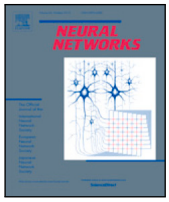
Neural Networks 180 (2024) 106649

Contents lists available at ScienceDirect

Neural Networks

journal homepage: www.elsevier.com/locate/neunet



# Deep graph representation learning for influence maximization with accelerated inference

Tanmoy Chowdhury [a,1], Chen Ling [b,1], Junji Jiang [c], Junxiang Wang [d], My T. Thai [e], Liang Zhao [b,*]

[a] Richland County Government, Columbia, SC, USA
[b] Emory University, Atlanta, GA, USA
[c] Fudan University, Shanghai, China
[d] NEC Laboratories America, Princeton, NJ, USA
[e] University of Florida, Gainesville, FL, USA

ARTICLE INFO

*Keywords:*
Influence maximization
Diffusion model
Combinatorial optimization
Deep learning
Supervised learning
Unsupervised learning

ABSTRACT

Selecting a set of initial users from a social network in order to maximize the envisaged number of influenced users is known as influence maximization (IM). Researchers have achieved significant advancements in the theoretical design and performance gain of several classical approaches, but these advances are almost reaching their pinnacle. Learning-based IM approaches have emerged recently with a higher generalization to unknown graphs than conventional methods. The development of learning-based IM methods is still constrained by a number of fundamental hardships, including (1) solving the objective function efficiently, (2) struggling to characterize the diverse underlying diffusion patterns, and (3) adapting the solution to different node-centrality-constrained IM variants. To address the aforementioned issues, we design a novel framework DeepIM for generatively characterizing the latent representation of seed sets, as well as learning the diversified information diffusion pattern in a data-driven and end-to-end way. Subsequently, we design a novel objective function to infer optimal seed sets under flexible node-centrality-based budget constraints. Extensive analyses are conducted over both synthetic and real-world datasets to demonstrate the overall performance of DeepIM.

## 1. Introduction

In the realm of network analysis, Influence Maximization (IM) stands as a foundational research problem. Its core objective is to discover a set of seed nodes that can maximize the diffusion of influence throughout a social network. IM has garnered significant attention in recent times due to its substantial commercial implications. For instance, consider the context of viral marketing (Chen, Wang, & Wang, 2010) for promoting a commercial product. In this scenario, a company aims to spread the adoption of a new product through a selected group of initial users. These users are expected to propagate information about the product within their social circles, leading to a cascading effect and ultimately a considerable portion of users trying the product. Beyond viral marketing, IM holds pivotal importance in various applications including network monitoring (Wang, Fan, Li, & Tan, 2017), curbing misinformation (Yang, Li, & Giua, 2020), and enhancing friend recommendations (Ye, Liu, & Lee, 2012).

IM poses a classic combinatorial optimization challenge, involving the identification of an optimal or near-optimal seed set to maximize influence in a network. This task is intricate due to the stochastic nature of information propagation and the inherent complexity of the problem. Traditional IM methods (Kempe, Kleinberg, & Tardos, 2003; Leskovec et al., 2007; Nguyen, Thai, & Dinh, 2016; Saito, Kimura, Ohara, & Motoda, 2012; Tang, Shi, & Xiao, 2015; Tang, Xiao, & Shi, 2014) have made remarkable strides, even achieving exact solutions under specific diffusion models (Li, Smith, Dinh, & Thai, 2019). These approaches usually demand an explicit information diffusion model as input, yet real-world diffusion processes are multifaceted and cannot be encapsulated by fixed models. With the emergence of machine and

* Corresponding author.
*E-mail addresses:* tchowdh6@gmu.edu (T. Chowdhury), chen.ling@emory.edu (C. Ling), junji.anjou@gmail.com (J. Jiang), junxiang.wang@alumni.emory.edu (J. Wang), mythai@cise.ufl.edu (M.T. Thai), liang.zhao@emory.edu (L. Zhao).
*URLs:* https://www.tanmoychowdhury.com (T. Chowdhury), https://www.lingchen0331.github.io/ (C. Ling), https://xianggebenben.github.io/Junxiang_Wang.github.io/ (J. Wang), https://www.cise.ufl.edu/~mythai (M.T. Thai), https://www.cs.emory.edu/~lzhao41/ (L. Zhao).
[1] Authors contributed equally.

https://doi.org/10.1016/j.neunet.2024.106649

Available online 31 August 2024
0893-6080/Published by Elsevier Ltd.

deep learning, a learning-based approach appears promising to capture the underlying diffusion dynamics.

Despite notable progress in the field, learning-based solutions for IM are still nascent due to several fundamental challenges. First, the efficient optimization of the objective function remains a hurdle. Learning-based methods attempt to tackle the discrete problem within a continuous space, predominantly leveraging deep network representations (Kumar, Mallik, Khetarpal, & Panda, 2022; Zhang, Li, Wei, Liu, & Li, 2022) and deep reinforcement learning (Li et al., 2022; Tian, Mo, Wang, & Peng, 2020). Even though they could attain a competitive performance with traditional methods, their scalability and execution efficiency are problematic due to *(a)* the need to iteratively update all node embeddings at each action and *(b)* the #P-hardness of computing the influence spread (Lin, Chen, & Lui, 2017). Second, the automatic identification and modeling of actual diffusion processes pose difficulties. The underlying diffusion pattern profoundly influences the spread of information in a network. However, both traditional and learning-based methods lack the ability to characterize these processes without resorting to heuristics. They typically rely on predefined diffusion models (e.g., Linear Threshold and Independent Cascade) as inputs, which are limited in capturing the complexity of real-world scenarios. Third, adapting solutions to diverse node-centrality-constrained IM problems remains a challenge. Variants of IM that involve node centrality constraints (such as seed node count or total degree constraints) lack a unified approach in current learning-based methods, posing adaptivity challenges.

To address these challenges, we propose an innovative framework - DeepIM. Our primary contribution is to develop a novel objective function that allows for efficient optimization within a continuous space, diverging from traditional discrete approaches that often face scalability and local optima issues. This framework introduces a novel strategy of embedding the discrete optimization domain into a larger continuous space. Notably, it advocates learning the latent representation of seed sets and optimizing directly in this continuous space to alleviate computational complexity. To jointly consider the optimization with the information diffusion modeling, we introduced a learning-based diffusion model that adapts to various real-world scenarios without the need for predefined models. Moreover, our framework enhances adaptability through its ability to handle different node-centrality constraints, which is crucial for applying IM across diverse network structures and requirements. Finally, our proposed method significantly reduces the computational overhead compared to existing solutions and highlights the practical applicability of our approach in real-world scenarios where computational resources are a constraint. In summary, our contributions encompass:

- **Problem Formulation:** We cast the learning-based IM challenge as the embedding of the initial discrete optimization domain into a continuous space, highlighting the challenges intrinsic to real-world applications.
- **Framework Development:** We propose a joint approach involving the latent space representation of seed sets and a model that learns graph diffusion processes in an end-to-end fashion.
- **Adaptivity Enhancement:** Our novel constrained optimization objective leverages deep graph embeddings, facilitating optimal seed set inference under diverse node-centrality-related constraints.
- **Evaluation:** Through extensive experimentation on four real-world datasets, we showcase DeepIM's performance. It outperforms state-of-the-art methods across various application scenarios in the pursuit of identifying seed sets for maximal influence.

## 2. Related work

### 2.1. Learning-based influence maximization

Most conventional approaches to Influence Maximization (IM) can be classified into simulation-based, proxy-based, and heuristic-based categories. These traditional methods have achieved nearly exact or precise solutions within specific diffusion models, demonstrating efficiency. It is worth noting that while there have been indications of learning influence from cascade data in works such as (Du, Liang, Balcan, & Song, 2014; Vaswani et al., 2017), they still assumed the guidance of a predefined model for diffusion patterns, specifically the Coverage function. For a more comprehensive overview of traditional methods, readers can refer to recent surveys like (Banerjee, Jenamani, & Pratihar, 2020; Li, Fan, Wang, & Tan, 2018).

In contrast, learning-based approaches harness the power of deep learning to address the limitations of traditional IM methods, primarily their lack of generalization capabilities. Pioneering efforts such as Ali, Wang, and Chen (2018), Lin, Lin, and Chen (2015) initially combined reinforcement learning with IM, igniting a wave of research that utilizes deep reinforcement learning to tackle the IM challenge. Modern state-of-the-art solutions like Chen, Yan, Guo, and Wu (2022), Li et al. (2022), Li, Xu, et al. (2019), Tian et al. (2020) adhere to a similar framework: learning latent embeddings for nodes or networks, treating the current node embedding as the agent's state to select the next seed node as an action, with the reward being its incremental influence gain.

Apart from reinforcement learning-based IM methods, there are also techniques like Kamarthi, Vijayan, Wilder, Ravindran, and Tambe (2019), Kumar et al. (2022), Panagopoulos, Malliaros, and Vazirgiannis (2020) that exclusively employ graph neural networks to encode social influence into node embeddings and steer the node selection process. Nevertheless, the current learning-based IM methods share a common challenge: their model complexity and adaptability still lag behind traditional methods. Specifically, existing ML-based algorithms struggle to handle diverse diffusion patterns and cannot ensure solution quality and model scalability as effectively as traditional methods do.

### 2.2. Graph neural network

Graph Neural Networks (GNNs) (Wu et al., 2020) belong to a category of deep learning techniques designed to handle data represented as graphs. The fundamental approach of GNNs involves iteratively transforming node features and aggregating information from neighboring nodes. For a GNN with $K$ layers, each node gathers information within its $K$-hop neighborhood. Concretely, the transformation at the $k$th layer is as follows:

$$a^k = \mathcal{A}^k(h^{k-1}; \theta^k), h^k = C^k(a^k; \theta^k), \forall 1 \leq k \leq K. \tag{1}$$

Here, $a^k$ represents the aggregated features, and $h^k$ corresponds to the node features at the $k$th layer. The choice of aggregation function $\mathcal{A}(\cdot)$ and combination function $C(\cdot)$ gives rise to distinct GNN models (Kipf & Welling, 2016; Veličković et al., 2017; Xu, Hu, Leskovec, & Jegelka, 2018). These high-level node or graph representations find applications in diverse tasks. GNNs have been effectively used in various domains such as estimating information diffusion and source localization (Ling, Jiang, et al., 2023; Ling, Jiang, Wang, & Zhao, 2022; Wang, Jiang, & Zhao, 2022; Xia, Li, Wu, & Li, 2021), generating complex graphs (Guo, Wang, & Zhao, 2022; Ling, Cao, & Zhao, 2023; Ling, Yang, & Zhao, 2021, 2023; Wang, Guo, & Zhao, 2022), and addressing reasoning challenges (Chowdhury et al., 2023; Ling, Chowdhury, et al., 2022). In our endeavor, GNNs serve as a tool to characterize underlying diffusion patterns and construct an end-to-end model for estimating influence.

### 2.3. Combinatorial optimization with GNNs

Observational data is frequently required to address uncertainty in real-world decision-making systems. As a result, combinatorial optimization (CO) issues where the objective function is unknown commonly arise and require being refuted using actual data. Graphs are a major topic of research in the CO area due to the discrete structure of most CO problems and the abundance of network data in the actual world. Because there is no unique representation in the graph; machine

learning algorithms have extra hurdles, such as being invariant to node permutation, expressiveness, scalability, leveraging new information, and, most importantly, achieving data-efficient generalization. The parameterized aggregation stage of GNNs, on the other hand, offers representation by learning the crucial graph topologies and scaling linearly with the number of edges and parameters. Because of its invariance to permutations and awareness of input sparsity, GNNs' inductive bias effectively encodes combinatorial and relational data. As a result, GNNs have evolved into an end-to-end solution (Nowak, Villar, Bandeira, & Bruna, 2018) or integrated component either in traditional optimization algorithms (Li, Chen, & Koltun, 2018) or with other machine learning algorithms (Khalil, Dai, Zhang, Dilkina, & Song, 2017). There are a lot of applications of combinatorial optimization by GNNs for solving NP-hard problems like satisfiability (SAT), Maximal Independent Set (MIS), Minimum Vertex Cover (MVC), and Maximal Clique (MC), Minimum Coverage Problem (MCP), and etc. For example, Li, Chen, and Koltun (2018) utilized GCN to direct tree search operations for solving NP-hard problems like SAT, MIS, MVC, MC. Influence Maximization (IM) was first formulated as a combinatorial optimization problem by Kempe et al. (2003), which has inspired extensive research and applications in the next decade. Sahil et al. (Manchanda et al., 2019) solve IM problem on billion-size graphs with GCN assisting in encoding the influence of a node on the solution set. The minimum dominating set problem (MDSP), a variant of IM problems, was solved by the use of a hybrid algorithm that fuses a biased random key genetic algorithm with a graph neural network (Sartori & Blum, 2022). To produce high-quality solutions, deep Q-learning (DQN) has made extensive use of a variety of GNNs, including the graph convolution network (GCN) (Li, Chen, & Koltun, 2018), message-passing neural network (MPNN) (Barrett, Clements, Foerster, & Lvovsky, 2020), and graph attention network (GAT) (Cappart, Moisan, Rousseau, Prémont-Schwarz, & Cire, 2021). However, utilizing GNNs can still present issues like over-smoothing and information squashing (Wang, Li, et al., 2022; Zhou et al., 2020), particularly in unsupervised tasks (Yang, Gu, et al., 2020; Zhang & Zhao, 2022). We quote recent surveys (Cappart et al., 2023; Vesselinova, Steinert, Perez-Ramirez, & Boman, 2020) as additional in-depth analysis sources.

## 3. Problem formulation

Given a graph $G = \{V, E\}$, the primary goal of Influence Maximization (IM) is to strategically choose an optimal seed node set $\mathbf{x} \subseteq V$ in order to maximize the number of nodes influenced within the graph $G$. The effectiveness of IM is evaluated through an influence diffusion model, characterized by parameter $\theta$: $\mathbf{y} = M(\mathbf{x}, G; \theta)$. In the case of an independent cascade model, $\theta$ may represent the set of infection probabilities associated with each node. Alternatively, if $M(\cdot)$ is GNN-based, $\theta$ could encompass parameters related to aggregation and combination functions. We denote $\mathbf{x} \in \{0,1\}^{|V|}$ as the vector representation of the source node set, where the $i$th element $x_i = 1, x_i \in \mathbf{x}$ if $v_i \in \mathbf{x}$ and $x_i = 0$ otherwise. The output $y \in \mathbb{R}_+$ denotes the total count of infected nodes in the graph (Li, Fan, et al., 2018). Building upon this conceptualization of influence spread, the IM problem is formally defined as follows:

**Definition 1** (*Influence Maximization*)**.** The generic *IM* problem requires selecting a set of $k$ users from $V$ as the seed set to maximize the influence spread:

$$\tilde{\mathbf{x}} = \arg\max_{|\mathbf{x}| \le k} M(\mathbf{x}, G; \theta), \tag{2}$$

where $\tilde{\mathbf{x}}$ is the optimal seed node set that can produce a maximal influence spread in $G$.

Certainly, the process of selecting $\tilde{\mathbf{x}}$ is intricately tied to the nuances of the underlying diffusion dynamics. Over time, numerous endeavors have emerged that employ GNNs and reinforcement learning algorithms to address this challenge. Nonetheless, the potential of existing learning-based IM frameworks is somewhat constrained, predominantly due to the following challenges: Firstly, a major challenge in existing learning-based IM frameworks is the computational burden associated with calculating latent node embeddings for selecting highly influential nodes. These frameworks typically necessitate the iterative update of embeddings for each node during every action or optimization step, regardless of their inclusion in the current $\mathbf{x}$. This scalability issue becomes particularly pronounced when dealing with networks comprising millions of nodes. Secondly, even though deep node and network embeddings, along with diverse reward functions, are employed to guide node selection, current frameworks remain tailored to specific diffusion models. For instance, they often model $M(\cdot)$ as explicit Independent Cascade (IC) or Linear Threshold (LT) models. Regrettably, these simplistic diffusion models fall short of meeting the demands of real-world applications. Moreover, to mitigate the computational overhead of influence estimation, which is intrinsically #P-hard, learning-based IM methods often resort to techniques from traditional methods, like proxy-based and sampling-based estimation. Paradoxically, this approach exacerbates challenges related to scalability and generalization. Lastly, the IM problem features various node-centrality-constrained variants. Beyond merely regulating the seed node budget, scenarios might necessitate the control of the total cost incurred by selecting seed nodes. Learning-based IM solutions tend to employ diverse objective functions tailored to specific application contexts. As a result, there lacks a unified framework for addressing the wide spectrum of node-centrality-related constraints. L3t5Pub1!5 h

## 4. DeepIM

In this section, we introduce the DeepIM framework designed to mitigate the computational complexities associated with learning-based IM methods, while also automating the identification of underlying diffusion patterns. This framework is structured into two distinct phases: The *learning phase*, utilized to capture the probabilities associated with the observed seed set and to model the distribution governing the propagation of information within the network. The *inference phase*, employed to optimize the selection of seed nodes within a continuous space, with the ultimate objective of maximizing the spread of influence. This two-phase structure of the DeepIM framework provides a systematic approach to address both computational efficiency and diffusion pattern modeling in the context of IM.

### 4.1. Learning representation of seed set

To construct an effective and efficient objective function, our approach involves the characterization of the probability distribution $p(\mathbf{x})$ across the space of seed node sets $\mathbf{x}$, with respect to the graph $G$. This strategic pursuit of learning $p(\mathbf{x})$ serves to uncover the intrinsic characteristics of the seed sets. However, this endeavor is no straightforward feat, mainly due to the intricate interconnections among nodes within each seed set and the high degree of correlation influenced by the topology of the graph $G$. These intricate interdependencies render the relationships between nodes exceedingly complex, rendering the task hard to decipher than other similar combinatorial problems.

#### 4.1.1. Learning probability over seed nodes.

Rather than attempting to model the highly intricate probability distribution $p(\mathbf{x})$ directly, we introduce a latent variable $z$ to serve as a representative of $\mathbf{x}$. This latent variable allows us to define a conditional distribution $p(\mathbf{x}|z)$ that quantifies the likelihood of observing $\mathbf{x}$ given $z$. The latent variables $z$ possess significantly lower dimensions compared to the observed sub-optimal seed sets, leading to a more compressed representation. In particular, we perform a process of marginalization over the latent variables, yielding the expression $p(\mathbf{x}) = \int p(\mathbf{x}, z), dz = \int p(\mathbf{x}|z)p(z), dz$. This integral operation enables us to compute the posterior likelihood $p(z|\mathbf{x}) = p(\mathbf{x}|z) \cdot p(z)/p(\mathbf{x})$, allowing

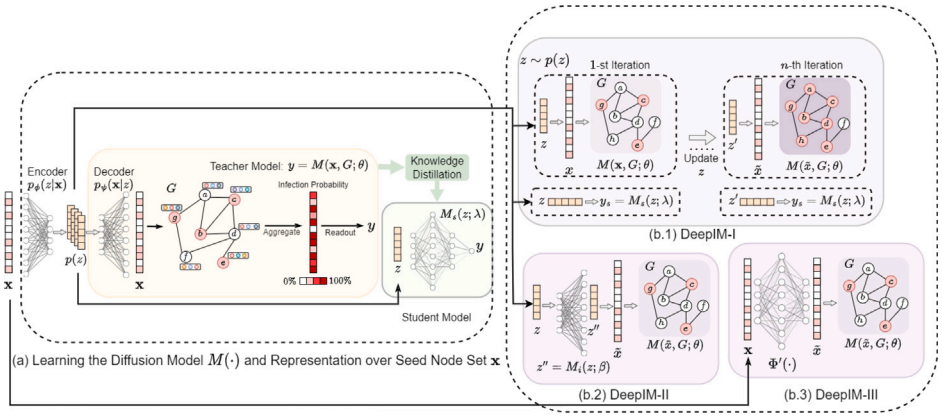



**Fig. 1.** DeepIM consists of two parts. (a) we leverage the autoencoder to learn and compress the latent distribution of seed node sets into lower dimension $p(z)$. The lower dimensional $p(z)$ is then leveraged to learn an end-to-end and monotonic diffusion model $M(\mathbf{x}, G; \theta)$ for accurately predicting the spread. In addition, we employ a knowledge distillation module to train a lightweight student model to retain efficiency in predicting the influence spread. (b) the seed set inference follows three different types of schemes. (b.1) iteratively optimizes the proposed objective function by updating the latent variable $z$, initially sampled from the learned $p(z)$, to maximize the influence spread. (b.2) Utilized the updated latent variable $z'$ during the training session so, during inference, it can skip the iterative process of updating $z$. (b.3) This process trained the network by using the initial seed node set $\mathbf{x}$ and the Graph to get the final seed node set $\tilde{x}$.

us to infer the latent variable $z$ based on the observed seed sets $\mathbf{x}$. In our methodology, we adopt an autoencoder architecture to generatively infer the posterior distribution. This autoencoder employs both an encoder $f_\phi$ (parameterized by $\phi$) and a decoder $f_\psi$ (parameterized by $\psi$) to model the likelihood of both the posterior and the conditional distribution, respectively. The overarching goal of the autoencoder is to maximize the joint likelihood, as encapsulated by the following objective function:

$$\max_{\phi,\psi} : \mathbb{E}\left[p_\psi(\mathbf{x}|z) \cdot p_\phi(z|\mathbf{x})\right]. \tag{3}$$

#### 4.1.2. Learning the end-to-end diffusion model

After acquiring knowledge about the latent distribution of seed nodes, denoted as $p(\mathbf{x})$, the subsequent phase involves updating the seed node set $\mathbf{x}$ to enhance the influence spread's marginal gain. Existing learning-based solutions for influence maximization (IM) still depend on predefined mathematical models to compute the influence spread ($M(\mathbf{x}, G; \theta)$). However, real-world information diffusion is intricate, making it challenging to determine an optimal diffusion model in practice. A selected model might not accurately capture real-world data, leading to substantial model bias. Furthermore, the underlying diffusion network structure can be concealed, necessitating the learning of both diffusion model parameters and network structure (Du et al., 2014).

This research introduces a diffusion model based on Graph Neural Networks (GNNs) denoted as $M(\cdot)$, aimed at precisely modeling the connection between $\mathbf{x}$ and $y$ while considering the global graph topology. The GNN-based diffusion function $M(\cdot)$ comprises two components: (1) $\tau = g_u(\mathbf{x}, G; \theta)$, where $g_u(\cdot)$ represents a GNN-based aggregation function, and $\tau \in [0,1]^{|V|}$ denotes intermediate output after aggregating multi-hop neighborhood information. $\tau$ signifies the "infection probability" of each node. (2) $y = g_r(\tau; \xi)$, where $y \in \mathbb{R}_+$ represents the final information spread, $g_r(\cdot)$ is a normalization function (e.g., $l$-1 norm), and $\xi$ serves as the threshold to convert probability into discrete values. The structure of the GNN-based $M(\cdot)$ is illustrated in Fig. 1(a).

**Definition 2** (***Score Monotonicity and Infection Monotonicity***). For a GNN-based diffusion model $M(\cdot) : 2^{|V|} \to \mathbb{R}_+$ and any two subsets $S, T \subseteq V$, $M(\cdot)$ exhibits score monotonicity if $x_S \preceq x_T$ ($S \subseteq T$) implies $M(\mathbf{x}_S, G; \theta) \leq M(\mathbf{x}_T, G; \theta)$. Here, $\mathbf{x}_S, \mathbf{x}_T \in 0,1^{|V|}$ denote vector representations of seed sets $S$ and $T$, respectively. $M(\cdot)$ demonstrates infection monotonicity if $x_S \preceq x_T$ implies $\tau_S \preceq \tau_T$, where $\tau_S, \tau_T \in [0,1]^{|V|}$ indicate infection probabilities of seed sets $S$ and $T$, respectively. This ensures that $M(\cdot)$ captures the inherent diffusion network structure and emulates real-world diffusion patterns (Dolhansky & Bilmes, 2016).

Monotonicity is a crucial property for modeling the overall diffusion network structure. A monotonic diffusion model signifies that the spread of influence will consistently increase. If a larger community $\mathbf{x}'$ is chosen as the seed set, it naturally infects at least as many nodes across the network as a smaller seed set $\mathbf{x}$ when $\mathbf{x} \preceq \mathbf{x}'$. Enforcing both score and infection monotonicities assists in accurately characterizing the underlying diffusion network structure and mimicking real-world diffusion behavior (Dolhansky & Bilmes, 2016). To ensure these properties, constraints are incorporated into the GNN-based diffusion model $M(\mathbf{x}, G; \theta)$ to maintain monotonicity during influence spread estimation.

**Theorem 1** (***Monotonicity of GNN Models***). *For any GNN-based $M(\mathbf{x}, G; \theta) = g_r \circ g_u(\mathbf{x}, G; \theta)$, where $g_u(\mathbf{x}, G; \theta)$ follows Eq. (1), $M(\cdot)$ is both score and infection monotonic if $\mathcal{A}^k$ and $C^k, k \in [1, K]$, are non-decreasing according to Eq. (1), and $g_r$ is also non-decreasing.*

**Proof.** The GNN model is structured as $g_u(x, G; \theta) = \mathcal{A}^1 \circ (C^1 \circ \mathcal{A}^2 \circ C^2 \cdots \circ \mathcal{A}^K \circ C^K)$ through recursive iteration of Eq. (1). Since $\mathcal{A}^k$ and $C^k$ are both non-decreasing, the composition $\mathcal{A}^1 \circ C^1 \cdots \circ \mathcal{A}^K \circ C^K$ is also non-decreasing, constituting $g_u$. This establishes the infection monotonicity of $M$. Additionally, due to the non-decreasing nature of $g_u$ and $g_r$, $M$ itself is non-decreasing, ensuring score monotonicity. □

Furthermore, the Graph Attention Network (GAT), a well-known neural network architecture, is demonstrated to exhibit both score and infection monotonicity under specified constraints, as articulated in Corollary 2.

**Corollary 2** (***Monotonicity of GAT***). *$M$ is score and infection monotonic when $g_u$ adopts GAT, assuming $\theta^k \geq 0$ in Eq. (1), and $g_r$ is also non-decreasing.*

**Proof.** In the GAT model, $h_i^k = C^k(a^k, \theta^k) = \sum j \in N(i)\ softmax(a^k i, j)\theta_1^k h^{k-1} j$ and $a_{i,j}^k = \mathcal{A}^k(h_i^k, h_j^k, \theta^k) = \theta_1^k(\theta_2^k h^{k-1} i || \theta_2^k h^{k-1} j)$. Because $\theta^k \geq 0$ (i.e., $\theta_1^k \geq 0$ and $\theta_2^k \geq 0$), $\mathcal{A}^k$ and $a^k i, j$ are non-negative. This property extends to $h_i^k$ due to the non-negativity of the softmax operator and $\theta_1^k$. Therefore, the LeakyReLU and $\max(\bullet, 0)$ operations in $C^k$ can be removed, simplifying $h_i^k$ to $\sum_{j \in N(i)} softmax(a_{i,j}^k)\theta_1^k h_j^{k-1}$. Given the non-decreasing nature of softmax and $\theta_1^k$, $C^k$ is non-decreasing. This complies with the conditions outlined in Theorem 1, establishing score and infection monotonicity for the GAT model. □

In accordance with Theorem 1 and Corollary 2, the GNN-based $M(\mathbf{x}, G; \theta)$ maintains monotonicity, and the objective of learning this GNN-based $M(\mathbf{x}, G; \theta)$ is framed as maximizing the probability below while adhering to the constraint $\theta \geq 0$:

$$\max_{\theta} \mathbb{E}\big[p_\theta(y|\mathbf{x}, G)\big]. \tag{4}$$

*4.1.3. Knowledge distillation for diffusion estimation efficiency.*

In our research, we have successfully acquired a profound understanding of the latent representations associated with seed nodes. Moreover, we have developed an end-to-end diffusion model that provides a strong guarantee of monotonicity. Despite these achievements, our empirical analysis has revealed that the process of estimating influence spread, denoted as $M(\mathbf{x}, G; \theta)$, involves a sequence of three key steps: (1) The initial decoding of a node vector $\mathbf{x}$ through the utilization of the learned posterior distribution $p(\mathbf{x}|z)$. (2) The subsequent execution of the GNN-based diffusion model $M(\mathbf{x}, G; \theta)$ within the graph $G$. (3)The normalization of the probabilistic output $\tau$ generated by $M(\mathbf{x}, G; \theta)$ to arrive at the final, actual influence spread value denoted as $y$. While the accuracy of the prediction outcomes is satisfactory, it is evident that the computational overhead associated with this procedure can pose a significant challenge, particularly when dealing with large-scale networks encompassing millions of nodes. Drawing inspiration from recent advancements in the field of knowledge distillation, we introduce an innovative approach aimed at enhancing computational efficiency while preserving the accuracy of influence spread estimation.Our proposal involves leveraging a compact yet potent "student model" that operates under the guidance of the original $M(\mathbf{x}, G; \theta)$ model. This student model, denoted as $M_s(z; \lambda)$, is a lightweight neural network that is parameterized by $\lambda$. It directly accepts the latent variable $z$ sampled from the learned distribution $p(z)$ as input and produces an estimate of the influence spread, denoted as $y_s$, as its output. To ensure effective training of the student model, we introduce a distillation loss that quantifies the dissimilarity between the predictions generated by the "teacher model" ($y = M(\mathbf{x}, G; \theta)$) and those produced by the student model ($y_s = M_s(z; \lambda)$). This distillation loss is computed as the squared Euclidean distance between the predicted influence spread values, expressed as $\|y - y_s\|_2^2$.

In essence, our approach offers a solution to improve the efficiency of influence spread estimation through knowledge distillation. By introducing a lightweight yet capable student model that operates in conjunction with the original diffusion model, we aim to reduce the computational burden associated with large-scale networks, thereby enabling more efficient and expedited inference processes.

*4.1.4. End-to-end learning objective.*

To seamlessly integrate representation learning and the acquisition of the diffusion model, we propose a unified end-to-end objective function that combines the expressions from Eqs. (3) and (4). This objective function is defined as follows:

$$\mathcal{L}_{\text{train}} = \max_{\theta,\lambda,\psi,\phi} \mathbb{E}\big[p_\theta(y|\mathbf{x}, G) \cdot p_\lambda(y_s|z) \cdot p_\psi(\mathbf{x}|z) \cdot p_\phi(z|\mathbf{x})\big], \text{ s.t. } \theta \geq 0. \tag{5}$$

However, dealing with the optimization of joint probabilities within the expectation can pose computational challenges. To address this, we derive the negative logarithm of Eq. (5) and establish a lower bound for the final learning objective using Jensen's inequality in Eq. (6). The overall learning objective comprises three components: (1) Minimizing the empirical error $-\log[p_\theta(y|\mathbf{x}, G)]$ associated with predicting $y$ using the reconstructed $\mathbf{x}$ as input. (2) Minimizing the reconstruction error, encompassing $-\log[p_\psi(\mathbf{x}|z)]$, for effective learning of the seed set distribution. (3) Minimizing the distillation loss $-\log[p_\lambda(y_s|z)]$ to facilitate the training of the student model in parallel with the comprehensive training process.

$$\begin{aligned}\mathcal{L}_{\text{train}} &= \min_{\theta,\lambda,\psi,\phi} -\log\Big[\mathbb{E}\big[p_\theta(y|\mathbf{x}, G) \cdot p_\lambda(y_s|z) \cdot p_\psi(\mathbf{x}|z) \cdot p_\phi(z|\mathbf{x})\big]\Big], \text{ s.t. } \theta \geq 0. \\ &\geq \min_{\theta,\lambda,\psi,\phi} \mathbb{E}\Big[-\log\big[p_\theta(y|\mathbf{x}, G) \cdot p_\lambda(y_s|z) \cdot p_\psi(\mathbf{x}|z) \cdot (p_\phi(z|\mathbf{x}))\big]\Big], \text{s.t. } \theta \geq 0. \\ &= \min_{\theta,\lambda,\psi,\phi} \mathbb{E}\Big[-\log\big[p_\theta(y|\mathbf{x}, G)\big] - \log\big[p_\lambda(y_s|z)\big] - \log\big[p_\psi(\mathbf{x}|z) \cdot (p_\phi(z|\mathbf{x}))\big]\Big], \text{s.t. } \theta \geq 0.\end{aligned} \tag{6}$$

The overarching framework for training end-to-end diffusion models, coupled with the autoencoder for seed set distribution learning, is visualized in Fig. 1(a).

### 4.2. Seed node set inference

To infer the set of highly influential seed nodes within the testing domain, we capitalize on the joint utilization of the latent distribution $p(\mathbf{x})$ pertaining to the seed node set and the end-to-end diffusion model $M(\cdot)$ as defined in Eq. (6). In this context, two key attributes of the autoencoder play a crucial role: *continuity*, where nearby points in the latent space should yield somewhat similar content upon decoding, and *completeness*, which ensures that a point sampled from the latent space within a chosen distribution produces meaningful content upon decoding.

Given that the autoencoder in Eq. (3) is well-trained and possesses both continuity and completeness, it can generate content by leveraging the latent feature space $p(z)$ learned from all the examples it was trained on, denoted as $p(\mathbf{x})$. In light of this, we introduce an innovative approach of searching for the optimal seed node set $\tilde{x}$ within the lower-dimensional and less-noisy latent space $p(z)$. The ensuing corollary serves to demonstrate the equivalence of estimating the influence spread utilizing the latent variable $z$ instead of the high-dimensional $\mathbf{x}$, contingent upon the autoencoder effectively upholding both continuity and completeness principles.

**Corollary 3** (***Influence Estimation Consistency***). *For any $M(f_\psi(z^{(i)}), G; \theta) > M(f_\psi(z^{(j)}), G; \theta)$, we have $M(x^{(i)}, G; \theta) > M(x^{(j)}, G; \theta)$.*

**Proof.** According to Theorem 1, the GNN-based $M(x, G; \theta)$ is monotonic. Then for any two $x^{(i)} > x^{(j)}$, we have $M(x^{(i)}, G; \theta) > M(x^{(j)}, G; \theta)$. If the reconstruction error is minimized during the training of $f_\psi(\cdot)$, we also have $f_\psi(z^{(i)}) > f_\psi(z^{(j)})$. Hence, $M(f_\psi(z^{(i)}), G; \theta) > M(f_\psi(z^{(j)}), G; \theta)$ also holds. □

As per the corollary, we are enabled to identify the optimal seed set that can yield maximal influence by optimizing over the variable $z$ within the subsequent joint probability expression: $\max_z \mathbb{E}\big[p_\theta(y|\mathbf{x}, G) \cdot p_\psi(\mathbf{x}|z)\big]$.

**Adaptation to Different Influence Maximization Variants with Node Centrality Constraints.** Over the years, Influence Maximization (IM) has been explored in various budget-constrained scenarios on nodes since its inception in Kempe et al. (2003). To enhance the versatility of DeepIM, we introduce a unified constraint framework capable of inferring seed sets across diverse node budgets. Specifically, the objective $\mathcal{L}_{\text{pred}}$ is defined as follows:

$$\mathcal{L}_{\text{pred}} = \max_{z} \mathbb{E}\big[p_\theta(y|\mathbf{x}, G) \cdot p_\psi(\mathbf{x}|z)\big] \qquad \text{s.t.} \sum_{i=0}^{|V|} \mathcal{F}(v_i, G) \cdot x_i \leq k, \tag{7}$$

Here, $\sum^{|V|} i = 0\mathcal{F}(v_i, G) \cdot x_i$ represents a generalized node-wise budget constraint, and $k$ is the actual budget available. For the standard IM problem that entails selecting a given count of seed nodes, $\sum^{|V|} i = 0\mathcal{F}(v_i, G) \cdot x_i$ can be computed as $\|x \cdot \mathbf{1}\|1$, where $\mathbf{1} \in 1^{N\times 1}$ denotes an all-one vector, reflecting uniform selection costs for each node. Additionally, in the context of IM problems involving node degree constraints (Leskovec et al., 2007; Nguyen, Thai, & Dinh, 2017), $\sum^{|V|} i = 0\mathcal{F}(v_i, G) \cdot x_i$ can be formulated as $\|x \cdot A\|_1$, with $A \in 0, 1^{N\times N}$ as the adjacency matrix of network $G$, and $\|\mathbf{x} \cdot A^i\|_1 \leq k$ representing an upper limit on the total seed node degree indicated by the $l1$-norm and constrained by budget $k$. This budget constraint $\mathcal{F}(v_i, G) \cdot x_i \leq k$ can further be conveniently adapted and integrated to tackle IM variants encompassing varying node costs. Through this versatile constraint

formulation, the adaptability challenges faced by IM methods can be effectively tackled.

#### 4.2.1. Implementation by iterative process. (DeepIM-I)

We present our inference procedure through the schematic shown in Fig. 1(b). In this framework, the inference process begins by sampling a latent variable $z$ from the learned distribution $p(z)$. This latent variable $z$ is then progressively optimized based on the inference objective function, Eq. (7), to maximize the marginal gain in influence spread. It is important to note that the diffusion model for learning $p_\theta(y|\mathbf{x}, G)$ can be switched between the student diffusion model $M_s(z;\lambda)$ and the GNN-based diffusion model $M(\mathbf{x}, G;\theta)$ to prioritize either efficiency or effectiveness. Additionally, since the constrained objective function in Eq. (7) is not directly computable, we offer a practical variant of the inference objective function. Given that the observed influence $y$ follows a Gaussian distribution and the seed set $\mathbf{x}$ conforms to a Bernoulli distribution, we can simplify Eq. (7) as follows:

$$\mathcal{L}_{\text{pred}} = \min_z \Big[ -\log\Big[\prod_{i=0}^{|V|} f_\psi(z_i)^{x_i}(1-f_\psi)(z_i)^{1-x_i}\Big] + \Big\|\tilde{y}-y\Big\|_2^2\Big] \qquad \text{s.t.} \sum_{i=0}^{|V|} \mathcal{F}(v_i, G)\cdot x_i \le k, \tag{8}$$

Here, $\tilde{y}$ is set to the optimal influence spread ($\tilde{y} = |V|$). The derivation for Eq. (8) is provided below:

$$\mathcal{L}_{\text{pred}} = \min_z \mathbb{E}\big[-\log p_\theta(y|x, G) - \log p_\psi(x|z)\big], \qquad \text{s.t.} \sum_{i=0}^{|V|} \mathcal{F}(v_i, G)\cdot x_i \le k \tag{9}$$

Since we consider $\tilde{y} = |V|$ as optimal and $y$ is predicted in the range $[0, |V|]$, the optimization goal is to maximize $y$ until complete infection is reached. Consequently, the first term in Eq. (8) can be expressed as Mean Squared Loss (MSE): $\|\tilde{y} - M(x, G;\theta)\|2^2$. As for the second term in Eq. (8), the value range of $x$ after the autoencoder transforms to $[0, 1]$, representing the probability of node selection. Since $x \in [0, 1]$ conforms to the binomial distribution, minimizing the negative log-likelihood is equivalent to minimizing the probability mass function. Thus, the second term can be reduced to $-\log\big[\prod_i^{|V|} f\psi(z_i)^{x_i}(1 - f_\psi)(z_i)^{1-x_i}\big]$. The combination of both terms yields the final form of Eq. (8):

$$\mathcal{L}_{\text{pred}} = \max_z \mathbb{E}\big[p_\theta(y|x, G)\cdot p_\psi(x|z)\big] \qquad \text{s.t.} \sum_{i=0}^{|V|} \mathcal{F}(v_i, G) \le k. \tag{10}$$

Furthermore, we employ Projected Gradient Descent and introduce a regularization function $\Phi(\mathbf{x})$ to keep the predicted seed set $\mathbf{x}$ within a valid region, considering various constraints. For instance, when dealing with Eq. (7), $\Phi(\mathbf{x})$ can be defined as selecting the top-$k$ nodes with the highest probabilities. For problems involving non-uniform node costs, $\Phi(\mathbf{x})$ can be designed to cost-efficiently select the top-$k$ nodes from $\mathbf{x}/c(\mathbf{x})$, where $c(\mathbf{x})$ signifies the cost of selecting one node (e.g., node degree). In Algorithm 1, we outline the optimization procedure. The algorithm initializes by sampling an initial latent variable $z$ (Line 1) and subsequently iteratively solves the optimization problem from Eq. (7) using a gradient descent optimizer (e.g., Adam) while ensuring the predicted seed set is constrained within valid bounds through $\Phi(\cdot)$ (Line 2–6). Specifically, in order to ensure the validity of the predicted $x$ during optimization, we regulate the value of $x$ in the range of $[0, 1]$ by using $trim(x, 0, 1)$ on Line 4. The overall process of learning the inference objective is depicted in Fig. 1(b).

#### 4.2.2. Implementation by supervised manner (DeepIM-II)

In the iterative process outlined in Section 4.2.1, we obtain an updated representation of $z$ in Line 5 of Algorithm 1, which is also illustrated in Fig. 1 (b.1) and denoted as $z'$. Despite achieving accurate prediction results, the computational overhead remains significant due to the iterative nature of the process. To address this challenge, we draw inspiration from recent advances in knowledge distillation. We introduce a compact yet potent student model $M_i(z;\beta)$, which is a lightweight neural network parameterized by $\beta$. This model takes the latent variable $z$ sampled from the learned distribution $p(z)$ as input and directly produces the updated latent representation $z''$ as output. The distillation loss, in this context, can be elegantly formulated as follows:

$$min_{M_i} \|z' - M_i(z;\beta)\|_2^2 \tag{11}$$

Here, the objective is to minimize the squared Euclidean distance between the original updated latent representation $z'$ and the one produced by the student model $z''$. This approach captures the essence of the iterative process within a lightweight and efficient neural network model, effectively reducing the computational burden while retaining the benefits of accurate prediction.

**Algorithm 1:** DeepIM-I Prediction Framework

**Input:** $\mathcal{L}_{\text{pred}}$; $f_\psi(\cdot)$; $\Phi(\cdot)$; number of training instances $N$; the number of iteration $\eta$; learning rate $\alpha$.

1: $z = 1/N \sum_{i=0}^{N} f_\psi(\mathbf{x})$ {$\mathbf{x}$ sampled from training set.}
2: **for** $i = 0, ..., \eta$ **do**
3: $\quad \mathbf{x} \leftarrow f_\psi(z)$ {seed set $\mathbf{x}$.}
4: $\quad \mathbf{x} \leftarrow \Phi(\mathbf{x})$ {Regularize $\mathbf{x}$ into valid regions.}
5: $\quad z \leftarrow z - \alpha \cdot \nabla \mathcal{L}_{\text{pred}}(\mathbf{x}, z)$
6: **end for**
7: $\tilde{\mathbf{x}} \leftarrow \Phi(f_\psi(z))$

#### 4.2.3. Implementation by unsupervised manner (DeepIM-III)

In the preceding Section 4.2.1, we utilized the latent distribution $p(\mathbf{x})$ of the seed node-set and the end-to-end diffusion model $M(\cdot)$ to formulate the function $\Phi(\cdot)$ for searching the optimal node set $\tilde{x}$. However, due to the iterative nature of this process ($\Phi(\cdot)$), testing becomes time-consuming and computationally intensive, especially when dealing with large-scale networks. To overcome the issue, we proposed another inference technique in 4.2.2. It helped us to eradicate the iteration process during the testing inference phase. But, The training of $M_i(z;\beta)$ is supervised learning which depends highly on the updated $z'$ from the iteration process. Moreover, we need to depend on $M_i(z;\beta)$ to update the latent distribution $z \sim p(x)$, and the decoder $f_\psi$ to update the seed node during testing inference. We introduce a new function $\Phi'(\cdot)$ that not only eliminates the need for iteration ($\Phi(\cdot)$) during the testing phase but also addresses the dependency on the latent distribution $p(x)$. Instead of depending on two functions namely $\Phi(\cdot)$, $f_\psi$ in Section 4.2.1, or $M_i(z;\beta)$, $f_\psi$ in Section 4.2.2; it depends on only $\Phi'(\cdot)$ during testing inference.

This function, $\Phi'(\cdot)$ utilizes the initial vector representation of the source node set $x$ and the graph $G$ to capture the latent distribution $p(x, G)$. $\Phi'(\cdot)$ is trained in an unsupervised manner, effectively replacing the iterative process for selecting the optimal seed node $\tilde{x}$ during testing. To achieve this, we replace the supervised loss in Eq. (11) with the direct goal of our influence maximization, whose gradient can always guide the solution $\tilde{x}$ toward local optima. We train a network of the form $p(y|x, G) = p(y|\mathbf{r}[\Phi'(x;\varphi), M(\tilde{x}, G;\theta)])$, where $\tilde{x} = \Phi'(x;\varphi)$. Here, $y$ represents the total number of infected nodes, capturing the relationship between $x$ and $(\tilde{x}, G)$ predicted by $\mathbf{r}(\Phi'\varphi, M_\theta)$. The term $M(\tilde{x}, G;\theta)$ denotes a controlled transformation of $\tilde{x}$ through the diffusion model. The objective function $\mathcal{L}_{\text{proxy}}$ is given as follows:

$$\begin{aligned}\mathcal{L}_{\text{proxy}} &= \max_{x,G} \mathbb{E}\big[p_\varphi(\mathbf{x})\cdot p_\theta(y|\tilde{x}, G)\big] \\ &= \min_x \Big[-\log\Big[\prod_{i=0}^{|V|} \Phi'_\varphi(x_i)(1-\Phi'_\varphi(1-x_i))\Big] + \Big\|\tilde{y}-y\Big\|_2^2\Big]\end{aligned} \tag{12}$$

In Eq. (12), $\tilde{y}$ represents the optimal influence spread, which in this case is $|V|$. The objective is to maximize the relationship between $x$ and $(\tilde{x}, G)$ as predicted by the combined network $\mathbf{r}(\Phi'\varphi, M\theta)$. The first term in Eq. (12) represents the negative log-likelihood of $\Phi'_{\varphi}(x_i)$, which captures the probability distribution of selecting nodes. The second term involves minimizing the mean squared error between the predicted influence spread ($y$) and the optimal influence spread ($\tilde{y}$). This approach effectively eliminates the need for iterative testing, making the process more efficient and scalable.

**Algorithm 2:** DeepIM-III Prediction Framework

**Input:** $\mathcal{L}_{\text{proxy}}$; $\Phi_{\varphi}(\cdot)$; $M_{\theta}(\cdot)$; number of training instances $N$; Graph $G$; learning rate $\alpha$.

1: $\tilde{x} = 1/N \sum_{i=0}^{N} \Phi_{\varphi}^{prime}(\mathbf{x})$ {$\mathbf{x}$ sampled from training set.}
2: $y \leftarrow M_{\theta}(\tilde{x})$ {infected node set $\mathbf{y}$.}
3: **if** *fine tuning* **then**
4: $\tilde{x} \leftarrow \tilde{x} - \alpha \cdot \nabla \mathcal{L}_{\text{pred}}(\mathbf{x}, G)$
5: $\tilde{\mathbf{x}} \leftarrow \Phi_{\varphi}^{prime}(\mathbf{x})$)

## 5. Experiment

In this section, we conduct a comprehensive evaluation of the performance of our proposed DeepIM framework across six real-world networks, aiming to maximize the influence under various application scenarios. We also present case studies to qualitatively demonstrate the effectiveness of DeepIM. We provided experiments with budget constraints and graph diffusion visualization.

### 5.1. Datasets

We compare the performance of DeepIM with other existing approaches using six real-world datasets: Cora-ML, Network Science, Power Grid, Jazz, Digg, and Weibo. Additionally, we employ a synthetic dataset generated using the Erdos–Renyi algorithm with 50,000 nodes. Dataset statistics are presented in Table 1, along with detailed descriptions of each dataset: *Jazz* (Rossi & Ahmed, 2015): This dataset represents a collaboration network among Jazz musicians, where nodes correspond to musicians and edges indicate collaborative performances. *Cora-ML* (McCallum, Nigam, Rennie, & Seymore, 2000): A network of computer science research papers, nodes represent papers, and edges represent citations between papers. *Power Grid* (Rossi & Ahmed, 2015): A topology network of the US Western States Power Grid, where nodes represent generators, transformers, or substations, and edges represent power supply lines. *Network Science* (Rossi & Ahmed, 2015): A coauthorship network of scientists in network theory, with nodes representing scientists and edges indicating collaboration. *Digg* (Panagopoulos et al., 2020): A directed social media network, where users follow each other and vote on posts to share them. *Weibo* (Panagopoulos et al., 2020): A directed follower network on a microblogging platform, where a cascade is initiated by the first tweet and subsequent retweets contribute to the spread. In our experiments, we randomly sample seed node sets $\mathbf{x}$, with the seed size proportional to the total number of nodes $|V|$ in each network. We employ the IC, LT, and SIS models to calculate the influence spread $y$. The resulting set of pairs $(\mathbf{x}, y)$ serves as the training set for our DeepIM.

### 5.2. Experimental setup

Our main objective is to assess the expected influence spread, as defined in Eq. (2), across various scenarios in the context of Influence Maximization (IM). DeepIM is designed to be versatile and adaptable to different diffusion patterns. To illustrate this adaptability, we consider two widely used IM models, namely the Linear Threshold (LT) model and the Independent Cascade (IC) model. Additionally, we evaluate the IM problem under the susceptible–infectious–susceptible (SIS) epidemic model (Kermack & McKendrick, 1927), which allows activated nodes to become de-activated over time. In our experimentation, we meticulously set hyperparameters based on the original papers of each baseline method. We further fine-tune these hyperparameters on each individual dataset to ensure optimal performance. Below, we provide details about the configurations of different diffusion models used in our evaluations:

- *IC Model*: We employ a weighted cascade version of the Independent Cascade (IC) model. Specifically, the propagation probability $p_{u,v}$ for each edge $e = (u, v)$ on graph $G$ is set to $1/d_v^{in}$, where $d_v^{in}$ signifies the in-degree of node $v$.
- *LT Model*: For the Linear Threshold (LT) model, we set the threshold $\theta$ to be uniformly sampled from the interval $[0.3, 0.6]$ for each node $v$.
- *SIS Model*: In the Susceptible–Infectious–Susceptible (SIS) epidemic model, we establish the infection probability and recovery probability as 0.001.

**Table 1**
The overview of dataset.

| | Digg | Weibo | Power grid | Network science | Cora-ML | Jazz | Synthetic |
|---|---|---|---|---|---|---|---|
| Nodes | 279,613 | 2,251,166 | 4,941 | 1,565 | 2,810 | 198 | 50,000 |
| Edges | 1,170,689 | 225,877,808 | 6,594 | 13,532 | 7,981 | 2,742 | 250,000 |

As for DeepIM, we utilize a 2-layer Graph Attention Network (GAT)-structured diffusion estimation model. Each layer comprises 4 attention heads, with a dimension of 64 for each attention channel. Both the encoder and decoder are symmetric 4-layer Multi-Layer Perceptrons (MLPs), with hidden sizes of 512, 1024, 1024, and 1024 for each respective layer. We employ the Adam optimizer, setting learning rates of 0.001 for optimizing Eq. (6) and 0.0001 for optimizing Eq. (7). Our hyperparameter choices are guided by both the specific characteristics of each dataset and the requirements of the respective models. This approach ensures that we achieve the most effective and optimal performance for each experimental scenario.

### 5.3. Comparison method.

In our comparative analysis, we assess the performance of our proposed models against a comprehensive set of baseline methods, which are categorized as: *Traditional IM*: IMM (Tang et al., 2015) OPIM-C (Tang, Tang, Xiao, & Yuan, 2018) SubSIM (Guo, Wang, Wei, & Chen, 2020) *Learning-based IM*: IMINFECTOR (Panagopoulos et al., 2020) PIANO (Li et al., 2022) ToupleGDD (Chen et al., 2022) *Online IM*: OIM (Lei, Maniu, Mo, Cheng, & Senellart, 2015) *Budget-constraint IM*: CELF (Leskovec et al., 2007) In our evaluations, we compare the performance of four variants of our proposed DeepIM model: DeepIM-I, which employs the GNN-based diffusion model $M(x, G; \theta)$ with iterative inference. DeepIM$_s$-I, which utilizes the student diffusion model $M_s(z; \lambda)$ with iterative inference. DeepIM-II, which employs the GNN-based diffusion model $M(x, G; \theta)$ with knowledge distillation inference. DeepIM-III, which employs the GNN-based diffusion model $M(x, G; \theta)$ with proxy task inference. This comprehensive comparison allows us to assess the effectiveness and efficiency of our proposed methods across a diverse range of scenarios, spanning traditional, learning-based, online, and budget-constraint influence maximization problems. By including both the GNN-based and student diffusion models, we can capture a comprehensive understanding of the trade-offs between accuracy and computational efficiency.

### 5.4. Quantitative analysis

We conduct a thorough evaluation of DeepIM's influence maximization performance in comparison to other approaches across various

**Table 2**
Performance comparison under IC diffusion pattern. – indicates out-of-memory error.

| Methods | Cora-ML | | | | Network Science | | | | Power Grid | | | | Jazz | | | | Synthetic | | | | Digg | | | | Weibo | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | 1% | 5% | 10% | 20% | 1% | 5% | 10% | 20% | 1% | 5% | 10% | 20% | 1% | 5% | 10% | 20% | 1% | 5% | 10% | 20% | 1% | 5% | 10% | 20% | 1% | 5% | 10% | 20% |
| IMM | 8.1 | 26.2 | 37.3 | 50.2 | 5.2 | 16.8 | 27.0 | 45.7 | 4.3 | 17.4 | 31.5 | 51.1 | 2.6 | 20.1 | 31.4 | 42.8 | 9.2 | 26.2 | 36.3 | 51.6 | 7.4 | 18.4 | 32.8 | 49.6 | 9.5 | 23.8 | 36.4 | 50.5 |
| OPIM | 13.4 | 26.9 | 37.4 | 50.9 | 6.6 | 19.4 | 28.9 | 48.6 | 5.7 | 17.7 | 29.7 | 50.1 | 2.4 | 20.1 | 34.4 | 46.8 | 9.6 | 25.3 | 36.6 | 51.7 | 7.6 | 18.5 | 32.9 | 48.9 | 9.7 | 23.7 | 36.6 | 50.3 |
| SubSIM | 10.1 | 25.7 | 36.8 | 51.1 | 4.8 | 15.4 | 27.9 | 44.8 | 4.6 | 19.2 | 31.7 | 50.2 | 3.6 | 18.8 | 37.6 | 44.7 | 9.5 | 26.7 | 36.5 | 51.5 | 7.5 | 18.9 | 33.3 | 49.4 | 9.3 | 23.1 | 36.5 | 50.6 |
| OIM | 8.9 | 27.6 | 38.0 | 51.3 | 4.2 | 16.7 | 26.5 | 48.2 | 5.7 | 17.5 | 31.9 | 50.8 | 2.0 | 18.5 | 36.3 | 42.2 | 9.6 | 26.2 | 36.7 | 51.3 | 7.8 | 18.2 | 33.1 | 49.6 | – | – | – | – |
| IMINFECTOR | 9.6 | 26.8 | 37.7 | 50.6 | 5.4 | 17.9 | 27.8 | 47.6 | 5.4 | 18.2 | 31.6 | 50.9 | 3.6 | 19.7 | 37.5 | 45.9 | 9.1 | 26.2 | 36.1 | 51.5 | 7.9 | 18.6 | 33.5 | 49.8 | 9.4 | 23.5 | 36.9 | 50.3 |
| PIANO | 9.8 | 25.2 | 37.4 | 51.1 | 4.7 | 16.3 | 27.1 | 47.2 | 5.3 | 18.1 | 31.7 | 50.2 | 2.2 | 19.2 | 36.6 | 43.2 | 9.1 | 26.4 | 36.2 | 51.6 | – | – | – | - | – | – | – | – |
| ToupleGDD | 10.6 | 27.5 | 38.5 | 51.5 | 6.3 | 17.8 | 28.3 | 50.5 | 5.4 | 19.3 | 31.6 | 51.3 | 3.3 | 20.4 | 37.2 | 45.7 | 9.5 | 26.8 | 37.1 | 51.4 | – | – | – | - | – | – | – | – |
| DeepIM$_s$-I | 13.6 | 27.7 | 38.5 | 51.8 | 6.9 | 19.1 | 29.3 | 50.5 | 5.9 | 20.2 | 31.7 | 51.5 | 3.8 | 21.4 | 38.9 | 47.1 | 10.2 | 26.8 | 37.5 | 51.8 | 7.9 | 18.8 | 33.7 | 50.3 | 10.1 | 24.7 | 36.8 | 50.8 |
| DeepIM-I | **14.1** | 28.1 | **39.6** | **52.4** | **7.8** | **20.9** | **31.5** | **51.2** | 6.3 | 21.0 | 32.5 | 52.4 | 4.9 | 23.3 | **41.5** | 49.9 | 11.6 | 27.4 | 38.7 | 52.1 | 8.4 | 19.3 | **34.2** | 51.3 | 11.2 | 26.5 | 37.9 | **51.8** |
| DeepIM-II | 14.08 | **30.71** | 35.54 | 50.53 | 6.75 | 18.89 | 29.74 | 44.16 | 10.03 | **21.96** | **32.62** | 49.56 | 5.01 | **26.21** | 33.03 | 49.95 | 11.82 | 27.23 | 38.91 | 52.15 | **8.6** | **19.5** | **34.2** | **51.7** | 11.5 | **26.8** | **38.1** | **51.8** |
| DeepIM-III | 11.22 | 26.05 | 35.84 | 49.47 | 7.51 | 17.90 | 30.15 | 43.98 | **10.87** | 21.21 | 32.27 | **49.59** | **6.52** | 20.96 | 35.45 | **51.01** | **11.91** | **28.10** | **39.58** | **53.43** | 8.1 | 19.3 | 33.7 | 50.8 | **11.8** | **26.8** | 37.9 | 51.6 |

influence maximization scenarios. For each dataset, we consider different seed node percentages: 1%, 5%, 10%, and 20% of the total nodes. We allow each diffusion model to simulate until the diffusion process naturally halts, and we record the average influence spread over 100 simulation runs. The performance metric we use is the percentage of final infected nodes, which is calculated as the ratio of the number of infected nodes to the total number of nodes in the network. This metric provides insight into the effectiveness of each approach in propagating influence within the network.

#### 5.4.1. IM under IC model.

We begin by examining the effectiveness of DeepIM compared to other baseline methods under the IC diffusion pattern. The results in Table 2 illustrate that DeepIM consistently outperforms other methods across all datasets. Among the traditional methods, IMM, OPIM, and SubSIM employ reserve-set sampling and various approximation techniques, yielding similar results across datasets. However, these methods rely on heuristics for guiding node selection, which may hinder their ability to decode the underlying distribution of seed sets. OIM achieves superior performance compared to traditional methods in most datasets, owing to its automatic iterative edge weight updating mechanism. Nonetheless, OIM's drawback is evident: it is tailored to the specific IC diffusion model, limiting its applicability in real-world scenarios. Learning-based IM methods (IMINFECTOR, PIANO, and ToupleGDD) achieve competitive and often superior results compared to traditional approaches due to their larger model sizes and enhanced generalization capabilities. However, learning-based methods that rely on reinforcement learning encounter scalability challenges and struggle to handle billion-scale networks (e.g., Digg and Weibo). Consequently, these methods are less feasible for real-world applications. In contrast, DeepIM-I offers a robust approach by learning the end-to-end diffusion model and directly searching for high-influential node sets in the latent space. This enables DeepIM-I to better capture underlying diffusion dynamics and address scalability issues. Additionally, DeepIM$_s$-I introduces a lightweight end-to-end diffusion model and DeepIM-II, DeepIM-III introduce two more inference approach for DeepIM-I. These strike a balance between efficacy and efficiency that surpasses other learning-based methods.

#### 5.4.2. IM under LT model.

Next, we evaluate the final influence spread while varying the initial seed set size, assuming the LT diffusion model. The results in Table 3 highlight that DeepIM consistently outperforms other methods, achieving a significant advantage across all datasets. Notably, DeepIM's superiority is especially evident in the Synthetic dataset, where selecting 20% of nodes as the initial seed set results in spreading influence to the entire network. In contrast, other methods achieve infection rates of at most 70% of the nodes in the network. Particularly striking is DeepIM's performance in the Jazz dataset, where DeepIM-I and DeepIM$_s$-I surpass other methods by an average of 200% in influence spread. Similarly, in the Synthetic dataset, DeepIM-I and DeepIM$_s$-I outperform other methods by approximately 30%. On the other hand, our faster inference approaches introduced in DeepIM-II and DeepIM-III also provide competitive results. This success can be attributed to DeepIM's robust generalization capability across various diffusion models, setting it apart from other methods that struggle to maintain effectiveness under diverse diffusion scenarios.

#### 5.4.3. IM under non-progressive diffusion model.

We proceed to demonstrate the performance of each model under the non-progressive SIS model, as presented in Table 4. A notable observation is the substantial reduction in performance concerning the final influence spread when compared to the results in Table 2 and Table 3. This decrease can be attributed to the inherent complexity of the SIS diffusion model, which accounts for the possibility of nodes transitioning from an activated to a deactivated state with a certain probability. Despite the challenges posed by the SIS diffusion dynamics, DeepIM maintains its competitive edge. It outperforms other methods by an average of 10% across all datasets, showcasing its ability to adapt to intricate diffusion scenarios. This robustness is a result of DeepIM's holistic approach, encompassing the joint learning of seed set representations and end-to-end diffusion estimation models. DeepIM can successfully navigate diverse underlying diffusion patterns and consistently generate competitive and reliable influence spread results.

### 5.5. Scalability analysis

We conducted an analysis of the runtime for seed set inference across different node sizes in comparison to other learning-based IM approaches. The results, as presented in Table 5, showcase that DeepIM exhibits nearly linear growth in runtime as the graph size increases. Additionally, it boasts a shorter inference time, clocking in at an average of 20% faster compared to the second-fastest method, IMINFECTOR. Moreover, our DeepIM$_s$-I, which incorporates a lightweight end-to-end diffusion model, significantly reduces the computational burden associated with estimating expected influence spread. On average, DeepIM$_s$-I achieves a remarkable 82.32% improvement in inference time compared to our DeepIM-I model. On the other hand, DeepIM-II and DeepIM-III showed on average 89.04% and 87.80% improvement in inference time compared to our DeepIM-I model. This highlights the efficiency and effectiveness of our proposed DeepIM frameworks (with four variants) in handling large-scale networks.

### 5.6. Timing analysis

In Section 4.2, we introduced three inference methods. A comparison of these methods is illustrated in Figs. 2 and 3. Instead of training from scratch, the supervised approach (DeepIM-II) used transfer learning and an updated variable ($z'$) to speed up learning. In contrast, the unsupervised method (DeepIM-III) relied solely on a seed node ($x$) and took longer to learn the data distribution. Across all inference cases, a consistent pattern emerges: DeepIM-I requires more time compared to the other two inference methods. The most efficient timing is achieved by the DeepIM-III method. On average, DeepIM-II yields a 98.63% improvement in inference time, while DeepIM-III

**Table 3**
Performance comparison under LT diffusion pattern. – indicates out-of-memory error.

| Methods | Cora-ML | | | | Network Science | | | | Power Grid | | | | Jazz | | | | Synthetic | | | | Digg | | | | Weibo | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | 1% | 5% | 10% | 20% | 1% | 5% | 10% | 20% | 1% | 5% | 10% | 20% | 1% | 5% | 10% | 20% | 1% | 5% | 10% | 20% | 1% | 5% | 10% | 20% | 1% | 5% | 10% | 20% |
| IMM | 1.7 | 34.8 | 52.2 | 66.4 | 2.5 | 11.9 | 18.1 | 33.6 | 4.6 | 19.9 | 31.7 | 56.9 | 1.4 | 5.7 | 13.4 | 24.5 | 1.1 | 5.2 | 13.1 | 66.9 | 2.4 | 10.8 | 37.4 | 55.6 | 1.6 | 6.7 | 19.3 | 45.2 |
| OPIM | 2.3 | 36.9 | 51.2 | 71.5 | 1.6 | 12.0 | 18.8 | 34.1 | 4.4 | 21.6 | 29.4 | 55.5 | 1.4 | 6.9 | 12.6 | 20.9 | 1.3 | 5.2 | 12.6 | 62.1 | 2.1 | 11.3 | 38.2 | 57.1 | 1.8 | 6.1 | 18.7 | 46.6 |
| SubSIM | 1.7 | 33.6 | 54.7 | 70.1 | 1.8 | 10.4 | 19.2 | 34.1 | 4.5 | 21.1 | 31.2 | 57.4 | 1.4 | 5.9 | 11.4 | 21.2 | 1.4 | 5.5 | 13.1 | 69.6 | 2.4 | 11.3 | 37.9 | 56.9 | 1.7 | 6.7 | 19.2 | 46.8 |
| IMINFECTOR | 2.1 | 33.9 | 51.3 | 70.6 | 2.1 | 11.8 | 18.7 | 34.5 | 4.2 | 21.3 | 31.6 | 56.2 | 1.4 | 6.2 | 13.5 | 22.8 | 1.3 | 5.2 | 12.9 | 67.4 | 2.2 | 11.1 | 38.9 | 58.7 | 1.8 | 6.4 | 18.6 | 47.5 |
| PIANO | 2.1 | 33.5 | 53.3 | 69.8 | 2.1 | 11.3 | 19.1 | 33,9 | 4.3 | 21.3 | 31.4 | 57.1 | 1.1 | 6.2 | 12.1 | 22.4 | 1.2 | 5.2 | 12.9 | 67.4 | – | – | – | - | – | – | – | – |
| ToubleGDD | 2.3 | 36.2 | 54.5 | 70.9 | 2.8 | 12.4 | 19.8 | 34.6 | 4.8 | 21.9 | 32.6 | 58.1 | 1.4 | 6.5 | 12.9 | 23.6 | 1.3 | 5.5 | 13.4 | 70.2 | – | – | – | - | – | – | – | – |
| DeepIM$_s$-I | 10.7 | 65.6 | 75.1 | 85.2 | 3.5 | 14.6 | 23.8 | 37.8 | 5.1 | 22.9 | 40.3 | 65.1 | 1.4 | 6.5 | 14.2 | 85.3 | **1.5** | 6.0 | 14.2 | 90.3 | 3.1 | 13.3 | 39.2 | 67.9 | 2.5 | 7.1 | 32.6 | 68.4 |
| DeepIM-I | **13.4** | **69.2** | **83.5** | **94.1** | **4.1** | 16.6 | 26.7 | 41.5 | 6.3 | 24.4 | 46.8 | 71.7 | **1.9** | **6.5** | **16.4** | **99.1** | **1.5** | 6.5 | 15.5 | **99.9** | 3.5 | 15.9 | 41.3 | **76.2** | **3.1** | **7.6** | 39.3 | 72.4 |
| DeepIM-II | 12.28 | 68.93 | 81.67 | 92.74 | 3.40 | **16.68** | **27.94** | **41.79** | 7.73 | **37.00** | **66.20** | **86.64** | 1.01 | 5.56 | 11.11 | 81.82 | **1.5** | 6.31 | 15.73 | **99.9** | **3.7** | **16.4** | **41.5** | 75.9 | **3.1** | 7.5 | **39.9** | **74.2** |
| DeepIM-III | 6.69 | 64.02 | 76.44 | 92.74 | 2.08 | 14.29 | 27.63 | 41.60 | **8.46** | 36.53 | **66.20** | **86.64** | 0.51 | 5.56 | 13.64 | 73.74 | **1.5** | **6.74** | **16.19** | **99.9** | 3.5 | 16.3 | 40.2 | 74.8 | **3.1** | 7.1 | 38.1 | 70.3 |

**Table 4**
Performance over comparison methods under SIS diffusion pattern. (Best is highlighted with bold.)

| Methods | Cora-ML | | | | Network Science | | | | Power Grid | | | | Jazz | | | | Synthetic | | | | Digg | | | | Weibo | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | 1% | 5% | 10% | 20% | 1% | 5% | 10% | 20% | 1% | 5% | 10% | 20% | 1% | 5% | 10% | 20% | 1% | 5% | 10% | 20% | 1% | 5% | 10% | 20% | 1% | 5% | 10% | 20% |
| Greedy | 1.6 | 8.3 | 14.8 | 26.1 | 1.1 | 5.4 | 11.6 | 20.8 | 1.1 | 5.0 | 10.2 | 21.3 | 17.4 | 33.7 | 49.6 | 64.2 | 2.5 | 12.1 | 19.5 | 35.5 | 1.9 | 8.6 | 15.6 | 31.2 | 1.5 | 7.2 | 13.9 | 28.7 |
| IMINFECTOR | 2.1 | 9.4 | 16.1 | 27.9 | 1.7 | 5.8 | 12.4 | 22.3 | 1.3 | 5.5 | 12.4 | 23.1 | 8.8 | 35.4 | 54.8 | 66.2 | 2.5 | 12.4 | 20.5 | 36.2 | 2.3 | 9.1 | 16.4 | 32.4 | 2.5 | 8.5 | 15.5 | 29.6 |
| IMM | 2.0 | 9.5 | 15.4 | 27.6 | 1.3 | 5.6 | 12.2 | 22.1 | 1.1 | 5.6 | 11.0 | 22.9 | 7.6 | 37.8 | 55.6 | 67.1 | 2.7 | 12.6 | 20.9 | 37.3 | 2.5 | 9.4 | 16.3 | 32.6 | 2.3 | 8.1 | 15.7 | 29.4 |
| OPIM | 2.3 | 9.3 | 16.2 | 27.2 | 1.4 | 5.9 | 13.0 | 22.1 | 1.2 | 5.9 | 11.2 | 22.4 | 5.7 | 44.7 | 58.6 | 68.3 | 2.8 | 12.5 | 20.2 | 36.1 | 2.3 | 9.3 | 16.5 | 32.1 | 2.3 | 8.5 | 15.3 | 29.7 |
| SubSIM | 2.3 | 9.2 | 16.9 | 28.8 | 1.5 | 5.6 | 12.2 | 23.3 | 1.2 | 5.6 | 11.4 | 21.9 | 2.9 | 30.1 | 53.8 | 67.0 | 2.5 | 12.6 | 20.2 | 36.5 | 2.5 | 9.5 | 16.1 | 32.3 | 2.3 | 8.3 | 15.6 | 29.4 |
| DeepIM-I | **7.1** | **16.1** | **21.9** | **30.8** | **2.7** | **8.7** | **15.1** | **25.1** | **1.9** | **7.6** | 13.3 | **23.8** | 27.1 | **57.1** | 68.1 | **74.1** | 3.2 | **14.4** | 24.5 | **39.1** | **5.6** | **11.4** | 18.8 | **36.3** | **6.5** | 13.1 | **17.1** | **32.3** |
| DeepIM-II | 6.4 | 15.50 | 20.70 | 29.4 | **2.7** | 8.6 | 14.9 | 24.8 | 1.7 | 7.56 | **13.37** | 23.48 | **31.71** | 56.62 | **68.64** | 73.18 | **3.44** | 13.9 | **25.33** | 39.85 | 5.30 | 10.2 | **19.10** | 35.92 | 5.70 | **13.4** | 16.50 | 32.13 |
| DeepIM-III | 6.1 | 15.3 | 20.4 | 29.7 | 2.5 | 8.3 | 14.1 | 24.3 | 1.48 | 7.11 | 13.33 | 23.68 | 26.41 | 54.44 | 65.45 | 73.08 | 3.13 | 14.12 | 23.14 | 38.85 | 4.9 | 10.9 | 18.30 | 35.7 | 6.2 | 12.8 | 15.9 | 31.44 |

**Table 5**
The average inference runtime (in seconds) with regard to the increase of node size (10,000, 20,000, 30,000, and 50,000). We also demonstrate the average training time by using 50,000 nodes graph. We select 10% of nodes as the seeds uniformly.

| | 10,000 | 20,000 | 30,000 | 50,000 | 50,000 (Training) |
|---|---|---|---|---|---|
| IMINFECTOR | 3.478 s | 7.842 s | 12.376 s | 16.492 s | 4753.67 s |
| PIANO | 5.948 s | 10.532 s | 16.575 s | 28.437 s | 14732.63 s |
| ToupleGDD | 10.476 s | 19.583 | 32.792 s | 58.985 s | – |
| DeepIM$_s$-I | 0.312 s | 0.616 s | 0.847 s | 1.275 s | 503.12 s |
| DeepIM-I | 1.402 s | 2.798 s | 5.124 s | 12.882 s | 1244.56 s |
| DeepIM-II | 0.093 s | 0.344 s | 0.674 s | 1.514 s | 1352.64 s |
| DeepIM-III | 0.121 s | 0.392 s | 0.723 s | 1.550 s | 2525.174 s |

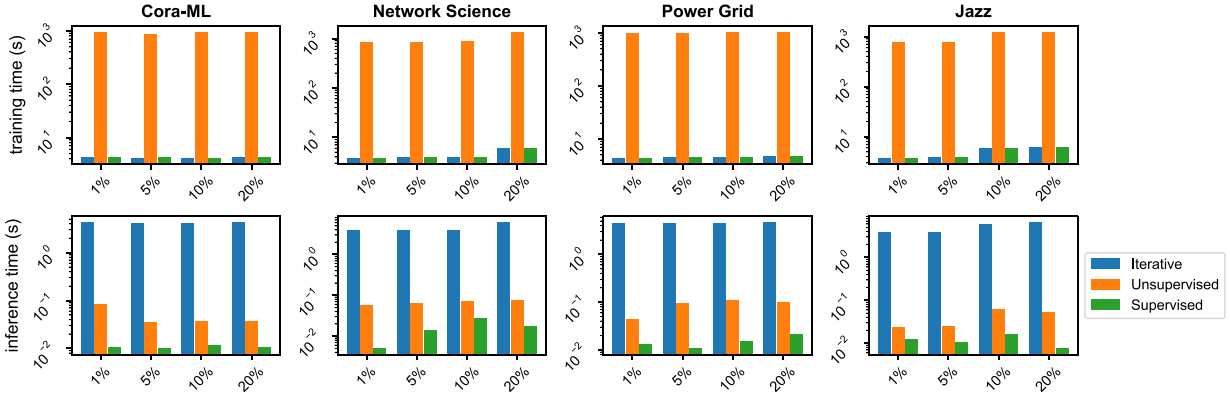



**Fig. 2.** Timing analysis for IC.

boasts an even greater improvement of 99.71% when compared to DeepIM-I. This underscores the enhanced efficiency offered by the DeepIM-II and DeepIM-III approaches in comparison to the DeepIM-I method.

### 5.7. IM with budget constraint.

To further evaluate the quality of seed sets generated by DeepIM and compare them with those produced by CELF under the IC and LT models, we introduce a budget constraint defined explicitly as the node degree. The outcomes of this comparison are depicted in Fig. 4. Across all networks of varying sizes, it is evident that DeepIM consistently outperforms CELF. This disparity is particularly pronounced under the LT model, as shown in Figs. 4(f) to 4(j). Moreover, the influence spread growth achieved by DeepIM exhibits fewer fluctuations in comparison to CELF across all datasets. This result further underscores the stability of DeepIM, attributing it to the model's capacity to identify latent distribution seed sets while accounting for the imposed budget constraint.

### 5.8. Case study: Inferred seed analysis

In a separate case study presented in Fig. 5, we examine the spread of the seed nodes inferred by the three inference methods of DeepIM. The visualization categorizes nodes into eight different types: Red

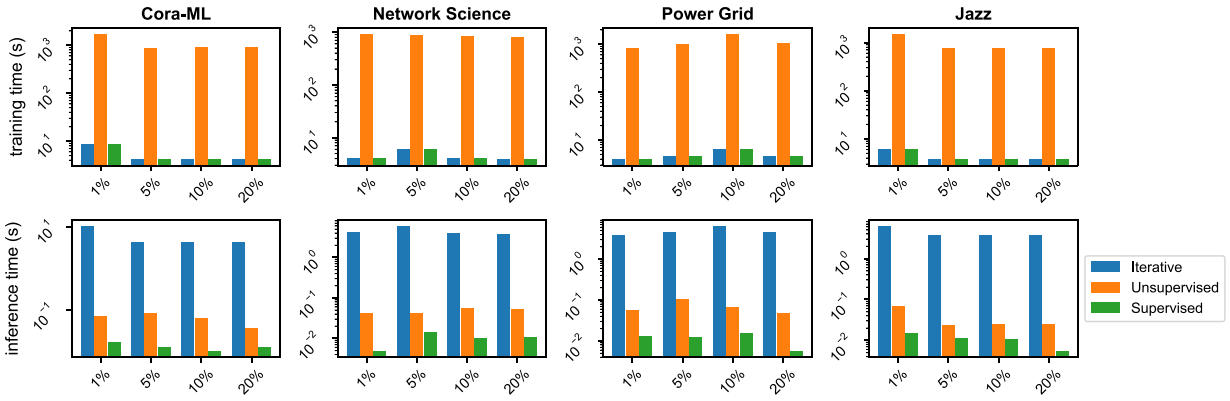



**Fig. 3.** Timing analysis for LT.

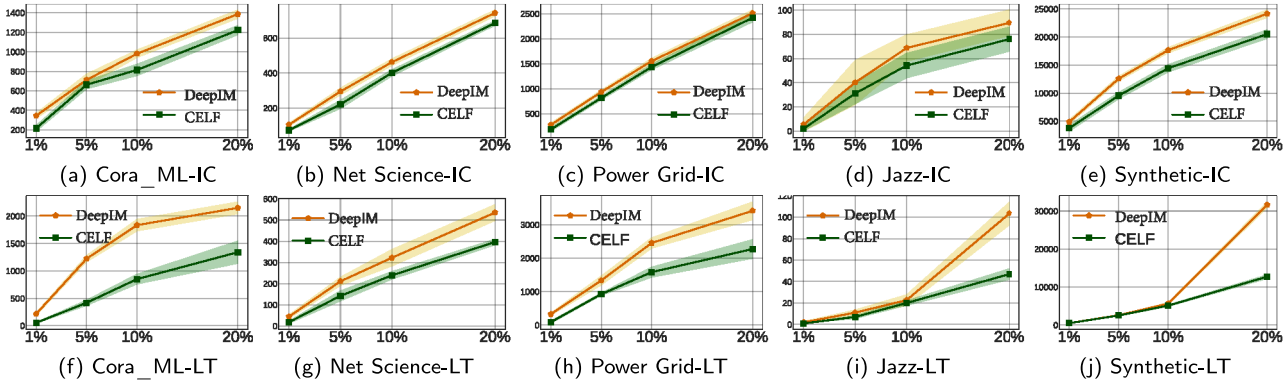



**Fig. 4.** The influence spread (total infected nodes) in the *y*-axis under the constraint of the budget with the node size growth (x-axis: 1%, 5%, 10%, and 20%). Fig. 4a–e and Fig. 4f–j are evaluated under the IC and LT model, respectively.

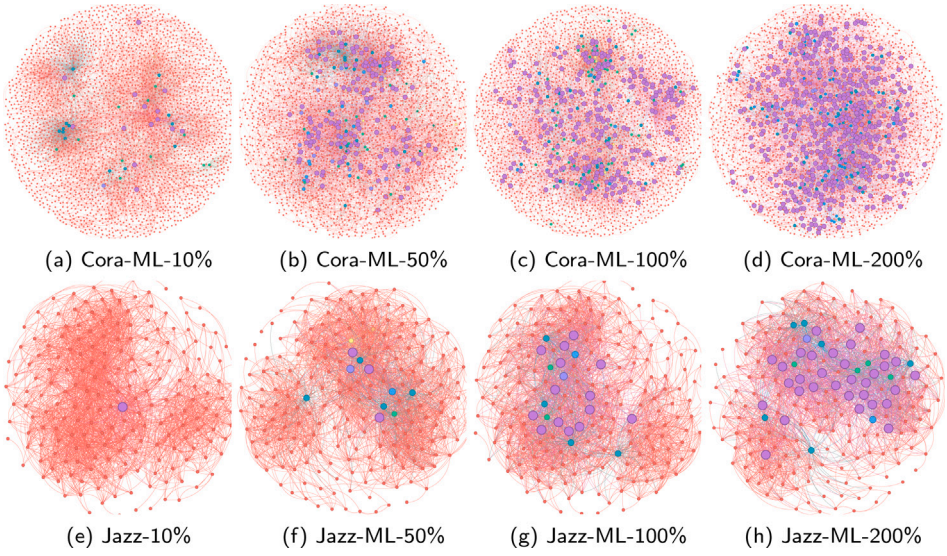



**Fig. 5.** Seed inference by three algorithms.

nodes represent non-seed nodes. Tangelo, flavescent, and eucalyptus-colored nodes indicate the seed nodes selected by DeepIM-I, DeepIM-II, or DeepIM-III exclusively. Lavender-colored nodes represent seed nodes selected by all three inference methods (I, II, and III). Turquoise, blue, and purple nodes denote the overlap selection between the two inference methods I+II, I+III, and II+III, respectively.

### *5.9. Case study: Graph diffusion visualization*

In a case study depicted in Fig. 6, we illustrate the distribution of selected seed nodes as well as the final infection status of all nodes. Blue nodes represent the initial seed nodes, red nodes signify infected nodes during the influence spread, and grey nodes denote uninfected nodes.

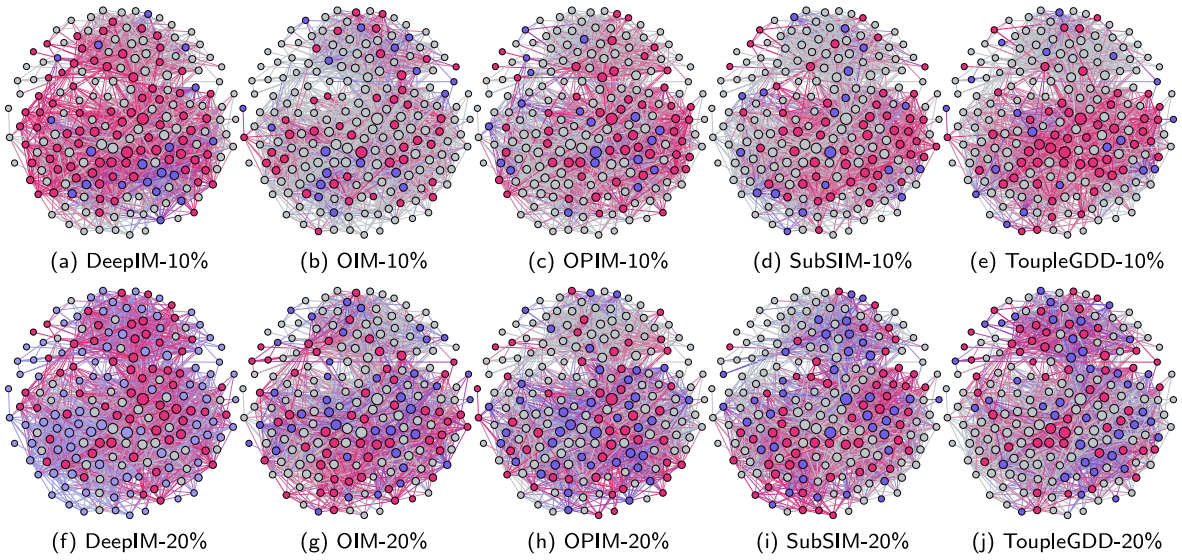



**Fig. 6.** The visualization of influence spread in Jazz dataset: The size of nodes is determined by the node degree, and the color on nodes determines the infection status: blue means the node is in seed set, red means the node is infected, and grey means the node is not infected.

We compare the influence spread outcomes for different initial seed set sizes: 10% and 20%. For clarity, we focus on visualizing the results for the Jazz dataset due to its smaller graph size. The visualization strongly supports DeepIM's superior performance in terms of achieving extensive influence spread. Particularly noteworthy is the observation in Figs. 6(a) and 6(f) that the final influence spread achieved by DeepIM with different initial seed set sizes is relatively small. This implies that DeepIM can achieve better outcomes with lower costs compared to other methods.

## 6. Conclusion

This work presents a pioneering framework that addresses the Influence Maximization (IM) problem with enhanced robustness and generalization compared to existing learning-based methods. In particular, our approach seeks to comprehensively capture the intricate characteristics of seed sets by directly characterizing their probability distribution, thereby enabling the search for more optimal seed sets within a continuous space. Moreover, we confront the challenge of modeling the underlying diffusion patterns by introducing two distinct learning-based diffusion models. These models effectively capture the diverse dynamics of diffusion with both efficiency and efficacy, offering a robust solution.

While our model did well at understanding the core qualities of seed sets, it could be even better if we considered the specific features of each seed node. These features could also help us handle situations where some information is missing, protect user privacy, and optimize influence maximization for groups, among other benefits.

Our framework introduces a novel objective function that accommodates various constraints for seed node set inference. We provided three different methods for seed node set inference. This versatility enables adaptation to diverse IM application scenarios. Extensive experiments and case studies conducted on both synthetic and real-world datasets showcase the superiority of DeepIM over existing state-of-the-art methods in terms of maximizing influence spread. Overall, DeepIM emerges as a promising approach that bridges the gap between traditional IM methods and the increasingly complex requirements of real-world influence maximization.

## CRediT authorship contribution statement

**Tanmoy Chowdhury:** Investigation, Methodology, Validation, Visualization, Writing – original draft, Writing – review & editing. **Chen Ling:** Conceptualization, Data curation, Formal analysis, Investigation, Methodology, Resources, Visualization, Writing – original draft, Writing – review & editing. **Junji Jiang:** Methodology. **Junxiang Wang:** Methodology. **My T. Thai:** Conceptualization. **Liang Zhao:** Conceptualization, Funding acquisition, Investigation, Methodology, Project administration, Supervision, Validation, Writing – review & editing.

## Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

## Data availability

Data will be made available on request.

## References

Ali, Khurshed, Wang, Chih-Yu, & Chen, Yi-Shin (2018). Boosting reinforcement learning in competitive influence maximization with transfer learning. In *2018 IEEE/WIC/ACM international conference on web intelligence* (pp. 395–400). IEEE.

Banerjee, Suman, Jenamani, Mamata, & Pratihar, Dilip Kumar (2020). A survey on influence maximization in a social network. *KAIS*, *62*(9), 3417–3455.

Barrett, Thomas, Clements, William, Foerster, Jakob, & Lvovsky, Alex (2020). Exploratory combinatorial optimization with reinforcement learning. In *Proceedings of the AAAI conference on artificial intelligence, vol. 34, no. 04* (pp. 3243–3250).

Cappart, Quentin, Chételat, Didier, Khalil, Elias B., Lodi, Andrea, Morris, Christopher, & Velickovic, Petar (2023). Combinatorial optimization and reasoning with graph neural networks. *Journal of Machine Learning Research*, *24*, 130–131.

Cappart, Quentin, Moisan, Thierry, Rousseau, Louis-Martin, Prémont-Schwarz, Isabeau, & Cire, Andre A. (2021). Combining reinforcement learning and constraint programming for combinatorial optimization. In *Proceedings of the AAAI conference on artificial intelligence, vol. 35, no. 5* (pp. 3677–3687).

Chen, Wei, Wang, Chi, & Wang, Yajun (2010). Scalable influence maximization for prevalent viral marketing in large-scale social networks. In *Proc. of the KDD* (pp. 1029–1038).

Chen, Tiantian, Yan, Siwen, Guo, Jianxiong, & Wu, Weili (2022). ToupleGDD: A fine-designed solution of influence maximization by deep reinforcement learning. arXiv preprint arXiv:2210.07500.

Chowdhury, Tanmoy, Ling, Chen, Zhang, Xuchao, Zhao, Xujiang, Bai, Guangji, Pei, Jian, et al. (2023). Knowledge-enhanced neural machine reasoning: A review. arXiv preprint arXiv:2302.02093.

Dolhansky, Brian W., & Bilmes, Jeff A. (2016). Deep submodular functions: Definitions and learning. *Advances in Neural Information Processing Systems*, *29*.

Du, Nan, Liang, Yingyu, Balcan, Maria, & Song, Le (2014). Influence function learning in information diffusion networks. In *ICML* (pp. 2016–2024).

Guo, Qintian, Wang, Sibo, Wei, Zhewei, & Chen, Ming (2020). Influence maximization revisited: Efficient reverse reachable set generation with bound tightened. In *Proc. of the SIGMOD* (pp. 2167–2181).

Guo, Xiaojie, Wang, Shiyu, & Zhao, Liang (2022). Graph neural networks: Graph transformation. In *Graph neural networks: Foundations, frontiers, and applications* (pp. 251–275). Springer.

Kamarthi, Harshavardhan, Vijayan, Priyesh, Wilder, Bryan, Ravindran, Balaraman, & Tambe, Milind (2019). Influence maximization in unknown social networks: Learning policies for effective graph sampling. arXiv preprint arXiv:1907.11625.

Kempe, David, Kleinberg, Jon, & Tardos, Éva (2003). Maximizing the spread of influence through a social network. In *Proc. of the KDD*.

Kermack, William Ogilvy, & McKendrick, Anderson G. (1927). A contribution to the mathematical theory of epidemics. *Proceedings of the Royal Society of London. Series A, Containing Papers of a Mathematical and Physical Character*, *115*(772), 700–721.

Khalil, Elias, Dai, Hanjun, Zhang, Yuyu, Dilkina, Bistra, & Song, Le (2017). Learning combinatorial optimization algorithms over graphs. *Advances in Neural Information Processing Systems*, *30*.

Kipf, Thomas N., & Welling, Max (2016). Semi-supervised classification with graph convolutional networks. arXiv preprint arXiv:1609.02907.

Kumar, Sanjay, Mallik, Abhishek, Khetarpal, Anavi, & Panda, B. S. (2022). Influence maximization in social networks using graph embedding and graph neural network. *Information Sciences*, *607*, 1617–1636.

Lei, Siyu, Maniu, Silviu, Mo, Luyi, Cheng, Reynolds, & Senellart, Pierre (2015). Online influence maximization. In *Proc. of the KDD*.

Leskovec, Jure, Krause, Andreas, Guestrin, Carlos, Faloutsos, Christos, Van-Briesen, Jeanne, & Glance, Natalie (2007). Cost-effective outbreak detection in networks. In *Proc. of the KDD*.

Li, Zhuwen, Chen, Qifeng, & Koltun, Vladlen (2018). Combinatorial optimization with graph convolutional networks and guided tree search. *Advances in Neural Information Processing Systems*, *31*.

Li, Yuchen, Fan, Ju, Wang, Yanhao, & Tan, Kian-Lee (2018). Influence maximization on social graphs: A survey. *TKDE*, *30*(10), 1852–1872.

Li, Xiang, Smith, J. David, Dinh, Thang N., & Thai, My T. (2019). Tiptop:(almost) exact solutions for influence maximization in billion-scale networks. *IEEE/ACM Transactions on Networking*, *27*(2), 649–661.

Li, Hui, Xu, Mengting, Bhowmick, Sourav S., Rayhan, Joty Shafiq, Sun, Changsheng, & Cui, Jiangtao (2022). PIANO: Influence maximization meets deep reinforcement learning. *IEEE Transactions on Computational Social Systems*.

Li, Hui, Xu, Mengting, Bhowmick, Sourav S., Sun, Changsheng, Jiang, Zhongyuan, & Cui, Jiangtao (2019). Disco: Influence maximization meets network embedding and deep learning. arXiv preprint arXiv:1906.07378.

Lin, Yishi, Chen, Wei, & Lui, John C. S. (2017). Boosting information spread: An algorithmic approach. In *2017 IEEE 33rd international conference on data engineering* (pp. 883–894).

Lin, Su-Chen, Lin, Shou-De, & Chen, Ming-Syan (2015). A learning-based framework to handle multi-round multi-party influence maximization on social networks. In *Proceedings of the 21th ACM SIGKDD international conference on knowledge discovery and data mining* (pp. 695–704).

Ling, Chen, Cao, Hengning, & Zhao, Liang (2023). Stgen: Deep continuous-time spatiotemporal graph generation. In *Machine learning and knowledge discovery in databases: European conference, ECML PKDD 2022, Grenoble, France, September 19–23, 2022, proceedings, part III* (pp. 340–356).

Ling, C., Chowdhury, T., Jiang, J., Wang, J., Zhang, X., Chen, H., et al. (2022). DeepGAR: Deep graph learning for analogical reasoning. In *2022 IEEE international conference on data mining* (pp. 1065–1070).

Ling, Chen, Jiang, Junji, Wang, Junxiang, Thai, My T., Xue, Renhao, Song, James, et al. (2023). Deep graph representation learning and optimization for influence maximization. In *International conference on machine learning* (pp. 21350–21361). PMLR.

Ling, Chen, Jiang, Junji, Wang, Junxiang, & Zhao, Liang (2022). Source localization of graph diffusion via variational autoencoders for graph inverse problems. In *Proc. of the KDD*.

Ling, Chen, Yang, Carl, & Zhao, Liang (2021). Deep generation of heterogeneous networks. In *2021 IEEE international conference on data mining* (pp. 379–388). IEEE.

Ling, Chen, Yang, Carl, & Zhao, Liang (2023). Motif-guided heterogeneous graph deep generation. *Knowledge and Information Systems*, 1–26.

Manchanda, Sahil, Mittal, Akash, Dhawan, Anuj, Medya, Sourav, Ranu, Sayan, & Singh, Ambuj (2019). Learning heuristics over large graphs via deep reinforcement learning. arXiv preprint arXiv:1903.03332.

McCallum, Andrew Kachites, Nigam, Kamal, Rennie, Jason, & Seymore, Kristie (2000). Automating the construction of internet portals with machine learning. *Information Retrieval*, *3*(2), 127–163.

Nguyen, Hung T., Thai, My T., & Dinh, Thang N. (2016). Stop-and-stare: Optimal sampling algorithms for viral marketing in billion-scale networks. In *Proc. of the SIGMOD*.

Nguyen, Hung T., Thai, My T., & Dinh, Thang N. (2017). A billion-scale approximation algorithm for maximizing benefit in viral marketing. *IEEE/ACM Transactions on Networking*, *25*(4), 2419–2429.

Nowak, Alex, Villar, Soledad, Bandeira, Afonso S., & Bruna, Joan (2018). Revised note on learning quadratic assignment with graph neural networks. In *2018 IEEE data science workshop* (pp. 1–5). IEEE.

Panagopoulos, George, Malliaros, Fragkiskos, & Vazirgiannis, Michalis (2020). Multi-task learning for influence estimation and maximization. *IEEE Transactions on Knowledge and Data Engineering*.

Rossi, Ryan A., & Ahmed, Nesreen K. (2015). The network data repository with interactive graph analytics and visualization. In *AAAI*.

Saito, Kazumi, Kimura, Masahiro, Ohara, Kouzou, & Motoda, Hiroshi (2012). Efficient discovery of influential nodes for SIS models in social networks. *Knowledge and Information Systems*, *30*(3), 613–635.

Sartori, Camilo Chacón, & Blum, Christian (2022). Boosting a genetic algorithm with graph neural networks for multi-hop influence maximization in social networks. In *2022 17th conference on computer science and intelligence systems* (pp. 363–371). IEEE.

Tang, Youze, Shi, Yanchen, & Xiao, Xiaokui (2015). Influence maximization in near-linear time: A martingale approach. In *Proc. of the SIGMOD*.

Tang, Jing, Tang, Xueyan, Xiao, Xiaokui, & Yuan, Junsong (2018). Online processing algorithms for influence maximization. In *Proc. of the SIGMOD* (pp. 991–1005).

Tang, Youze, Xiao, Xiaokui, & Shi, Yanchen (2014). Influence maximization: Near-optimal time complexity meets practical efficiency. In *Proc. of the SIGMOD* (pp. 75–86).

Tian, Shan, Mo, Songsong, Wang, Liwei, & Peng, Zhiyong (2020). Deep reinforcement learning-based approach to tackle topic-aware influence maximization. *Data Science and Engineering*, *5*(1), 1–11.

Vaswani, Sharan, Kveton, Branislav, Wen, Zheng, Ghavamzadeh, Mohammad, Lakshmanan, Laks V. S., & Schmidt, Mark (2017). Model-independent online learning for influence maximization. In *ICML*.

Veličković, Petar, Cucurull, Guillem, Casanova, Arantxa, Romero, Adriana, Lio, Pietro, & Bengio, Yoshua (2017). Graph attention networks. arXiv preprint arXiv:1710.10903.

Vesselinova, Natalia, Steinert, Rebecca, Perez-Ramirez, Daniel F., & Boman, Magnus (2020). Learning combinatorial optimization on graphs: A survey with applications to networking. *IEEE Access*, *8*, 120388–120416.

Wang, Yanhao, Fan, Qi, Li, Yuchen, & Tan, Kian-Lee (2017). Real-time influence maximization on dynamic social streams. arXiv preprint arXiv:1702.01586.

Wang, Shiyu, Guo, Xiaojie, & Zhao, Liang (2022). Deep generative model for periodic graphs. *Advances in Neural Information Processing Systems*, *35*.

Wang, Junxiang, Jiang, Junji, & Zhao, Liang (2022). An invertible graph diffusion neural network for source localization. In *Proceedings of the ACM web conference 2022* (pp. 1058–1069).

Wang, Junxiang, Li, Hongyi, Chai, Zheng, Wang, Yongchao, Cheng, Yue, & Zhao, Liang (2022). Toward quantized model parallelism for graph-augmented MLPs based on gradient-free ADMM framework. *IEEE Transactions on Neural Networks and Learning Systems*, 1–11.

Wu, Zonghan, Pan, Shirui, Chen, Fengwen, Long, Guodong, Zhang, Chengqi, & Philip, S. Yu (2020). A comprehensive survey on graph neural networks. *IEEE TNNLS*, *32*(1), 4–24.

Xia, Wenwen, Li, Yuchen, Wu, Jun, & Li, Shenghong (2021). Deepis: Susceptibility estimation on social networks. In *Proc. of the WSDM* (pp. 761–769).

Xu, Keyulu, Hu, Weihua, Leskovec, Jure, & Jegelka, Stefanie (2018). How powerful are graph neural networks? arXiv preprint arXiv:1810.00826.

Yang, Liang, Gu, Junhua, Wang, Chuan, Cao, Xiaochun, Zhai, Lu, Jin, Di, et al. (2020). Toward unsupervised graph neural network: Interactive clustering and embedding via optimal transport. In *2020 IEEE international conference on data mining* (pp. 1358–1363). IEEE.

Yang, Lan, Li, Zhiwu, & Giua, Alessandro (2020). Containment of rumor spread in complex social networks. *Information Sciences*, *506*, 113–130.

Ye, Mao, Liu, Xingjie, & Lee, Wang-Chien (2012). Exploring social influence for recommendation: a generative model approach. In *Proc. of the SIGIR* (pp. 671–680).

Zhang, Cai, Li, Weimin, Wei, Dingmei, Liu, Yanxia, & Li, Zheng (2022). Network dynamic GCN influence maximization algorithm with leader fake labeling mechanism. *IEEE Transactions on Computational Social Systems*.

Zhang, Zheng, & Zhao, Liang (2022). Unsupervised deep subgraph anomaly detection. In *2022 IEEE international conference on data mining* (pp. 753–762). IEEE.

Zhou, Jie, Cui, Ganqu, Hu, Shengding, Zhang, Zhengyan, Yang, Cheng, Liu, Zhiyuan, et al. (2020). Graph neural networks: A review of methods and applications. *AI Open*, *1*, 57–81.