matrix or vector

For instance, let's consider the malware containment problem. Here, V_0 refers to the IoT device characteristics, while E_0 denotes the physical distance matrix between the devices before confinement. Subsequently, V' represents the condition of the IoT devices after containment, such as being compromised or not, while E' represents the distance matrix that has been adjusted through malware epidemic management protocols.

In the attributed graph transformation task, both the node and edge attributes can change. The commonly studied node embedding/classification problem can be seen as a special case of the attributed graph transformation problem where the edge features remain the same during the transformation. Learning the attributed graph transformation necessitates addressing two key considerations: 1) **Four types of atomic interactions**. As both node attributes and edge attributes change during the transformation, and they are dependent on each other, it is essential to consider four types of atomic interactions including nodes-to-edges, nodes-to-nodes, edges-to-nodes, and edges-to-edges interactions. For example, in the case of malware confinement where nodes represent IoT devices and edges reflect communication paths between devices, the transient state of a node or edge is impacted by both the incident nodes and edges [20]. A model for learning the attributed graph transformation must possess the ability to represent all four types of atomic interactions. 2) **Iterative long-range interactions**. While each type of atomic interaction could be straightforward to model, learning the attributed graph transformation process in an end-to-end fashion is challenging as the interactions often occur iteratively or asynchronously, and result in long-range dependencies. The target graph to predict is often a stable graph after complex interactions, not a graph after a specific number of atomic interactions. As an example, the PageRank algorithm can be viewed as a graph transformation process where the target graph is the graph

when the PageRank has converged. In this case, the local operation of PageRank can be easily described in a linear model, but the entire convergence process is non-trivial to model with any traditional feed-forward neural networks.

Attributed graph translation should consider both the above considerations, which cannot be comprehensively handled by existing methods. The typical practice of tackling the problem of long-range dependencies is using a large number of layers. However, besides this strategy being computationally expensive, it can lead to over-smoothing issues, resulting in the model failing to capture relevant information. Incorporating four types of atomic interactions within a single model and balancing the need for long-range dependency without over-smoothing present a significant challenge.

IV. Infinite-Depth Node-Edge Co-Evolution

In this section, we formally introduce the proposed NEC$^\infty$ and elaborate its implementation details.

A. Overall Architecture

The proposed infinite-depth model for the attributed graph transformation learning problem can be defined as:

$$\begin{aligned} &F' = g_{\Theta_e}(H_e^t), E' = f_{\Theta_v}(H_v^t), \\ &H_e^{t+1}, H_v^{t+1} = \mathcal{T}(H_e^t, H_v^t, F_0, E_0), \\ \text{subject to: } &H_e^t = H_e^{t+1} \text{ and } H_v^t = H_v^{t+1}. \end{aligned} \tag{1}$$

Here the function $\mathcal{T}$ is applied to both the node and edge embeddings recurrently until it converges to a fixed point, i.e., $H_e^t = H_e^{t+1}$, $H_v^t = H_v^{t+1}$, here $H_\square^t$ is the hidden representation after t-th layer/iteration. This type of formulation is commonly referred to as a deep equilibrium model (DEQ) and can be considered as a recurrent neural network operating on a graph with the constraint that a fixed-point solution must be found. Once the fixed-point is discovered, the function $\mathcal{T}$ is equivalent to a model with an infinite number of layers and is able to capture global-range node/edge dependencies, as both H_v and H_e will remain unchanged with the addition of extra layers. The key advantage of using this equilibrium formulation is that $\mathcal{T}$ is immune to oversmoothing because it is still considered as a single implicit layer. Specifically, the function $\mathcal{T}$ in Eq. (1) is not a feedforward neural network, but rather an implicit layer, as the output of the layer is defined by satisfying the equilibrium equation.

The proposed NEC$^\infty$ learns the graph transformation process through the use of a specifically designed DEQ function $\mathcal{T}$. In Section IV-B, we formally define the function $\mathcal{T}$ to tackle the different types of interactions among nodes and edges for the graph transformation problem. In Section IV-C, we discuss the strict mathematical condition for when Eq. (1) can function as intended (i.e., yielding a fixed-point solution as the output) and formulate the problem as a constrained optimization problem. In Section IV-D, an efficient optimization method based on projected gradient descent (PGD) and alternating direction method of multipliers (ADMM) is proposed for solving the constrained optimization problem.

B. Node-Edge Co-Evolution Message Passing

To handle the attributed graph transformation problem, the message passing function $\mathcal{T}$ in Eq. (1) must be able to capture the four types of atomic interactions among nodes and edges as outlined in Section III. In order to take into account the nature of recurrent models, function $\mathcal{T}$ is decomposed into two basic paths: the *node-to-edge message passing path* and the *edge-to-node message passing path*. These two paths are then applied iteratively, allowing for direct inferences of the nodes-to-edges and edges-to-nodes atomic interactions. As the message passing procedure is applied iteratively, the nodes-to-nodes and edges-to-edges interactions can also be inferred by stacking one layer of node-to-edge message passing path and one layer of edge-to-node message passing path, or one layer of edge-to-node message passing path and one layer of node-to-edge message passing path respectively. The two message-passing paths are defined in the following equations.

$$\begin{aligned} \text{Node-to-edge: } &H_e^{t+1} = \mathcal{T}_1(H_v^t, B) = \phi(W_{ve} H_v^t B), \\ \text{Edge-to-node } &H_v^{t+1} = \mathcal{T}_2(H_e^t, B) = \phi(W_{ev} H_e^t B^\intercal), \end{aligned} \tag{2}$$

where H_v^t and H_e^t are the node embedding and edge embedding on the $t-$th layer, respectively. B is the incident matrix. W_{ve} and W_{ev} are trainable weights. ϕ is an activation function. In this paper, we use ReLU activation, but it can be any activation function with CONE property (e.g. Sigmoid, tanh, ReLU, LeakyReLU, etc.). During the node/edge message passing procedures, the node embeddings $H_v \in \mathbb{R}^{p\times|\mathcal{V}|}$ always remain the same, while edge embeddings $H_e \in \mathbb{R}^{q\times|\mathcal{E}|}$ also remain the same. To map from the feature spaces $F \in \mathbb{R}^{D\times|\mathcal{V}|}/E \in \mathbb{R}^{K\times|\mathcal{E}|}$ to the embedding spaces $H_v \in \mathbb{R}^{p\times|\mathcal{V}|}$ / $H_e \in \mathbb{R}^{q\times|\mathcal{E}|}$, we add an MLP b_Ω. If we explicitly seek node embedding fixed points, the final message passing function is defined as :

$$\begin{aligned} H_v^* &= \mathcal{T}_v(H_v^*, B) \\ &= \phi(W_{ev}\phi(W_{ve}H_v^* B)B^\intercal + b_\Omega(E_0, F_0)), \\ H_e^* &= \mathcal{T}_e(H_v^*, B) = \phi(W_{ve}H_v^* B). \end{aligned} \tag{3}$$

Once the fixed-point for node embeddings is found as is shown in Eq. (3), the fixed-point for edge embeddings is also automatically found because of the nature of equilibrium functions. Assuming the fixed-point solution for the node embedding is H_v^*, then the edge embedding in the next layer is $\mathcal{T}_e(H_v^*, B)$. As the node embeddings have reached the fixed point, the node embedding in the next layer will still be H_v^*, thus the next of the next edge embedding will still be $\mathcal{T}_e(H_v^*, B)$. Alternatively, if we seek edge embedding fixed points first, the function b_Ω will be defined to output features in the space of $\mathbb{R}^{q\times|\mathcal{E}|}$.

C. Sufficient Well-posedness Condition for NEC$^\infty$

While our model in Eq. (3) is designed to generate a fixed-point solution, it may not always yield a solution for arbitrary input and weights. Thus, the notion of *well-posedness* and the sufficient well-posedness condition play an important role in our model as a deep equilibrium model. This notion has been previously introduced in [21] for ordinary implicit models and

in [15] for ordinary graph neural networks. To ensure the existence and uniqueness of the solution to Eq. (3), we define the notion of well-posedness for NEC$^\infty$ in attributed graph transformation problems.

Definition IV.1 (Well-posedness for NEC$^\infty$)**.** *The tuple (W_{ve}, W_{ev}, B) is said to be well-posed for ϕ if for any $b \in \mathbb{R}^{p\times|\mathcal{V}|}$, the solution $H_v \in \mathbb{R}^{p\times|\mathcal{V}|}$ of the following equation*

$$H_v = \phi(W_{ev}\phi(W_{ve}H_vB)B^\intercal + b), \tag{4}$$

exists and is unique.

Similar to [21], we derive the sufficient condition for the well-posedness to hold based on the proposed node-edge co-evolution message passing. According to Definition IV.1, the condition is about the relationship among three variables including W_{ve}, W_{ev}, and B.

Theorem IV.2 (PF sufficient condition for well-posedness on Eq. (3)**.** *Suppose ϕ is a component-wise non-expansive (CONE) activation map. In such a case, (W_{ve}, W_{ev}, B) is considered well-posed for any ϕ if $\lambda_{pf}((B\otimes W_{ev})^{|\cdot|}(B^\intercal \otimes W_{ve})^{|\cdot|}) < 1$. Additionally, the solution H_v to equation (3) can be obtained by iteratively applying equation (3).*

The proof of Theorem IV.2 is given in Appendix A

While the condition about the Perron-Frobenius (PF) eigenvalue $\lambda_{pf}(BB^\intercal)\lambda_{pf}(W_{ev}^{|\cdot|}W_{ve}^{|\cdot|}) < 1$ in Theorem IV.2 guarantees that the well-posedness can be obtained, calculating the PF eigenvalue is very costly. To avoid a costly spectral decomposition process, we enforce the following more strict condition $\|W_{ev}^{|\cdot|}W_{ve}^{|\cdot|}\|_\infty < \lambda_{pf}(BB^\intercal)^{-1}$. It has been proven in existing work that this is still equivalent to the sufficient PF condition [15]. More details can be found in Appendix C.

D. Optimization of NEC$^\infty$

The optimization of weights in NEC$^\infty$ differs from most existing neural networks because the layer is defined by equilibrium and the optimization process involves constraints. The loss function can be written as:

$$\begin{aligned}\min_{\Theta_v,\Theta_e,W_{ev},W_{ve},\Omega} & \mathcal{L}_1(F', g_{\Theta_e}(H_v)) + \mathcal{L}_2(E', f_{\Theta_e}(H_e)),\\ \text{subject to: } & H_e = \phi(W_{ve}H_vB),\\ & H_v = \phi(W_{ev}\phi(W_{ve}H_vB)B^\intercal + b_\Omega(E_0, F_0)),\\ & \lambda_{pf}(BB^\intercal)\lambda_{pf}(W_{ev}^{|\cdot|}W_{ve}^{|\cdot|}) < 1,\end{aligned} \tag{5}$$

where the constraint $\lambda_{pf}(BB^\intercal)\lambda_{pf}(W_{ev}^{|\cdot|}W_{ve}^{|\cdot|}) < 1$ is derived from Section IV-C. Readout functions g_{Θ_e} and f_{Θ_e} are MLPs.

To efficiently solve this problem while keeping the function stable and constrained, we use a combination of projected gradient descent (PGD) and implicit function theorem (IFT). Since H_v and H_e are also the fixed point solutions, we derive the analytic solution with IFT. In this way, we can avoid doing backpropagation on each layer, and the memory usage remains constant regardless of the number of layers before the model reaches the fixed-point. To ensure that the model will converge, while we move in the direction of the negative gradient, we also use PGD to "project" the weights onto the feasible set defined by the constraint. The projection itself is also a sub-problem of optimization and is solved by using alternating direction method of multipliers (ADMM) [22].

1) ADMM-based Projection: The projection operator can be expressed as the following optimization problem.

$$\begin{aligned}\{W_{ev}^+, W_{ve}^+\} = \operatorname{argmin} & ||W_{ve} - W_{ve}^+||_F^2 + ||W_{ev} - W_{ev}^+||_F^2,\\ \text{subject to: } & \lambda_{pf}(W_{ev}^{|\cdot|}W_{ve}^{|\cdot|}) < \kappa, \kappa = 1/\lambda_{pf}(BB^\intercal),\end{aligned} \tag{6}$$

where W_{ev}^+ and W_{ve}^+ refer to the updated versions of W_{ev} and W_{ve}, respectively, after the projection. We enforce the stricter condition $||W_{ev}^{|\cdot|}W_{ve}^{|\cdot|}||_\infty \le \kappa$. The projection operator now becomes a child optimization problem in Eq. (7).

$$\begin{aligned}\text{minimize} \quad & ||X - P||_F^2 + ||Y - Q||_F^2,\\ \text{subject to} \quad & ||X^{|\cdot|}Y^{|\cdot|}||_\infty \le \kappa.\end{aligned} \tag{7}$$

As the constraint in Eq. (7) is for $X^{|\cdot|}$, each dimension in X and P will always be of the same sign. If one dimension in X has the opposite sign with the corresponding dimension in P, the opposite of that dimension will also satisfy the same constraint but result in a smaller loss. Similarly, each dimension in Y and Q will also always have the same sign. For simplicity, we can first minimize $||X^{|\cdot|} - P^{|\cdot|}||_F^2 + ||Y^{|\cdot|} - Q^{|\cdot|}||_F^2$. Once the optimal $X' = X^{|\cdot|}$ and $Y' = Y^{|\cdot|}$ are obtained, the actual X and Y can be simply calculated by $X = \mathbf{sign}(P)\odot X'$ and $Y = \mathbf{sign}(Q)\odot Y'$. This trick simplifies the problem in Eq. (7) to the following version, assuming all values of X, Y, P, and Q are positive:

$$\begin{aligned}\text{minimize} \quad & ||X - P||_F^2 + ||Y - Q||_F^2,\\ \text{subject to} \quad & XY = C, ||C||_\infty \le \kappa.\end{aligned} \tag{8}$$

To solve Eq. (8) with ADMM, We first define the augmented Lagrangian, for a parameter $\rho > 0$:

$$\begin{aligned}L_\rho(X, Y, C, \lambda) = & ||X - P||_F^2 + ||Y - Q||_F^2\\ & + \langle\lambda, XY - C\rangle_F + (\rho/2)||XY - C||_F^2.\end{aligned} \tag{9}$$

ADMM consists of the following iterations:

$$X = (2P - \lambda Y^\intercal + \rho CY^T)(2I + \rho YY^\intercal)^{-1}, \tag{10}$$

$$Y = (2I + \rho X^\intercal X)^{-1}(2Q - X^\intercal\lambda + \rho X^\intercal C), \tag{11}$$

$$C = \operatorname*{argmin}_{||C||_\infty\le\kappa} \langle\lambda, XY - C\rangle_F, \tag{12}$$

$$\lambda = \lambda + \rho(XY - C). \tag{13}$$

See Appendix B for more details on how the analytical solutions for X, Y, and C are derived.

Denote optimal primal variables by X^* and Y^*, and the optimal dual variable by λ^*. The primal feasibility is measured by primal residual:

$$r = ||X^{k+1}Y^{k+1} - C^{k+1}||_F. \tag{14}$$

The dual residual can be defined as:

$$s = \rho X^{k+1}(Y^k - Y^{k+1})(Y^k)^\intercal. \tag{15}$$

Algorithm 1 summarizes the ADMM optimization iteration.

Algorithm 1 ADMM-based projection for PGD

Input: Weight metrics A and B, parameter κ
Choose $\varepsilon^{pri} > 0$, $\varepsilon^{dual} > 0$
repeat
  Update X by Eq. (10) // analytical solution
  Update Y by Eq. (11) // analytical solution
  Update C by infinity norm
  Update λ by $\lambda^{k+1} = \lambda^k + \rho(XY - C)$
  Calculate the primal residual by Eq. (14)
  Calculate the dual residual by Eq. (15)
  if $r > 10s$ **then**
    $\rho \leftarrow 2\rho$
  else if $10r < s$ **then**
    $\rho \leftarrow \rho/2$ //varying penalty parameter
  else
    $\rho \leftarrow \rho$ //varying penalty parameter
  end if
until $r < \varepsilon^{pri}, s < \varepsilon^{dual}$

2) Gradient Descent based on Implicit Function Theorem: After the projection, we calculate the gradients of loss with respect to weights in the equilibrium model by utilizing the implicit function theorem.

$$H_v = \phi(W_{ev}\phi(W_{ve}H_vB)B^\intercal + b_\Omega(E_0, F_0)). \tag{16}$$

From the chain rule, it is easy to obtain $\nabla_{H_v}\mathcal{L}$ for the internal state. In addition, we can write the gradient w.r.t scalar $q \in W_{ev}$ as follows:

$$\nabla_q\mathcal{L} = \langle\frac{\partial Z}{\partial q}, \nabla_Z\mathcal{L}\rangle, \tag{17}$$

where $Z = W_{ve}H_vB$ assuming fixed H_v. Unlike H_v that is implicitly defined, Z is a closed evaluation of $Z = W_{ve}H_vB$ assuming H_v doesn't change depending on Z. We find that $\nabla_Z\mathcal{L}$ can be calculated by solving the following equilibrium equation in Eq. (18). This format is similar with how IGNN was solved but with different derivation (see Appendix D for details).

$$\nabla_Z\mathcal{L} = D_1 \odot (W_{ev}^\intercal(D_2 \odot (W_{ve}^\intercal\nabla_Z\mathcal{L}B^\intercal))B + \nabla_{H_v}\mathcal{L}). \tag{18}$$

After $\nabla_Z\mathcal{L}$ is calculated, it is easy to infer the derivative of the loss with respect to W_{ev} and W_{ve}:

$$\begin{aligned}\nabla_{W_{ev}}\mathcal{L} &= \langle\frac{\partial Z}{\partial W_{ev}}, \nabla_Z\mathcal{L}\rangle = \nabla_Z\mathcal{L}R(\phi(W_{ve}H_vB))^\intercal.\\ \nabla_{W_{ve}}\mathcal{L} &= H_vB((B^\intercal\nabla_Z\mathcal{L}W_{ev}) \odot \phi'(W_{ve}H_vB)).\end{aligned} \tag{19}$$

Detailed partial derivative calculations for Eq. (19) can be found in Appendix E.

The updates of b_ω is done automatically with chain rule and autograd. In the backpropagation, $\nabla_Z\mathcal{L}$ is calculated first with the IFT-based algorithm (Appendix D). From there on, optimizing b_ω is not different from updating weights in a feedforward NN.

V. Experiments

In this section, we present the evaluation results over the proposed NEC$^\infty$ model.

A. Experimental Setup

1) Datasets: We performed experiments on a set of publicly available attributed graph transformation datasets, which consisted of four synthetic random graph datasets and four real-world datasets. In order to assess our model's capability in tracking long-range dependencies, we also created two additional synthetic dynamical system datasets.

Synthetic random graph datasets: To evaluate the ability of models to approximate predefined node-edge translation functions, we utilized four synthetic benchmark datasets Syn I - Syn IV [4][1]. These datasets were generated using different random graph generators and translation rules. The input graph structures were created using either the Erdős-Rényi (ER) model or the Barabási-Albert (BA) model, with the number of nodes ranging from 20 to 60. The target graph structure is defined as the multi-hop graph of the input graph, where each edge in the target graph represents multi-hop reachability in the input graph. The node attributes in the input graphs correspond to the node degrees, while the node attributes in the target graphs are calculated using a predefined polynomial function applied to the node attributes in the input graphs. In both the input and target graphs, the edge attributes are binary variables that indicate whether an edge is present or absent. Each dataset consists of 500 pairs of input-output graphs.

Synthetic dynamical system dataset: To further demonstrate the models' performance in handling graph translation problems involving long-range node-edge interactions, we generated two additional discrete dynamical system datasets: Syn-V and Syn-VI. Each dataset contains 500 pairs of input-output graph pairs. The graph structures in Syn-V were created with ER model in the same way as Syn-I dataset. The graph structures in Syn-VI were created with BA model in the same way as Syn-IV dataset. The 1-dimensional input node features were sampled from a uniform distribution on the interval $[0, 1)$. The output node features are the stable-state results of the dynamical system with linear evolution functions.

Malware confinement dataset: Three malware datasets for IoT devices are used for evaluating the performance of malware confinement prediction tasks [23]. For all three datasets, the nodes in the input graph represent IoT devices in the system where the node attribute is a binary value referring to whether the device is compromised or not. The edge attribute between two nodes is defined as the physical distance between two devices. The target graphs represent the graphs with updated node and edge attributes after the malware confinement, which can be considered as stable graphs resulting from a dynamical system [20]. Each of the three datasets consists of 334 pairs of input and target graphs, with varying contextual parameters such as infection rate, recovery rate, and decay rate.

Molecule reaction dataset: We apply graph translation methods to a fundamental problem in organic chemistry, which involves predicting the product (target graph) of a chemical reaction given the reactant (input graph). The datasets used in

[1]source: https://github.com/xguo7/Dataset-for-Deep-Graph-Translation

our study were collected by Lowe [24] from USPTO granted patents for chemical reaction extraction studies [24], and they have been previously used for attributed graph transformation evaluations [4]. We work with a dataset comprising 5,000 reactant-product pairs, which are evenly divided into training and testing sets. The node features in this dataset include elemental identity, connectivity degree, hydrogen atoms, valence, and aromaticity, while the bond features capture bond type and connectivity.

2) Comparison Methods: To assess the effectiveness of our proposed method, we conducted comparisons with several existing approaches, namely *NR-DGT*, *NEC-DGT* [4], and *IGNN* [15]. NR-DGT is a node-edge co-evolution GNN model with two blocks/layers, while NEC-DGT is a variant of NR-DGT that incorporates graph spectral-based regularization. On the other hand, IGNN is a deep equilibrium model that only passes messages for node embedding. In comparison to these existing approaches, our proposed model shares similarities with IGNN in terms of being an infinite layer GNN and considers node-edge co-evolution, akin to NEC-DGT. Additionally, we also compared our proposed method against two categories of state-of-the-art techniques: 1) link attribute prediction/graph structure learning (GSL) methods, and 2) node classification/regression methods. These comparisons allowed us to thoroughly evaluate the performance of our approach and understand its relative strengths and weaknesses.

Link attribute prediction / GSL methods: *GT-GAN* is a recent generative adversarial network for graph topology learning [10]. *GraphRNN* is an LSTM-based deep autoregressive model that can approximate any distribution of graphs with minimal assumptions about their structure [25]. *GraphVAE* is a variational autoencoder-based graph topology generation method [26].

Node classification/regression methods: *IN* is a general GNN framework for learning node-level, edge-level and graph-level representations [27]. *DCRNN* is a holistic approach that captures both spatial and temporal dependencies using diffusion convolution [28]. *STGCN* constructs ST-Conv blocks with spatial convolution layers and residual connections [29].

3) Evaluation: The attributed graph transformation problem is a multi-objective machine learning problem. While node attributes and edge attributes are used as input at the same time, the node prediction and edge prediction are evaluated separately. For different datasets, the target attributes, for both nodes and edges, can be binary values or continuous values. We use MSE, R2, Pearson's r, and Spearman's r as metrics. For binary values, we use accuracy as the metric. For all datasets, we followed the same training and testing protocols as described in [4]. [2]

B. Performance

1) **Metric-based evaluation for synthetic datasets:** Results for synthetic random graph datasets (Syn-I - Syn-IV) are shown in Table II. It can be seen that the proposed model outperforms all comparison methods on both the node and edge attributes prediction. Specifically, in terms of node attribute prediction, the methods that consider both node and edge message passing (NR-DGT, NEC-DGT, NEC$^\infty$) perform significantly better than the methods that only do node-to-node message passing (IN, DCRNN, STGCN). This is because the traditional node-to-node message-passing methods only consider a static graph topology while the attributed graph transformation problem requires explicitly modeling the evolving edge attributes as well. Furthermore, the proposed NEC$^\infty$ performs better than NR-DGT and NEC-DGT. Especially on the Syn-IV dataset for BA graphs, the MSE has been decreased one order of magnitude (from 1.86 to 0.183) compared with the second-best performed method. The additional performance boost comes from the infinite depth equilibrium layer that captures global graph information. When it comes to edge attribute prediction, the proposed NEC$^\infty$ also shows significant advantages. It outperforms NEC-DGT by 30% on average and more for the rest methods. On Syn-IV, the edge prediction accuracy is improved to almost 100%. The results demonstrate the effectiveness of the infinite-depth node-edge co-evolution.

Results for synthetic dynamical system datasets are shown in Table III. Only node attribute prediction results are evaluated because the dynamical systems in Syn V and Syn VI only output new node attributes. Our main baseline method NEC-DGT still performs well but was not able to outperform STGCN on Syn-VI due to the unchanged edges. However, for both node MESE and Pearson's r, NEC$^\infty$ outperforms all the comparison methods. It is not a surprise because equilibrium functions are naturally the mathematical tools for modeling discrete dynamical systems.

2) **Evaluation of the learned translation mapping for synthetic data:** To illustrate whether the inherent mapping mechanism for both node and edge attributes in the attributed graph transformation problem is learned correctly by NEC$^\infty$, we visualize the ground-truth mapping and plot the learned distribution by NEC$^\infty$ for all the four synthetic datasets. The ground-truth mapping is drawn according to the predefined functions in the dataset generation described in section V-A1. Figure 1 shows a ground-truth line in green and predicted values in red dots. As shown in Figure 1, the predicted values are located closely around the ground-truth plot. This is mainly because the equilibrium architecture approximates the underlying transformation dynamics in a natural way.

Figure 2 visualizes the convergence processes for both the forward pass fixed-point solution in Eq. (3) of H_v and the intermediate variable $\nabla_Z \mathcal{L}$ in Eq. (18).

3) **Metric-based evaluation for malware confinement datasets:** Performance metrics for the malware detection task are shown in Table IV. The edge attributes are continuous values and thus are evaluated by E-MSE, E-R2, and E-P. The node attributes are evaluated by E-Acc. NEC$^\infty$ achieves the overall best performance. For node attributes prediction, NEC$^\infty$ performs the best on the first two datasets but slightly

[2]The source codes can be found at https://anonymous.4open.science/r/NEC-infty–6A1C/

TABLE II: 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)

Data	Method	N-MSE	N-R2	N-P	N-Sp	Method	E-Acc
Syn-I	IN	5.97	0.06	0.48	0.44	GraphRNN	0.621
	DCRNN	51.36	0.12	0.44	0.45	GraphVAE	0.659
	STGCN	15.44	0.19	0.42	0.56	GT-GAN	0.703
	IGNN	14.69	0.007	0.82	0.89	NR-DGT	0.701
	NR-DGT	2.13	0.87	0.90	0.89	NEC-DGT	0.712
	NEC-DGT	1.98	0.76	0.93	0.91	NEC$^\infty$	**0.944**
	NEC$^\infty$	**1.018**	**0.93**	**0.965**	**0.959**		
Syn-II	IN	1.36	0.85	0.77	0.87	GraphRNN	0.562
	DCRNN	71.07	0.11	0.39	0.37	GraphVAE	0.463
	STGCN	33.11	0.21	0.15	0.15	GT-GAN	0.700
	IGNN	3.76	0.88	0.94	0.94	NR-DGT	0.701
	NR-DGT	1.43	0.91	0.94	0.97	NEC-DGT	0.720
	NEC-DGT	1.91	0.93	0.97	0.97	NEC$^\infty$	**0.969**
	NEC$^\infty$	**1.203**	**0.96**	**0.982**	**0.983**		
Syn-III	IN	35.46	0.31	0.59	0.56	GraphRNN	0.452
	DCRNN	263.23	0.09	0.41	0.39	GraphVAE	0.370
	STGCN	43.34	0.22	0.48	0.47	GT-GAN	0.577
	IGNN	3.27	0.90	0.95	0.95	NR-DGT	0.625
	NR-DGT	5.90	0.90	0.94	0.92	NEC-DGT	0.658
	NEC-DGT	4.56	0.93	0.97	0.96	NEC$^\infty$	**0.955**
	NEC$^\infty$	**3.322**	**0.95**	**0.976**	**0.974**		
Syn-IV	IN	4.63	0.10	0.53	0.51	GraphRNN	0.517
	DCRNN	63.03	0.12	0.22	0.16	GraphVAE	0.300
	STGCN	6.52	0.08	0.11	0.10	GT-GAN	0.805
	IGNN	4.16	0.32	0.62	0.65	NR-DGT	0.670
	NR-DGT	4.49	0.12	0.55	0.54	NEC-DGT	0.843
	NEC-DGT	1.86	0.73	0.93	0.89	NEC$^\infty$	**0.998**
	NEC$^\infty$	**0.183**	**0.97**	**0.98**	**0.985**		

TABLE III: Evaluation of Generated Target Graphs for Dynamical System Dataset

Dataset	Method	N-MSE	N-P	Dataset	Method	N-MSE	N-P
Syn-V	IN	0.004	0.67	Syn-VI	IN	0.003	0.81
	DCRNN	0.005	0.44		DCRNN	0.011	0.70
	STGCN	0.005	0.53		STGCN	0.007	0.92
	IGNN	0.016	0.85		IGNN	0.0018	0.96
	NR-DGT	0.017	0.63		NR-DGT	0.017	0.74
	NEC-DGT	0.009	0.76		NEC-DGT	0.0009	0.87
	NEC$^\infty$	**0.003**	**0.92**		NEC$^\infty$	**0.0004**	**0.99**

worse than STGCN on Malware III. The most possible reason is that Malware III is less dependent on the node-edge interactions and NEC$^\infty$ has to be trained to perform both node attribute and edge attribute predictions at the same time. In summary, the node-edge message passing methods (NEC-DGT and NEC$^\infty$) can handle node and edge prediction at the same time better than the rest methods. By introducing the infinite-depth node-edge message passing, NEC$^\infty$ consistently performs better than its finite layer counterpart.

4) **Metric-based evaluation for molecule reaction datasets:** In this task, the proposed model is compared with the baselines (NEC-DGT and NR-DGT) and traditional reaction prediction method WLDN. Table V shows the performance of all methods on edge accuracy, the main metric for reaction prediction tasks. Even the baselines achieve good performance, they cannot reach a perfect 100 percent accuracy like NEC$^\infty$ does. This shows that NEC$^\infty$ can be applied to a wide range of real-world applications and make accurate node and edge attribute predictions at the same time.

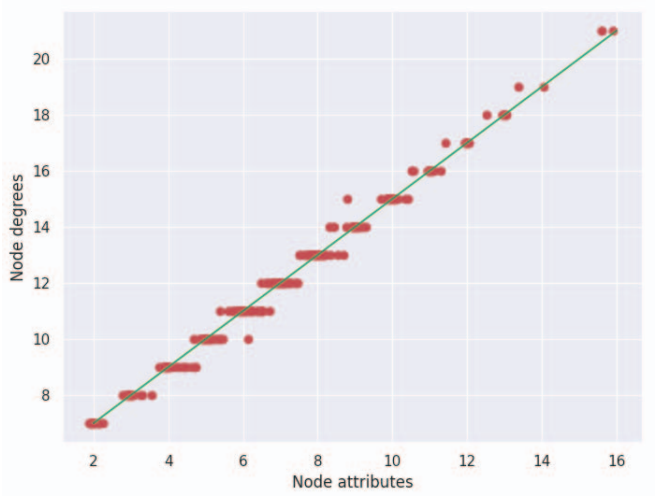

Fig. 1: Visualizations of predicted node attributes and the ground truth relationship for synthetic graphs.

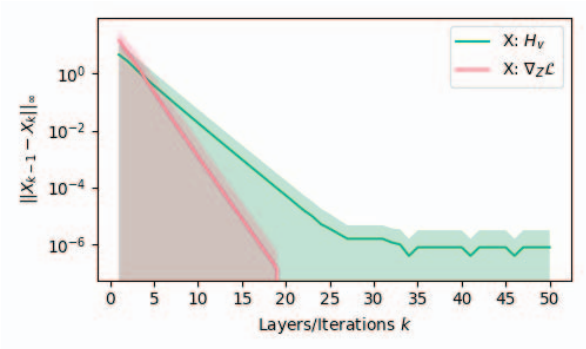

Fig. 2: Visualization of the fixed-point solution as results for node embedding H_v and $\nabla_Z \mathcal{L}$ in optimization.

C. Scalability Analysis

Due to the node-edge and edge-node message passing, the time complexity of NEC$^\infty$ is $\mathcal{O}(|\mathcal{V}|^2)$, matching that of NEC-DGT and proving more scalable than GraphVAE ($\mathcal{O}(|\mathcal{V}|^4)$). It's important to note that an equilibrium layer is generally slower than a feed-forward layer with the same function; hence, a NEC$^\infty$ layer is expected to be slower than a NEC-DGT layer at the same time complexity. However, it's worth highlighting that a single NEC$^\infty$ can be viewed as an infinite number of equivalent layers.

The proposed NEC$^\infty$ model with the IFT-based optimization algorithm is more efficient and scalable than the recurrent-version counterpart. We refer to the counterpart as *NEC-Rec* which is a recurrent model with the same module (node-edge co-evolution message passing) and weight tying. To demonstrate it, Figure 3 and 4 illustrates the scalability of NEC$^\infty$ with respect to the graph size and the number of layers respectively. The run time and memory usage of both NEC$^\infty$ and NEC-Rec increase superlinearly when the number of nodes increases due to the inevitable pairwise node-to-edge message passing. However, NEC$^\infty$ runs much faster and

TABLE IV: 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)

Malware-I						
Method	E-Acc	E-MSE	E-R2	E-P	Method	N-Acc
GraphRNN	0.610	1831.43	0.52	0.00	IN	0.878
GraphVAE	0.506	2453.61	0.00	0.04	DCRNN	0.878
GT-GAN	0.630	1718.02	0.42	0.11	STGCN	0.923
NR-DGT	0.910	668.57	**0.82**	0.91	IGNN	0.882
NEC-DGT	0.921	239.79	0.78	0.91	NR-DGT	0.910
NEC$^{\infty}$	**0.938**	**195.72**	0.78	**0.92**	NEC-DGT	0.929
					NEC$^{\infty}$	**0.934**
Malware-II						
Method	E-Acc	E-MSE	E-R2	E-P	Method	N-Acc
GraphRNN	0.705	1950.46	0.44	0.29	IN	0.882
GraphVAE	0.606	2410.57	0.73	0.16	DCRNN	0.879
GT-GAN	0.903	462.73	0.13	0.81	STGCN	0.933
NR-DGT	0.911	448.48	0.68	0.83	IGNN	0.843
NEC-DGT	0.938	244.40	0.81	0.91	NR-DGT	0.885
NEC$^{\infty}$	**0.941**	**233.66**	**0.81**	**0.92**	NEC-DGT	0.934
					NEC$^{\infty}$	**0.948**
Malware-III						
Method	E-Acc	E-MSE	E-R2	E-P	Method	N-Acc
GraphRNN	0.839	1775.58	0.16	0.23	IN	0.873
GraphVAE	0.8119	2109.64	0.39	0.32	DCRNN	0.873
GT-GAN	0.945	550.30	0.63	0.80	STGCN	**0.937**
NR-DGT	0.954	341.10	0.76	0.88	IGNN	0.876
NEC-DGT	0.960	273.67	0.81	0.90	NR-DGT	0.877
NEC$^{\infty}$	**0.978**	**231.99**	**0.83**	**0.92**	NEC-DGT	0.900
					NEC$^{\infty}$	0.928

TABLE V: 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	8e−2	0.46	0.13	0.12	GT-GAN	0.868
STGCN	6e−4	0.98	**0.99**	0.97	WLDN	0.966
IGNN	6e−4	0.98	**0.99**	**0.99**	NR-DGT	0.991
NR-DGT	8e−4	0.97	**0.99**	**0.99**	NEC-DGT	0.992
NEC-DGT	**4e−4**	**0.99**	**0.99**	**0.99**	NEC$^{\infty}$	**1.0**
NEC$^{\infty}$	**4e−4**	**0.99**	**0.99**	**0.99**		

becomes more memory efficient when the graph grows larger. The run time and memory usage of NEC$^{\infty}$ stay stable when the number of layers increases because the proposed optimization algorithm updates the weight in the model without the layer-by-layer backpropagation.

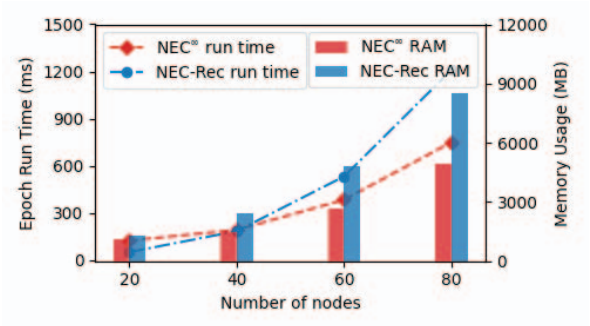

Fig. 3: Run time and RAM usage w.r.t the number of nodes.

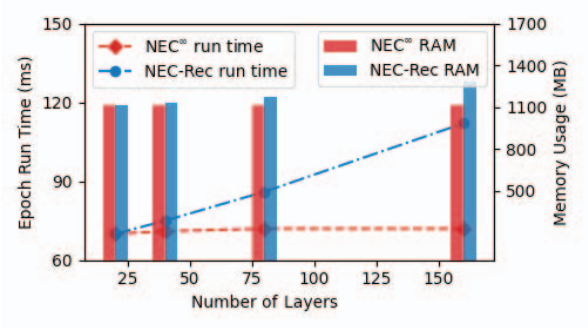

Fig. 4: Run time and RAM usage w.r.t number of layers.

VI. Conclusion

This paper focuses on a new problem, end-to-end attributed graph transformation. To achieve this, we propose a novel NEC$^{\infty}$ method consisting of a graph deep equilibrium model which translates an initial graph to a target graph with different node and edge attributes. To jointly tackle complicated node-edge dynamics, the infinite-depth node-edge co-evolution message passing is proposed. We also proposed an efficient implicit theorem-based optimization algorithm to avoid heavy computation and memory overhead. To the best of our knowledge, NEC$^{\infty}$ is the first work of its kind that is capable of incorporating both node and edge dynamics within an equilibrium architecture. Extensive experiments have been conducted on both synthetic and real-world datasets. Experiment results have shown that our NEC$^{\infty}$ can approximate the underlying ground-truth translation rules, even those with iterative graph-wide operations, and it significantly outperforms existing methods and baselines.

References

[1] L. Wu, P. Cui, J. Pei, L. Zhao, and L. Song, "Graph neural networks," in *Graph Neural Networks: Foundations, Frontiers, and Applications*. Springer, 2022, pp. 27–37.

[2] C. Graber and A. Schwing, "Dynamic neural relational inference for forecasting trajectories," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops*, 2020, pp. 1018–1019.

[3] X. Guo, L. Zhao, Z. Qin, L. Wu, A. Shehu, and Y. Ye, "Interpretable deep graph generation with node-edge co-disentanglement," in *Proceedings of the 26th ACM SIGKDD international conference on knowledge discovery & data mining*, 2020, pp. 1697–1707.

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

[5] Y. Du, X. Guo, Y. Wang, A. Shehu, and L. Zhao, "Small molecule generation via disentangled representation learning," *Bioinformatics (Oxford, England)*, p. btac296, 2022.

[6] J. Z. Kolter and G. Manek, "Learning stable deep dynamics models," *Advances in neural information processing systems*, vol. 32, 2019.

[7] Z. Huang, Y. Sun, and W. Wang, "Coupled graph ode for learning interacting system dynamics." in *KDD*, 2021, pp. 705–715.

[8] C. Zang and F. Wang, "Neural dynamics on complex networks," in *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, 2020, pp. 892–902.

[9] T. Kipf, E. Fetaya, K.-C. Wang, M. Welling, and R. Zemel, "Neural relational inference for interacting systems," in *International Conference on Machine Learning*. PMLR, 2018, pp. 2688–2697.

[10] X. Guo, L. Wu, and L. Zhao, "Deep graph translation," *IEEE Transactions on Neural Networks and Learning Systems*, 2022.
[11] X. Guo and L. Zhao, "A systematic survey on deep generative models for graph generation," *arXiv preprint arXiv:2007.06686*, 2020.
[12] S. Bai, J. Z. Kolter, and V. Koltun, "Deep equilibrium models," *Advances in Neural Information Processing Systems*, vol. 32, 2019.
[13] S. Bai, V. Koltun, and J. Z. Kolter, "Multiscale deep equilibrium models," *Advances in Neural Information Processing Systems*, vol. 33, pp. 5238–5250, 2020.
[14] R. T. Chen, Y. Rubanova, J. Bettencourt, and D. K. Duvenaud, "Neural ordinary differential equations," *Advances in neural information processing systems*, vol. 31, 2018.
[15] F. Gu, H. Chang, W. Zhu, S. Sojoudi, and L. El Ghaoui, "Implicit graph neural networks," *Advances in Neural Information Processing Systems*, vol. 33, pp. 11 984–11 995, 2020.
[16] J. Liu, K. Kawaguchi, B. Hooi, Y. Wang, and X. Xiao, "Eignn: Efficient infinite-depth graph neural networks," *Advances in Neural Information Processing Systems*, vol. 34, pp. 18 762–18 773, 2021.
[17] N. Anand and P. Huang, "Generative modeling for protein structures," *Advances in neural information processing systems*, vol. 31, 2018.
[18] W. Jin, K. Yang, R. Barzilay, and T. Jaakkola, "Learning multimodal graph-to-graph translation for molecular optimization," *arXiv preprint arXiv:1812.01070*, 2018.
[19] K. Do, T. Tran, and S. Venkatesh, "Graph transformation policy network for chemical reaction prediction," in *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, 2019, pp. 750–760.
[20] H. Sayadi, H. M. Makrani, S. M. P. Dinakarrao, T. Mohsenin, A. Sasan, S. Rafatirad, and H. Homayoun, "2smart: A two-stage machine learning-based approach for run-time specialized hardware-assisted malware detection," in *2019 Design, Automation & Test in Europe Conference & Exhibition (DATE)*. IEEE, 2019, pp. 728–733.
[21] L. El Ghaoui, F. Gu, B. Travacca, A. Askari, and A. Tsai, "Implicit deep learning," *SIAM Journal on Mathematics of Data Science*, vol. 3, no. 3, pp. 930–958, 2021.
[22] S. Boyd, N. Parikh, E. Chu, B. Peleato, J. Eckstein, *et al.*, "Distributed optimization and statistical learning via the alternating direction method of multipliers," *Foundations and Trends® in Machine learning*, vol. 3, no. 1, pp. 1–122, 2011.
[23] S. M. P. Dinakarrao, H. Sayadi, H. M. Makrani, C. Nowzari, S. Rafatirad, and H. Homayoun, "Lightweight node-level malware detection and network-level malware confinement in iot networks," in *2019 Design, Automation & Test in Europe Conference & Exhibition (DATE)*. IEEE, 2019, pp. 776–781.
[24] D. M. Lowe, "Extraction of chemical structures and reactions from the literature," Ph.D. dissertation, University of Cambridge, 2012.
[25] J. You, R. Ying, X. Ren, W. Hamilton, and J. Leskovec, "Graphrnn: Generating realistic graphs with deep auto-regressive models," in *International conference on machine learning*. PMLR, 2018, pp. 5708–5717.
[26] M. Simonovsky and N. Komodakis, "Graphvae: Towards generation of small graphs using variational autoencoders," in *International conference on artificial neural networks*. Springer, 2018, pp. 412–422.
[27] P. Battaglia, R. Pascanu, M. Lai, D. Jimenez Rezende, *et al.*, "Interaction networks for learning about objects, relations and physics," *Advances in neural information processing systems*, vol. 29, 2016.
[28] Y. Li, R. Yu, C. Shahabi, and Y. Liu, "Diffusion convolutional recurrent neural network: Data-driven traffic forecasting," in *International Conference on Learning Representations (ICLR '18)*, 2018.
[29] B. Yu, H. Yin, and Z. Zhu, "Spatio-temporal graph convolutional networks: A deep learning framework for traffic forecasting," in *Proceedings of the 27th International Joint Conference on Artificial Intelligence (IJCAI)*, 2018.

APPENDIX

A. PF sufficient condition for well-posedness

Theorem A.1 (PF sufficient condition for well-posedness on Eq. (3)**).** *Assume that ϕ is a component-wise non-expansive (CONE) activation map. Then, (W_{ve}, W_{ev}, B) is well-posed for any such ϕ if $\lambda_{pf}((B \otimes W_{ev})^{|\cdot|}(B^\intercal \otimes W_{ve})^{|\cdot|}) < 1$. Moreover, the solution H_v of equation (3) can be obtained by iterating equation (3).*

Proof.

$$H_v = \phi(W_{ev}\phi(W_{ve}H_vB)B^\intercal + b_\Omega(E_0, F_0)). \tag{20}$$

vectorize both sides:

$$\begin{aligned}\mathbf{vec}(H_v) &= \phi(\mathbf{vec}(W_{ev}\phi(W_{ve}H_vB)B^\intercal) + \mathbf{vec}(b)) \\ &= \phi((B \otimes W_{ev})\,\mathbf{vec}(\phi(W_{ve}H_v))) + \mathbf{vec}(b)).\end{aligned} \tag{21}$$

as ϕ is a component-wise non-expansive (CONE) activation map:

$$\begin{aligned}&\mathbf{vec}(H_v^{t+1} - H_v^t)^{|\cdot|} \\ &= (\phi((B \otimes W_{ev})\,\mathbf{vec}(\phi(W_{ve}H_v^tS))) + \mathbf{vec}(B)) \\ &- \phi((B \otimes W_{ev})\,\mathbf{vec}(\phi(W_{ve}H_v^{t-1}B))) + \mathbf{vec}(B)))^{|\cdot|} \\ &\le ((B \otimes W_{ev})\,\mathbf{vec}(\phi(W_{ve}H_v^tB)) - (B \otimes W_{ev})\,\mathbf{vec}(\phi(W_{ve}H_v^{t-1}B)))^{|\cdot|} \\ &\le (B \otimes W_{ev})^{|\cdot|}(\mathbf{vec}(\phi(W_{ve}H_v^tB)) - \mathbf{vec}(\phi(W_{ve}H_v^{t-1}B)))^{|\cdot|} \\ &\le (B \otimes W_{ev})^{|\cdot|}(\mathbf{vec}(W_{ve}H_v^tB) - \mathbf{vec}(W_{ve}H_v^{t-1}B))^{|\cdot|} \\ &= (B \otimes W_{ev})^{|\cdot|}(\mathbf{vec}(W_{ve}(H_v^t - H_v^{t-1})B))^{|\cdot|} \\ &= (B \otimes W_{ev})^{|\cdot|}((B^\intercal \otimes W_{ve})\,\mathbf{vec}((H_v^t - H_v^{t-1})))^{|\cdot|} \\ &\le (B \otimes W_{ev})^{|\cdot|}(B^\intercal \otimes W_{ve})^{|\cdot|}\,\mathbf{vec}(H_v^t - H_v^{t-1})^{|\cdot|}.\end{aligned} \tag{22}$$

According to Lemma A.2, Eq. (3) has a unique solution if $\lambda_{pf}((B \otimes W_{ev})^{|\cdot|}(B^\intercal \otimes W_{ve})^{|\cdot|}) < 1$

$\lambda_{pf}((B \otimes W_{ev})^{|\cdot|}(B^\intercal \otimes W_{ve})^{|\cdot|}) = \lambda_{pf}((BB^\intercal) \otimes (W_{ev}^{|\cdot|}W_{ve}^{|\cdot|})) = \lambda_{pf}(BB^\intercal)\lambda_{pf}(W_{ev}^{|\cdot|}W_{ve}^{|\cdot|}) < 1.$

□

Lemma A.2. *If ϕ is component-wise non-negative (CONE), M is some squared matrix and v is any real vector of compatible shape, the equation $x = \phi(Mx + v)$ has a unique solution if $\lambda_{pf}(|M|) < 1$. And the solution can be obtained by iterating the equation. Hence, $x = \lim_{t\to\infty} x_t$.*

$$x_{t+1} = \phi(Mx_t + v),\ x_0 = 0,\ t = 0, 1, \ldots \tag{23}$$

The proof of Lemma A.2 can be found in the supplementary material B.3 in [16].

B. ADMM-based Production

ADMM consists of the iterations:

$$\begin{aligned}X^{k+1} &= \operatorname*{argmin}_X(||X^k - P||_F^2 + \langle\lambda, X^kY^k - C^k\rangle \\ &\quad + (\rho/2)||X^kY^k - C^k||_F^2), \\ Y^{k+1} &= \operatorname*{argmin}_Y(||Y^k - Q||_F^2 + \langle\lambda, X^{k+1}Y^k - C\rangle \\ &\quad + (\rho/2)||X^{k+1}Y^k - C^k||_F^2), \\ C^{k+1} &= \operatorname*{argmin}_{||C||_\infty \le \kappa} \langle\lambda, X^{k+1}Y^{k+1} - C^k\rangle_F \\ &\quad + ||X^{k+1}Y^{k+^k} - C^k||_F^2), \\ \lambda^{k+1} &= \lambda^k + \rho(X^{k+1}Y^{k+1} - C^{k+1}).\end{aligned} \tag{24}$$

For X:

$$\begin{aligned}
&\frac{\partial||X-P||_F^2+\langle\lambda,XY-C\rangle_F+(\rho/2)||XY-C||_F^2}{\partial X}=0,\\
&2(X-P)+\lambda Y^T+\rho(XY-C)Y^T=0,\\
&2X-2P+\lambda Y^T+\rho XYY^T-\rho CY^T=0,\\
&X(2I+\rho YY^\intercal)=2P-\lambda Y^T+\rho CY^T,\\
&X=(2P-\lambda Y^T+\rho CY^T)(2I+\rho YY^\intercal)^{-1}.
\end{aligned}\tag{25}$$

For Y:

$$\begin{aligned}
&\frac{\partial||Y-Q||_F^2+\langle\lambda,XY-C\rangle_F+(\rho/2)||XY-C||_F^2}{\partial Y}=0,\\
&2(Y-Q)+X^\intercal\lambda+\rho X^\intercal(XY-C)=0,\\
&2Y-2Q+X^\intercal\lambda+\rho X^\intercal XY-\rho X^\intercal C=0,\\
&(2I+\rho X^\intercal X)Y=2Q-X^\intercal\lambda+\rho X^\intercal C,\\
&Y=(2I+\rho X^\intercal X)^{-1}(2Q-X^\intercal\lambda+\rho X^\intercal C).
\end{aligned}\tag{26}$$

For C:

The optimization can be decomposed along the rows of C. Each subproblem involves projecting onto an $\mathcal{L}_1$-ball, and efficient methods for this operation are available.

C. Additional information for the stricter sufficient well-posedness condition

We adopt a similar method to that used in [15] to derive a more stringent and computationally tractable sufficient condition for well-posedness of our model.

$$\begin{aligned}
&\lambda_{pf}(W_{ev}^{|\cdot|}W_{ve}^{|\cdot|})\\
=&\inf_S||SW_{ev}^{|\cdot|}W_{ve}^{|\cdot|}S^{-1}||_\infty : S=\mathbf{diag}(S), s>0.
\end{aligned}\tag{27}$$

In the case where $W_{ev}^{|\cdot|}W_{ve}^{|\cdot|}$ has simple PF eigenvalue, problem (27) admits positive optimal scaling factor $s>0$, a PF eigenvector of $W_{ev}^{|\cdot|}W_{ve}^{|\cdot|}$. And we can design the equivalent model with $\|W_{ev}'^{|\cdot|}W_{ve}'^{|\cdot|}\|_\infty<\lambda_{pf}(BB^\intercal)^{-1}$ by rescaling:

$$\begin{aligned}
&\tilde{f_\Theta}(\cdot)=f_\Theta(S^{-1}\,\cdot),\quad |W_{ev}'^{|\cdot|}W_{ve}'^{|\cdot|}=SW_{ev}^{|\cdot|}W_{ve}^{|\cdot|}S^{-1},\\
&\tilde{b_\Omega}(\cdot)=Sb_\Omega(\cdot),\quad \text{where } S=\mathbf{diag}(s).
\end{aligned}\tag{28}$$

D. Additional information for the IFT-based GD

To avoid taking derivatives of matrices by matrices, we again introduce the vectorized representation $vec(\cdot)$ of matrices. The vectorization of a matrix $H_v\in\mathbb{B}^{p\times|\mathcal{V}|}$ denoted vec(X), is obtained by stacking the columns of H_v into one single column vector of dimension $p|\mathcal{V}|$. For simplicity, we use $\overrightarrow{H_v}:=vec(H_v)$ as a shorthand notation of vectorization.

With vectorization, we have:

$$\overrightarrow{H_v}=\phi((B\otimes W_{ev})\phi((B^\intercal\otimes W_{ve})\overrightarrow{H_v})+\overrightarrow{B}).\tag{29}$$

similarly, now we change the definition of Z

$$\overrightarrow{Z}=((B\otimes W_{ev})\phi((B^\intercal\otimes W_{ve})\overrightarrow{H_v})+\overrightarrow{B}.\tag{30}$$

According to Lemma A.3 in Appendix D, we have

$$\begin{aligned}
\frac{\partial\overrightarrow{H_v}}{\partial\overrightarrow{Z}}=&\frac{\partial\phi((B\otimes W_{ev})\phi((B^\intercal\otimes W_{ve})\overrightarrow{H_v})+\overrightarrow{B})}{\partial\overrightarrow{Z}}\\
&+\frac{\partial\phi((B\otimes W_{ev})\phi((B^\intercal\otimes W_{ve})\overrightarrow{H_v})+\overrightarrow{B})}{\partial\overrightarrow{H_v}}\frac{\partial\overrightarrow{H_v}}{\partial\overrightarrow{Z}},
\end{aligned}\tag{31}$$

where

$$\begin{aligned}
&\frac{\partial\phi((B\otimes W_{ev})\phi((B^\intercal\otimes W_{ve})\overrightarrow{H_v})+\overrightarrow{B})}{\partial\overrightarrow{H_v}}\\
=&\frac{\partial\phi(\overrightarrow{Z})}{\partial\overrightarrow{Z}}\frac{\partial((B\otimes W_{ev})\phi((B^\intercal\otimes W_{ve})\overrightarrow{H_v})+\overrightarrow{B})}{\partial\overrightarrow{H_v}}\\
=&\tilde{D_1}(B\otimes W_{ev})^\intercal(B^\intercal\otimes W_{ve})^\intercal\tilde{D_2},
\end{aligned}\tag{32}$$

and

$$\begin{aligned}
\tilde{D_1}&=\frac{\partial\phi(\overrightarrow{Z})}{\partial\overrightarrow{Z}},\\
\tilde{D_2}&=\frac{\partial\phi((B^\intercal\otimes W_{ve})\overrightarrow{H_v})}{\partial(B^\intercal\otimes W_{ve})\overrightarrow{H_v}}.
\end{aligned}\tag{33}$$

$$\nabla_{\overrightarrow{Z}}\mathcal{L}=(\frac{\partial\overrightarrow{H_v}}{\partial\overrightarrow{Z}})^\intercal\nabla_{\overrightarrow{H_v}}\mathcal{L}.\tag{34}$$

Plugging Eq. (31) to (34), we arrive at the following equilibrium equation

$$\nabla_{\overrightarrow{Z}}\mathcal{L}=\tilde{D_1}(B\otimes W_{ev})^\intercal(B^\intercal\otimes W_{ve})^\intercal\tilde{D_2}\nabla_{\overrightarrow{Z}}\mathcal{L}+\tilde{D_1}\nabla_{\overrightarrow{H_v}}\mathcal{L},\tag{35}$$

or in the devectorized form:

$$\nabla_Z\mathcal{L}=D_1\odot(W_{ev}^\intercal(D_2\odot(W_{ve}^\intercal\nabla_Z\mathcal{L}B^\intercal))B+\nabla_{H_v}\mathcal{L}).\tag{36}$$

here the $\nabla_{H_v}\mathcal{L}$ can be easily obtained through modern autograd frameworks so $\nabla_Z\mathcal{L}$ can be calculated in the fixed-point function in Eq. (36).

Lemma A.3. *Using Implicit Function Theorem. The function $W\mapsto H$ is defined implicitly by $H=f(H,W)$.*
Then, $H'(W)=\frac{\partial f}{\partial H}H'(W)+\frac{\partial f}{\partial W}$.

E. Details for the derivative calculation in Section

For calculating $\nabla_{W_{ve}}\mathcal{L}$, it requires calculating matrix to matrix derivatives so we use the the vectorized representations again.

$$\begin{aligned}
\nabla_{\overrightarrow{W_{ve}}}\mathcal{L}&=\langle\frac{\partial\overrightarrow{Z}}{\partial\overrightarrow{W_{ve}}},\nabla_{\overrightarrow{Z}}\mathcal{L}\rangle=\langle\frac{\partial\overrightarrow{Z}}{\partial\overrightarrow{\phi}}\frac{\partial\overrightarrow{\phi}}{\partial\overrightarrow{W_{ve}}},\nabla_{\overrightarrow{Z}}\mathcal{L}\rangle\\
&=\langle(R\otimes W_{ev})\frac{\partial\overrightarrow{\phi}}{\partial\overrightarrow{W_{ve}}},\nabla_{\overrightarrow{Z}}\mathcal{L}\rangle\\
&=\langle(R\otimes W_{ev})\frac{\partial\overrightarrow{\phi}}{\partial\overrightarrow{W_{ve}H_vB}}\frac{\partial\overrightarrow{W_{ve}H_vB}}{\partial\overrightarrow{W_ve}},\nabla_{\overrightarrow{Z}}\mathcal{L}\rangle\\
&=\nabla_{\overrightarrow{Z}}\mathcal{L}((R\otimes W_{ev})\phi'(\overrightarrow{W_{ve}H_vB})((B^\intercal H_v^\intercal)\otimes I)).
\end{aligned}\tag{37}$$

Here I is a $q\times q$ identity matrix.

Thus

$$\nabla_{W_{ve}}\mathcal{L}=H_vB((B^\intercal\nabla_Z\mathcal{L}W_{ev})\odot\phi'(W_{ve}H_vB)).\tag{38}$$