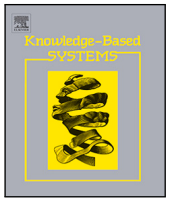
Knowledge-Based Systems 294 (2024) 111714

Contents lists available at ScienceDirect

Knowledge-Based Systems

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



Learning the long-tail distribution in latent space for Weighted Link Prediction via conditional Invertible Neural Networks

Yajing Wu [a], Chenyang Zhang [a], Yongqiang Tang [a,*], Xuebing Yang [a,*], Yanting Yin [b], Wensheng Zhang [a,b,c]

[a] State Key Laboratory of Multimodal Artificial Intelligence Systems, Institute of Automation, Chinese Academy of Sciences, Beijing, China
[b] Tianjin Key Laboratory of Network and Data Security Technology, College of Computer Science, Nankai University, Tianjin, China
[c] Guangzhou University, Guangzhou, Guangdong, China

ARTICLE INFO

Keywords:
Link prediction
Weighted dynamic network
Generative model
Invertible Neural Network

ABSTRACT

Link prediction is an important problem in dynamic systems, where the goal is to predict future unweighted or weighted link topologies using historical context. Compared with unweighted links, weighted links can preferably reveal the nature and strength of the interactions among entities. However, weighted links also bring greater challenges because they require subtle structural adjustments and numerical variations to be captured. Existing methods are primarily tailored for unweighted links and most generally suffer from low-quality performance when applied to Weighted Link Prediction (WLP) task. In this study, we propose a novel generative framework that adopts conditional Invertible Neural Networks (INNs) to achieve WLP. The proposed framework leverages the benefits of conditional INNs to exactly optimize the log-likelihood in the latent space conditioned on the historical context, which can be sensitive to minor replacements in real-world systems and derive accurate WLPs. Furthermore, to deal with the long-tail statistical phenomenon of edge weights observed in real life, a tail-adaptive distribution is learned in latent space to capture the tail properties and enhance the model's ability. To verify the effectiveness of the proposed method, we conduct extensive experiments on four datasets from different systems. The experimental results demonstrate that our model achieves impressive results compared to state-of-the-art competitors.

1. Introduction

Many ubiquitous real-world systems, such as social networks and communication, are complex in nature and evolve over time [1]. These dynamic systems typically provide a generalized abstraction of their behavior through a sequence of temporal snapshots. Each snapshot delineates the system entities and their interactions at a specific time step, as represented by a set of nodes and edges. Because of intrinsic dynamics, the links between nodes change dynamically over time, rendering them a compelling object of study. As a fundamental task in mining this type of data, link prediction has been widely utilized in various practical applications, *e.g.*, social media analysis and user behavior inference [2–4].

Considerable attention has been paid to the problem of link prediction. Link prediction can be roughly divided into two categories: unweighted and weighted. Unweighted link prediction aims to identify the presence or absence of links using observed information. Extensive research has been conducted in this field, yielding substantial advancements [5–7]. In contrast, in weighted systems, a weight is attached to each link, which serves to capture and retain additional information pertaining to the nature and strength of the interactions among the entities comprising a given system [8–11]. For instance, link weights can encompass significant information regarding the trustworthiness rating, flow, signal strength, or distance of network systems. Link weights play a pivotal role in characterizing the intricate nature of interactions in complex systems. However, conventional link prediction models, primarily designed for unweighted systems, inadequately capture the nuanced relationships embedded in weighted systems, thus prompting the need to advance predictive modeling for weighted systems. The motivation is rooted in recognizing that Weighted Link Prediction (WLP) algorithms explicitly account for edge weights, as they inherently encode crucial information regarding the strength, reliability, or distance within complex systems. In this scenario, the challenge of

* Corresponding authors.
E-mail addresses: yajing.wu@ia.ac.cn (Y. Wu), zhangchenyang2016@ia.ac.cn (C. Zhang), yongqiang.tang@ia.ac.cn (Y. Tang), yangxuebing2013@ia.ac.cn (X. Yang), yanting.yin@mail.nankai.edu.cn (Y. Yin), zhangwenshengia@hotmail.com (W. Zhang).

https://doi.org/10.1016/j.knosys.2024.111714
Received 3 June 2023; Received in revised form 19 March 2024; Accepted 26 March 2024
Available online 4 April 2024

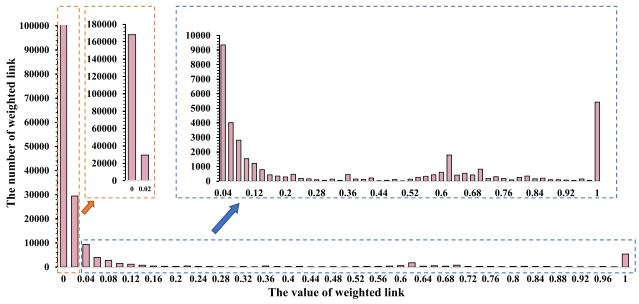


Fig. 1. Statistical number of weighted links of different values; a large number of values are concentrated at zero, and the remainder approximately follow a long-tailed distribution.

employing WLP necessitates not only identifying future links but also forecasting the accompanying link weights.

Numerous deep neural network-based methods have been developed for link prediction and achieved excellent performance [5,12,13]. Existing approaches such as DynGEM [5] and DynamicTriad [12] primarily focus on capturing dynamic system evolution variations, making predictions based on decoded temporal features. To control the length of the temporal patterns learned, DynRNN and DynAERNN [14] use Recurrent Neural Networks (RNNs) to learn the temporal transitions. Evolve-GCN [6] utilizes a combination of Graph Convolutional Networks (GCNs) and RNNs to extract features and learn sequences. To capture the evolution, DyCSC imposes dynamic constraints on the tendency in the dynamic system [15]. However, the majority of them are designed for unweighted link prediction task. Although several studies might own the potential of being applied to predict weighted links, they still have a significant limitation in that the fine-grained information contained in edge weights is gruffly overlooked.

To mitigate this issue, recent advances in deep generative models have led to significant progress in WLP. In general, generative models can explore the underlying distribution of realistic data and learn latent representations [16]. They play a crucial role in generating realistic data, enabling applications in image synthesis, data augmentation, *etc*. Currently, Generative Adversarial Networks (GANs) [17] and Variational Auto-Encoders (VAEs) [18] are widely adopted frameworks that have achieved remarkable success in high-quality prediction. For instance, GCN–GAN leverages the benefits of GCN and GAN to strengthen the representation learning for WLP [19]. Along this path, Att-GAN utilizes an attention-based GAN to solve the nonlinearity and spatial sparsity [20]. TVAE, a novel VAE-based model, utilizes latent space representations to describe the evolution of the topology [21]. In fact, weighted links in a complex system incorporate information with finer granularity, mainly reflected in the subtle structural adjustments and numerical variations. Nevertheless, present deep generative methods cannot well depict these tiny changes in the weighted links. Specifically, likelihood-based methods like VAEs, which optimize a lower bound on the log-likelihood, inevitably entail information loss [22]. Conventional GANs do not use encoders to infer latent variables [23]. These inherent constraints in the above generative models prompt us to explore a model which is able to truly acquire and encapsulate nuanced and granular information.

Fortunately, flow-based methods, as a new family of generative models, open up a compelling door to learn the underlying distribution of data [22]. In contrast to other generative models, flow-based methods are conceptually attractive owing to tractable log-likelihood and exact latent-variable inference. Notably, they enable the parallelizability of both training and synthesis via Invertible Neural Networks (INNs). Owing their invertibility, INNs can prevent information loss and preserve data details [24]. Notable examples, including NICE [25], MAF [26], RealNVP [27], Glow [23], and NSF [28], are widely used in density estimation, image generation, noise modeling, physics, and *etc* [29]. More recently, efforts such as GraphAF [30] and iGNN [31] have demonstrated their applicability for graph generation. By harnessing these benefits, INNs provide a promising direction to enhance model sensitivity for minor perturbations, thus allowing the desired precision in WLP.

Additionally, the WLP task itself has unique characteristics that require considerations. Among others, as shown in Fig. 1, the long-tail property of WLP is a crucial problem that deserves our attention. It is observed that a number of normalized link weights are concentrated at zero, and the remainder approximately follow a long-tail distribution. The primary reason behind the long-tail property in real-world applications is that links with larger weights have a weaker effect on the link formation than those with lower weights [8–10]. Hence, the model's ability to distinguish among lower weights and predict according to the long-tail distribution is crucial to further enhance the performance.

In this study, we propose a novel framework named CInvNet (**C**onditional **Inv**ertible neural **Net**work) for WLP to address the aforementioned challenges. In contrast to previous generative models, we introduce a conditional INN to address inference and generation in the forward and backward processes within a single model. With this explicitly invertible structure, our framework can preserve the exact likelihood in inference, which enforces model sensitivity to minor perturbations in a weighted network. Meanwhile, an informative weighted topology can be generated more finely by simply feeding a latent vector into the model. To further consider the unique long-tail property mentioned above, we propose constructing a Student's t-distribution rather than the commonly adopted Gaussian distribution in latent space to match the tail of weighted links in real-world data. In summary, we highlight our main contributions as follows:

- We propose a novel framework called CInvNet for the WLP task. CInvNet can prevent information loss and preserve data details, which can enhance model sensitivity to minor perturbations. Our framework provides a new paradigm for deep generative-based WLP. It can generate fine-grained and diverse weighted topology with high quality.
- For the long-tail property observed in realistic weighted systems, we propose to push Student's t-distribution, a tractable density with known tails, in latent space. With the need for these specific considerations, CInvNet can capture this unique pattern and generate weights with similar tail properties to match those in real-world data.
- We conduct a wide spectrum of experiments that demonstrate the consistently superior performance of CInvNet over state-of-the-art methods in WLP tasks.

The remainder of this paper is organized as follows. We cover the most related works in Section 2. Section 3 provides formal problem statements of this study and some necessary background context. In Section 4, the proposed CInvNet is detailed. Section 5 elaborates on experiments configurations for the evaluation of CInvNet, the experimental results and the corresponding analysis. Finally, Section 6 draws conclusions and indicates future research directions.

2. Related work

This section presents a concise review of related studies that are relevant to the main problem addressed in this study. Specifically, the related works are categorized into three groups: (1) studies focusing on the WLP task in dynamic systems, (2) widely used generative models and (3) flow-based models for generation.

2.1. WLP in dynamic systems

The task of link prediction in dynamic systems involves inferring the potential future topology based on its historical state [1]. This topic has

attracted significant attention in various fields, and considerable efforts have been devoted to solving the problems with different motivations.

Generally, capturing the variations during dynamic evolution of a system is challenging. Existing approaches such as DynGEM [5] and DynamicTriad [12] can overcome this challenge by simplifying assumptions. DynGEM [5] uses learned embedding from previous time step graphs to initialize the current time step embedding. Although DynGEM does not explicitly use regularization, such an initialization implicitly maintains the new embedding close to the previous embedding. DynamicTriad [12] relaxes the temporal smoothness assumption but only considers patterns spanning two time steps. These pioneering methods closely consider only the previous time steps and use the pattern over a short duration (length 2) to predict new links. To control the length of the temporal patterns learned, in recent studies [14], DynRNN and DynAERNN further use recurrent layers to learn the temporal transitions. In recent years, GCNs are gaining fast momentum for link prediction task in dynamic networks. For example, Evolve-GCN [6] utilities an RNN to regulate the GCN model (*i.e.*, network parameters) at each time step. Recently, DyCSC imposes dynamic constraints on the tendency in the dynamic system to a given number of clusters to capture the evolution [15]. Readers interested in delving deeper into this topic may refer to recent survey papers for a comprehensive review of the current state of research and an overview of the existing literature [1,32].

The aforementioned methods primarily focus on unweighted systems, whereas most real-world networks have weighted connections. Several pioneering studies [8–11] have extended these unweighted versions to weighted ones, in which the weights of links are explicitly taken into consideration. Nevertheless, these traditional methods cannot distinguish between small and zero weights, resulting in low-quality predictions. To obtain high-quality prediction results, several advanced methods such as GCN–GAN [19] and NetworkGAN [16] incorporate adversarial learning in conjunction with error minimization objectives. Inspired by the use of GANs, Tang et al. propose [20] to integrate GCN and temporal self-attention mechanism to generate an accurate future weighted structure. The performance evaluation clearly indicates that generative models offer a promising avenue for addressing the challenge of WLP.

2.2. Widely used generative models

More recent trends in the field of dynamic link prediction leverage generative models, which aim to directly address the difficulties in accurately reflecting the observed behaviors and connectivity patterns in complex systems. Architecturally, these models proposed a latent space model that favors a smooth evolution by inferring the latent positions of nodes or the topology of the system [3].

The current generative models for link prediction predominantly fall into three categories: Auto-Regressive models (ARs), VAEs, and GANs. ARs factorize the generation process into a sequence of steps, predicting the next output based on the previous output in the sequence. More explicitly, AR-based methods effectively utilize a mature framework in which RNNs are leveraged to gain insight into the distribution of numerous representations of systems that vary with the node ordering [33,34]. VAE-based models employ a simple variational distribution for latent representation vectors [21]. The third category of models implicitly learns the empirical distribution, particularly using GAN architecture [16,19]. GANs eschew the assumption of a simple variational distribution for latent representation vectors and instead adopt a more flexible approach to learning the distribution of data. In contrast to VAEs, GANs are best known for their ability to synthesize high-fidelity data. Nonetheless, they still have several drawbacks, including a lack of reliable latent-space encoders that can hinder interpretability and controllability [35]. Moreover, GANs are notoriously difficult to optimize and fine-tune, which can lead to unstable training [23].

Among the existing methods for WLP, there is a need to develop more advanced generative models that can accurately synthesize more intricate details of realistic data. Our model differs from these previous methods in the formulation of the entire framework. Unlike VAEs or GANs, we utilize the flow-based approach to achieve precise likelihood maximization. We believe precise optimization can yield models that are highly sensitive to minor replacements in real-world systems, thereby enhancing their accuracy and robustness.

2.3. Flow-based model for generation

In this study, link prediction is formalized as a conditional generation task using INNs. The INNs approach is a flexible and pleasingly simple way developed based on a flow-based model. Therefore, a brief overview of recent flow-based approaches is provided in this section.

Considerable research on flow-based models has focused on producing informative targets for a wide range of applications, including density estimation, image or graph generation, noise modeling, physics, and *etc* [22,29]. Additionally, INNs are popular flow-based models, that work through reversible transformations, enabling tractable log-determinant calculations. Notable examples include NICE [25], MAF [26], RealNVP [27], Glow [23], and NSF [28].

For the WLP problem, inspiration can be drawn from the application of flow-based models used in image or graph generation. In image generation applications, significant advancements have been made in utilizing flow-based models to model the image rescaling process. Recently, SRFlow [36] and HCFlow [37] leverage the invertibility of flow-based models to generate high-quality images with varying resolutions. Subsequently, owing to the considerable expressive power of flow-based models in practical scenarios, some research efforts have started to explore their application in graph generation. For example, GraphNVP [38] generates a graph and its atomic features in a one-shot and sequential manner, respectively. This is followed by other flow-based molecular graph generation approaches, such as GraphAF [30] and Moflow [39]. Moreover, iGNN tackles the inverse prediction problem on graphs by casting it as a conditional generative task [31]. Flow-based models have been shown to offer a promising solution for graph generation. To date, the application of flow-based models in graphs is mostly limited to the domain of molecular design in drug discovery.

Flow-based models for WLP have rarely been studied. As an extension, we propose flow-based invertible transformations for WLP. Building on this framework, we enforce validity constraints in the latent space to address the long-tail problem in WLP.

3. Preliminaries

Our framework formulates WLP as a deep generative model learning task. Therefore, in this section, we provide formal problem statements and review critical concepts to familiarize readers with the notation used throughout this paper.

3.1. Problem definition

WLP is of great significance in numerous applications. This paper focuses on predicting the weighted links within a system and only considers undirected ones whose scale will remain constant. We assume that the nodes are not attributed and focus solely on the structure. The weighted system is represented by the graph $\mathcal{G}_t = (\mathcal{V}, \mathcal{E}_t, \mathbf{A}_t)$ at time slice t, in which $\mathcal{V} = \{v_1, v_2, \ldots, v_N\}$ is the node set, $\mathcal{E}_t$ represents the set of links among the nodes, and $\mathbf{A}_t \in \mathbb{R}^{N\times N}$ is the adjacency matrix containing the structure of the system. To further clarify, if there is a link between nodes v_i and v_j, the value of $(\mathbf{A}_t)_{i,j} \in (0, 1]$ represents the weight of the link; otherwise, $(\mathbf{A}_t)_{i,j} = 0$. A value of $(\mathbf{A}_t)_{i,j}$ close to 1 indicates a strong relationship between the nodes.

Unlike previous prediction models that learn the mapping relationship directly from past snapshots to the next moment, we view WLP as a structure generation task. The goal is to learn the distribution

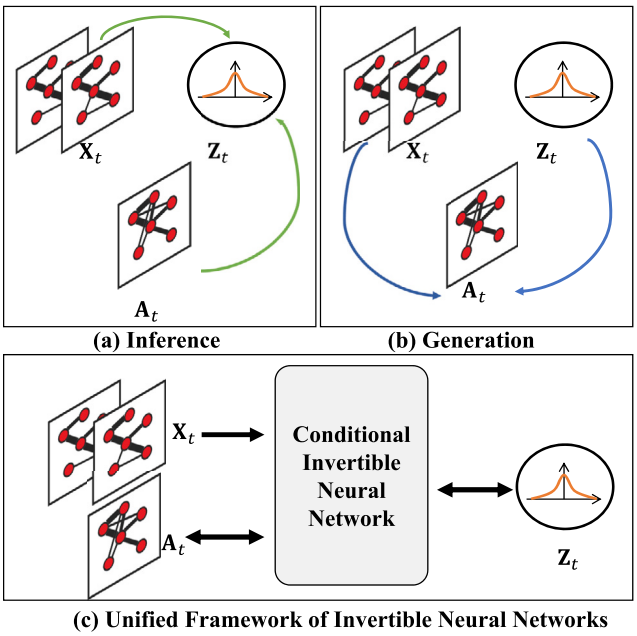


Fig. 2. Problem definition in the generative model. $\mathbf{X}_t$, $\mathbf{A}_t$ and $\mathbf{Z}_t$ denote $\mathcal{G}$ in previous snapshots, predictive weighted link of $\mathcal{G}$ and the latent variable, respectively.

of future states and generate a diverse and realistic future structure. Specifically, a simple generative model comprises two parts: inference and generation, as illustrated in Fig. 2(a) and (b).

Since prediction is based on past snapshots, conditional generation is further considered as the framework. The transformation between the predicted weighted links in $\mathbf{A}_t$ and the latent variable $\mathbf{Z}_t$ is conditional on the previous snapshots $\mathbf{X}_t = \mathbf{A}_{(<t)}$. For the sake of brevity, we omit the subscript t from the variables when the value of t is clarified and the subscript remains the same in a given representation. Thus, $\mathbf{X}$ is synonymous with $\mathbf{X}_t$, $\mathbf{A}$ is synonymous with $\mathbf{A}_t$, and $\mathbf{Z}$ is synonymous with $\mathbf{Z}_t$. Specifically, in the inference stage (a), with conditional generation, link embedding can be derived by inferring the posterior distribution of $\mathbf{Z}$:

$$\mathbf{Z} = F(\mathbf{A}; \mathbf{X}) \tag{1}$$

where $F(\cdot)$ is a learned function in the inference. In the generative procedure (b), both $\mathbf{X}$ and $\mathbf{Z}$ are considered as network inputs. The predicted weighted structure $\mathbf{A}$ can then be reconstructed from the latent encoding. Generative models are interested in learning the latent variable $\mathbf{Z}$ which can be characterized by (1) revealing the historical evolutionary trend of $\mathcal{G}$ in previous snapshots, (2) being highly indicative of weighted links in the future.

3.2. Invertible Neural Networks (INNs)

The core concept of INNs is that they are guaranteed to be invertible and can be used to parameterize F in Eq. (1). Samples from INNs can be drawn by first sampling $\mathbf{Z}$ with a simple tractable distribution $\mathbf{Z} \sim p(\mathbf{Z})$ (*e.g.*, multivariate Gaussian distribution), and then computing in conditional generation setting as follows:

$$\mathbf{A} = F^{-1}(\mathbf{Z}; \mathbf{X}) \tag{2}$$

INNs aim to learn a bijective mapping between the target space and the latent space as shown in Fig. 2(c), where $\mathbf{A}$ is transformed to $\mathbf{Z}$, Conversely, $\mathbf{Z}$ can be used to recover $\mathbf{A}$. Concretely, the function $F(\cdot)$ is constructed to be bijective, efficiently invertible, and with a tractable Jacobian determinant. We focus on functions F composed of a sequence of transformations: $F = f_1 \circ f_2 \circ \cdots \circ f_K$. Therefore, the relationship between $\mathbf{A}$ and $\mathbf{Z}$ can be expressed as follows:

$$\mathbf{A} \overset{f_1}{\longleftrightarrow} \mathbf{h}_1 \overset{f_2}{\longleftrightarrow} \mathbf{h}_2 \cdots \overset{f_K}{\longleftrightarrow} \mathbf{Z} \tag{3}$$

conditioned on input $\mathbf{X}$. This sequence of invertible transformations is also called a normalizing flow [40]. In this study, we consider four types of transformations and extend them to WLP in the next section.

According to the change in the variable formula and the chain rule, the logarithmic probability $\log(\mathbf{A})$ can be exactly calculated as follows:

$$\log\left(p\left(\mathbf{A}|\mathbf{X}\right)\right) = \log\left(p\left(F\left(\mathbf{A};\mathbf{X}\right)\right)\right) + \sum_{k=1}^{K} \log\left|\det \frac{\partial f_k}{\partial \mathbf{h}^{k-1}}\left(\mathbf{h}^{k-1}; g(\mathbf{X})\right)\right| \tag{4}$$

where $\log\left|\det \frac{\partial f_k}{\partial \mathbf{h}^{k-1}}\left(\mathbf{h}^{k-1}; g(\mathbf{X})\right)\right|$ is the logarithm of the absolute value of the determinant of the Jacobian of f_k at $\mathbf{h}^{k-1}$. $g(\mathbf{X})$ encodes the conditional input using a deep network that extracts a rich representation suitable for conditioning for following normalizing flow layers. INNs can thus be optimized by minimizing the Negative Log-Likelihood (NLL) loss.

4. The proposed model framework

In this section, we present a novel framework for WLP called CInvNet. The overall architecture is shown in Fig. 3. We propose a unified framework that uses INNs for inference and generation. Algorithmically, our objective is to address both the inference and generation processes through a single model that operates in the forward and backward directions. This framework aims to learn a latent space conditioned on the previous state of the system, to predict the weighted link that occurs at a later stage. Hence, we first explain the learning of the latent space through inference. Then, we describe the long-tailed distribution in the latent space. Finally, we introduce how to generate the predicted weighted links in our proposed unified framework.

4.1. Learning the latent space in the inference

Many generative methods introduce latent variables to explain the observed data. Let $\mathbf{Z} \in \mathbb{R}^{N \times N}$ be latent variables that are also random variables with a known and tractable probability density function. In the training phase, F maps a pairwise input $(\mathbf{A}, \mathbf{X})$ to a latent variable $\mathbf{Z}$. Eq. (4) allows the network be trained to force the latent variables to obey this simple distribution.

As illustrated in Fig. 3, the proposed CInvNet consists of (1) a feature extractor and (2) INN blocks. Thus, following most conditional INNs, we first encode the previous structure using a shared feature extractor $g(\mathbf{X})$, which extracts a rich representation suitable for conditioning at all layers.

4.1.1. Feature extractor

Since $g(\mathbf{X})$ does not need to be invertible, it is allowed to be any differentiable architecture. Based on the above, we have two considerations to take into account. First, it is preferable for the encoder $g(\mathbf{X})$ to be compatible with subsequent INNs as described previously. In addition, the previous structure $\mathbf{X}$ serves as the input to the encoder, which is consistent with the image data format. Building upon the existing mature made in conditional INNs [36,37], we draw inspiration to employ the Residual-in-Residual Dense Blocks (RRDB) [41] as our encoder as depicted in Fig. 3(1). RRDB contains residual structures at different levels and employs dense skip connections without any batch normalization layers. Specifically, the dense block which combines Convolution (Conv) and Rectified Linear Unit (ReLU) is used in the main path as shown in Fig. 4. Concretely, the input channels are formed by stacking historical snapshots, and we utilize three individual three-block RRDB networks for feature extraction. The capacity of CInvNet is thus increased, benefiting from conditioned historical context.

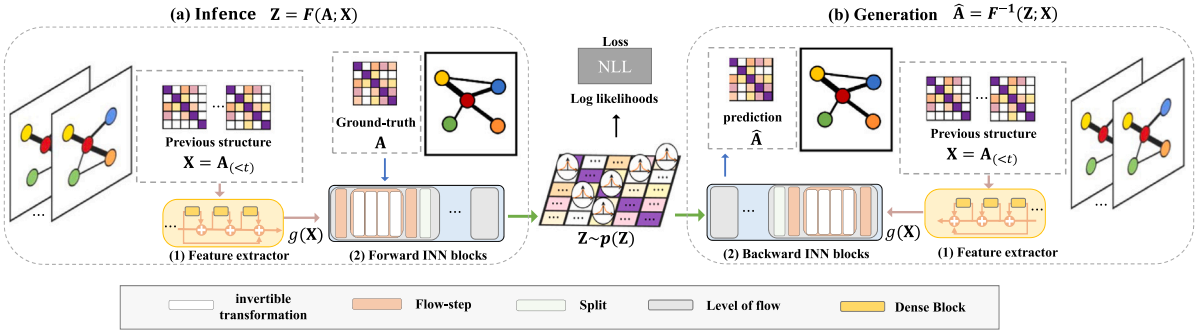


Fig. 3. Overview of our proposed framework called CInvNet. CInvNet consists of (1) a feature extractor and (2) INN blocks. The networks and parameters within CInvNet are the same for both (a) inference and (b) generation. (a) In the inference stage, the input pair $[\mathbf{A}, \mathbf{X}]$ is first delivered into the feature extractor. Next, the forward INN blocks project $\mathbf{A}$ to the latent space under the conditioned feature extracted from $\mathbf{X}$. The outcome component $\mathbf{Z}$ is assumed to obey a simple tractable distribution by a loss function. Our NLL loss replaces the often-used MAE loss. Conversely, (b) in the generative procedure: first, noise $\mathbf{Z}$ is sampled from the latent space and $g(\mathbf{X})$ is encoded by the feature extractor. Next, both are taken into the reversed backward INN blocks. Finally, the revealed prediction $\hat{\mathbf{A}}$ is recovered by drawing $\mathbf{Z}$ from the target distribution.

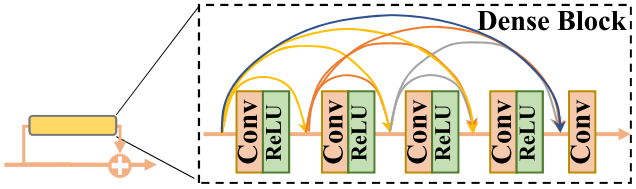


Fig. 4. Residual-in-Residual Dense Block (RRDB): the architecture combines residual network and dense connections.

4.1.2. INNs

The feature extractor can capture a richer representation of the temporal contexts, followed by the main INNs architecture. the main INNs architecture, as shown in Fig. 5, is organized into L levels, with each level containing K flow steps. Owing to the important properties of INNs that allow for the composability of invertible and differentiable transformations, four distinct transformations are executed sequentially at each flow step. As shown on the right of Fig. 5(a), Actnorm is applied first, followed by the Permutation. Next, we apply the Affine Injector followed by the Conditional Affine Coupling. Similar to most INN architectures [23,36], CInvNet does not maintain fixed dimensions of the channel (edge) throughout the architecture for the sake of computational cost. After K flow steps, the split operation is applied to evenly split off 50% of the channel dimensions ($N \times N \times C$ to $N \times N \times \frac{C}{2}$) before proceeding to the next level. Here, C is the channel dimension. In the following, each of the four transformations will be detailed.

Actnorm. We use an actnorm layer [23] to normalize the dimensions in each channel (edge) over a batch by an affine transformation, which is similar to batch normalization. The actnorm layer performs through a learned scaling $\mathbf{s}$ and bias $\mathbf{b}$: $\mathbf{h}_{i,j}^{k+1} = \mathbf{s} \odot \mathbf{h}_{i,j}^{k} + \mathbf{b}$, where $\odot$ represents element-wise multiplication. $\mathbf{s}$ and $\mathbf{b}$ are initialized with zero mean and unit variance and regraded as regular trainable parameters of the network. It can thus be inferred that the log-Jacobin determinant of each channel dimension is formulated as:

$$\log \left| \det \frac{\partial \left(\mathbf{s} \odot \mathbf{h}_{i,j}^{k} + \mathbf{b} \right)}{\partial \mathbf{h}_{i,j}^{k}} \right| = \sum \log |\mathbf{s}| \tag{5}$$

Permutation. The importance of the permutation operation in INNs lies in its ability to shuffle the input dimensions deterministically. Through the shuffling process, information is mixed more fully, ultimately resulting in a lower loss. In this work, Invertible 1×1 Convolution [23] serves as the surrogate of the permutation. Invertible 1×1 Convolution provides a learned permutation matrix that acts on each channel independently: $\mathbf{h}_{i,j}^{k+1} = W \mathbf{h}_{i,j}^{k}$. The log-Jacobin determinant is computed as follows:

$$\log \left| \det \frac{\partial \left(W \mathbf{h}_{i,j}^{k} \right)}{\partial \mathbf{h}_{i,j}^{k}} \right| = \log |\det (W)| \tag{6}$$

Affine Injector. The affine injector layer [36] facilitates the transfer of additional information from the conditioning temporal context to the main branch of the INNs. The conditioning temporal context $\mathbf{u} = g(\mathbf{X})$ if obtained, and the affine injector layer is achieved as $\mathbf{h}^{k+1} = \exp\left(f_k^{\mathrm{s}}(\mathbf{u})\right) \cdot \mathbf{h}^{k} + f_k^{\mathrm{b}}(\mathbf{u})$, where f_k^{s} and f_k^{b} can be any network. In our implementation, both are applied as two convolutional layers incorporating ReLU. The log-determinant of this transformation can be conveniently computed using $\sum_{i,j,k} f_k^{\mathrm{s}}(\mathbf{u})_{i,j,k}$, which is consistent with [36].

Conditional Affine Coupling. To achieve more conditional information in the main branch of INNs, we add the conditional affine coupling layer which is a conditional extension of the affine coupling layer [23]. In contrast to the affine injector layer, the conditional affine coupling layer splits the channel dimension: $\mathbf{h}^{k} = (\mathbf{h}_A^{k}, \mathbf{h}_B^{k})$. For one partition, $\mathbf{h}_A^{k+1} = \exp\left(f_k^{\mathrm{s}}(\mathbf{h}_B^{k}; \mathbf{u})\right) \cdot \mathbf{h}_A^{k} + f_k^{\mathrm{b}}(\mathbf{h}_B^{k}; \mathbf{u})$ is implemented. For the other partition, $\mathbf{h}_B^{k+1} = \mathbf{h}_B^{k}$ is simply implemented. Its log-determinant is similar to that of the affine injector layer.

4.1.3. Long-tail distribution in the latent space

In contrast to most studies that consider a simple Gaussian distribution as the underlying distribution in the latent space, we posit that a long-tail distribution is more suitable for modeling tail events. Specifically, we propose to model the underlying distribution using a standard Student's t-distribution, denoted as $\mathcal{T}(0, 1, \nu)$, where ν represents the degrees of freedom. Here, $\nu \in (1, +\infty)$ is a learnable parameter for each edge. Long-tail distribution-based methods provide a means of producing tail anisotropy for modeling real-world weighted link datasets. As shown in Fig. 3, our framework learns the long-tail distribution of sequence embedding instead of the isotropic Gaussian distribution used in other flow-based methods. Moreover, as the degrees of freedom parameter, ν, increases, the probability distribution, $\mathcal{T}(0, 1, \nu)$, exhibits a decrease in its tail heaviness, it approaches a standard Gaussian distribution as $\nu \to +\infty$. This simple probability distribution of our INNs remains tractable, allowing the transformations of INNs and degrees of freedom, ν, to be learned by maximizing the likelihood of the target long-tail distribution in the latent space.

Note that this work reinforces the view that high-quality generative models can be trained using the maximum likelihood loss alone. Therefore, by utilizing the Student's t-distribution, the NLL objective in Eq. (4) can be expressed as follows:

$$\begin{aligned} \mathcal{L} = & -\log\left(p_{\mathcal{T}(0,1,\nu)}(\mathbf{Z})\right) \\ & - \sum_{k=1}^{K} \log \left| \det \frac{\partial f_k}{\partial \mathbf{h}^{k-1}} \left(\mathbf{h}^{k-1}; g(\mathbf{X}) \right) \right| \end{aligned} \tag{7}$$

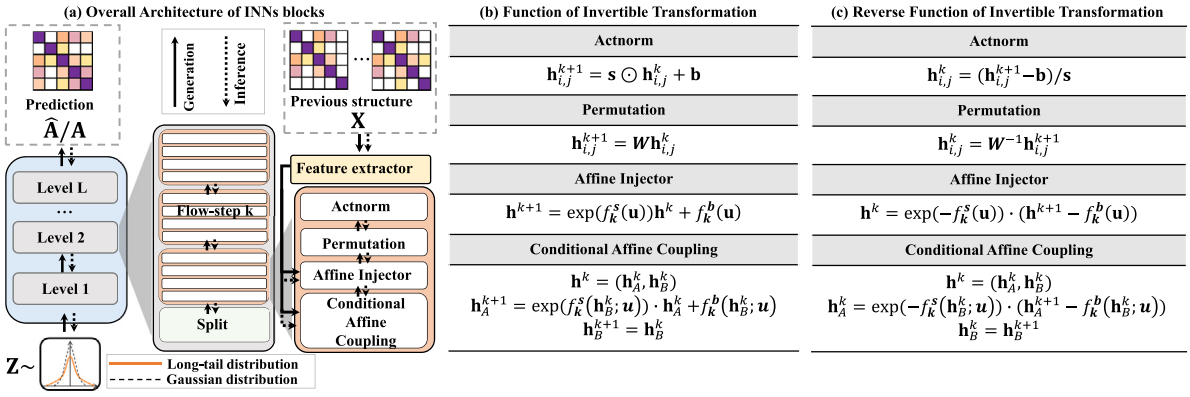


Fig. 5. (a) INNs blocks: the overall architecture consists of L levels. Each level contains K number of flow steps. At each flow step, four distinct transformations are executed in sequence; (b) The function of four transformations and (c) their reverses.

Here, the sum ranges over all log-Jacobin determinants of the flow steps in the INNs. During the inference stage, we only need to calculate the likelihood loss. This work studies the tail properties of the target density by requiring to push a tractable density in the latent space with known tails to the desired target density.

4.2. Predicting the weighted link in the generation

Once the latent variable is forced to obey the long-tail distribution during the inference stage, our framework is amenable to predicting the weighted link in the generative procedure. During the generation visualized in Figs. 3 and 5, the model converts noise (sampled from the multivariate long-tail distribution in latent space) into the desired target distributions ($\mathbf{Z} \rightarrow \mathbf{A}$).

The inverse of the inference process is the generation process. As shown in Fig. 3(b), predicted weighted links can be constructed progressively along the direction of the arrow. CInvNet generates a high-quality weighted link from a random variable sampled from the latent space using the conditioned historical context from the feature extractor. The feature extractor is consistent with that in the inference process. Hence, we focus on the reversed INNs in the generative procedure. The reverses of the four main components of our proposed framework can be trivially obtained as shown in Fig. 5(c). The reverse functions of the corresponding transformation are computationally efficient. Note that the sequence of transformations entered is also reversed.

With the well-trained framework, future weighted links are gradually constructed through a series of invertible operations. As our framework can be seen to model distribution in predictive space, it possesses great flexibility by capturing a variety of possible predictions. This enables the exploration of diverse predictions by incorporating supplementary guidance and stochastic sampling techniques.

5. Experimental results and analysis

In this section, we detail the datasets used to evaluate the performance of the dynamic WLP task, the evaluation metrics and the experimental results.

5.1. Datasets

In this work, we perform our proposed CInvNet on four different datasets to demonstrate its effectiveness. For each dataset, we pre-process the dynamic system into a series of successive structure snapshots. The details of the datasets are listed in Table 1.

Table 1
A summary of datasets in our experiments. The number of nodes, snapshot and system type are provided for each dataset.

Name	#Node	#Snapshot	Type
UCSB	38	1000	Wireless Mesh Network
KAIST	92	500	Human Mobility Position
NumFabric	128	350	Simulation data center flow
BJ-Taxi	256	565	Vehicle Mobility Position

- **UCSB** [42]: This is a popular dataset for link quality in wireless mesh networks. The nodes of the dynamic systems are the hosts in the network. Moreover, the link weights represent communication quality or flow between the corresponding pair of hosts in a snapshot.
- **KAIST** [43]: The KAIST dataset represents a human mobility system located on the campus of KAIST University. Specifically, each node in the dataset corresponds to a user, and the weight of each link represents the distance between any two users.
- **NumFabric** [44]: This is a widely used flow dataset designed to simulate dynamic workloads in a data center. The flow size between two hosts (nodes) is normalized to the link weights within a certain period of time.
- **BJ-Taxi** [45]: BJ-Taxi is the position dataset of a vehicle mobility network in Beijing. We construct a dynamic system based on the distances between 256 taxis, using GPS trajectories recorded on 2008-02-03. Moreover, the distances are quantified into weighted links that represent the distances between corresponding pairs of taxis.

Given that the four aforementioned datasets have been widely adopted in state-of-the-art methods for WLP, we maintain uniformity in our methodology by adhering to the same pre-processing as that adopted by these methods [19,20] to ensure a fair comparison. In the experiments, the first 70% of the sequences are used for training, the last 20% are used for testing and the remaining 10% are used for validation.

5.2. Comparative methods

To demonstrate the effectiveness of the proposed method, two groups of state-of-the-art models for temporal link prediction are considered.

Deep NN-Based Models: The development of deep learning has produced numerous prediction approaches owing to its powerful performance. Therefore, various outstanding deep NN-based models are considered as competitors.

Table 2
Average WLP results of four data sets based on the MAE and RMSE.

Methods	UCSB		KAIST		NumFabric		BJ-Taxi	
	MAE	RMSE	MAE	RMSE	MAE	RMSE	MAE	RMSE
DynRNN [14]	0.0487	0.0322	0.0694	0.0329	0.0240	0.0031	0.1387	0.0541
DynAERNN [14]	0.0452	0.0301	0.0593	0.0287	0.0249	0.0031	0.1512	0.0594
DynGEM [5]	0.0471	0.0263	0.1378	0.0574	0.0076	*0.0016*	0.1366	0.0598
DySAT [7]	0.0541	0.0480	0.1422	0.1814	0.0081	0.0067	0.2177	0.2384
EnvolveGCN-O [6]	0.0466	0.0151	0.0822	*0.0176*	0.0063	*0.0016*	0.1798	0.0528
EnvolveGCN-H [6]	0.0451	0.0132	0.0781	0.0190	0.0055	*0.0016*	0.1831	0.0554
GCN–GAN [19]	*0.0342*	*0.0194*	0.1288	0.0535	0.0045	*0.0016*	0.1711	0.0576
SIVGRNN [2]	0.0464	0.0425	0.0753	0.0512	*0.0041*	0.0021	0.1255	0.0646
TVAE [21]	0.0348	0.0264	*0.0494*	0.0277	0.0052	*0.0016*	*0.1044*	*0.0459*
CInvNet (ours)	**0.0177**	**0.0102**	**0.0106**	**0.0022**	**0.0031**	**0.0015**	**0.0459**	**0.0190**

The best results are in boldface. Italic with underlined ones represent the second-best results.

- **DynRNN**: DynRNN is a variation of the deep learning model proposed in dyngraph2vec [14] that uses Long Short Term Memory (LSTM) to handle long-term dependency problems in system evolution. DynRNN reduces reconstruction loss and generates node embedding predictions for the next snapshot.
- **DynAERNN**: Dyngraph2vec [14] also proposes a variation called DynAERNN, which utilizes fully connected layers to encode the representation of nodes instead of directly passing their vectors.
- **DynGEM** [5]: DynGEM is an efficient algorithm that uses deep auto-encoders to generate latent embedding of a dynamic system incrementally.
- **Evolve-GCN-O**: Evolve-GCN-O is a version proposed in EvolveGCN [6] that employs RNN to facilitate learning the weights of GCN instead of directly applying recurrent layers to refine the embeddings. In Evolve-GCN-O, the recurrent architecture uses GCN parameters as inputs/outputs.
- **Evolve-GCN-H**: Evolve-GCN-H is another version proposed based on EvolveGCN [6]. In Evolve-GCN-H, the GCN parameters are treated as hidden states of a recurrent architecture that takes node embeddings as input.
- **DySAT** [7]: DySAT is a dynamic self-attention network that learns node representations to capture the dynamic structural evolution of a system. DySAT has a clear architecture, consisting of two main blocks: a structural attention block and a temporal attention block.

Widely Used Generative Models: Generative models provide a new paradigm for applying neural networks in WLP problems. In this study, we thoroughly investigate the application of generative methods, focusing specifically on the prominent models.

- **GCN–GAN** [19]: This model combines the strengths of the GCN [46], LSTM and GAN to strengthen the representation learning of the network data and generate a high-quality structure snapshot in the next time slice.
- **SIVGRNN** [2]: SIVGRNN is a hierarchical variational model that uses a graph RNN to capture changes both in node attributes and topology in dynamic systems. Besides, with flexible non-Gaussian latent representations, SIVGRNN can boost expressive power in dynamic graph analytic tasks.
- **TVAE** [21]: Based on the VAE framework, this temporal network embedding utilizes method latent space representations to describe the evolution of the network topology.

As the WLP task on dynamic systems aims to predict possible future states using historical topology, 10 historical snapshots are delivered to all the methods for a fair comparison. For the comparative methods, we use their official public codes and select the parameters based on the recommended parameter settings. For our model, a grid-search algorithm is applied to automatically select the hyper-parameters. Specifically, the number of epochs is set to 500 for UCSB and 100 for the others, and an early stop strategy is used to alleviate model overfitting. The learning rate in training process is set to $5e-4$. Our main INNs architecture is organized into two levels. Each level contains three flow steps for all the datasets. Our experiments are implemented by PyTorch 3.7.3. All of the models are executed on a server with four NVIDIA Titan X GPU cards.

5.3. Evaluation criteria

Following the common procedure in WLP, four evaluation metrics are used: the Mean Absolute Error (MAE), the Root Mean Square Error (RMSE), the Edge-wise KL-divergence (EKL) and the Mismatch Rate (MR).

Given a ground-truth $\mathbf{A}$ and prediction $\hat{\mathbf{A}}$ for the test set, the MAE and RMSE are defined as follows:

$$\mathrm{MAE} = \frac{\left|\mathbf{A} - \hat{\mathbf{A}}\right|}{N \times N} \tag{8}$$

$$\mathrm{MSE} = \frac{\| \mathbf{A} - \hat{\mathbf{A}} \|_F^2}{N \times N} \tag{9}$$

In fact, MAE and MSE may be sensitive to large weights and struggle to distinguish the difference in magnitude that is important for small weights. the EKL can alleviate this issue by taking the magnitude difference of link weights into account. Formally, EKL is defined as follows:

$$\mathrm{EKL} = \sum_{i,j=1}^{N} \mathrm{KL}\left(\mathbf{P}_{i,j}, \mathbf{Q}_{i,j}\right) \tag{10}$$

$$\mathbf{P}_{i,j} = \frac{(\mathbf{A})_{i,j}}{\sum_{i,j=1}^{N} (\mathbf{A})_{i,j}}, \ \mathbf{Q}_{i,j} = \frac{\left(\hat{\mathbf{A}}\right)_{i,j}}{\sum_{i,j=1}^{N} \left(\hat{\mathbf{A}}\right)_{i,j}} \tag{11}$$

where KL is the standard form of the KL-divergence. Besides, the sparsity issue in the weighted dynamic network is typically significant. Thus, the following two cases are considered: (1) $(\mathbf{A})_{i,j} = 0$, whereas $\left(\hat{\mathbf{A}}\right)_{i,j} > 0$; (2) $(\mathbf{A})_{i,j} > 0$, whereas $\left(\hat{\mathbf{A}}\right)_{i,j} = 0$. Then, the frequency at the above two situations occur in the predicted network is computed as MR. MR represents the proportion of such mismatched edges in predictions, as an additional evaluation metric.

5.4. Comparisons with competitive methods

We compare our proposed CInvNet with the competitive methods described in Section 5.2, ranging from deep NN-based models to widely used generative models on the WLP task. The MAE and RMSE results are listed in Table 2. EKL and MR results are summarized in Table 3.

The results show that the prevailing deep NN-based models cannot achieve satisfactory performance on all datasets. Compared to the current deep NN-based models, the generative models generally perform better. Perhaps, this is due to the fact that generative models can learn real-world models and meaningful features of the input [23]. This confirms our initial intuition that the insight of generative model leads us

Table 3
Average WLP results of four data sets based on the KL and MR.

Methods	UCSB		KAIST		NumFabric		BJ-Taxi	
	EKL	MR	EKL	MR	EKL	MR	EKL	MR
DynRNN [14]	0.0235	0.4182	0.0061	0.3701	0.0024	0.1429	*0.0126*	0.4878
DynAERNN [14]	**0.0132**	0.3833	0.0031	0.2562	0.0024	0.1349	0.0128	0.4628
DynGEM [5]	0.0189	0.4064	0.0124	0.4442	0.0048	0.1406	0.0365	0.4271
DySAT [7]	0.2215	0.3935	0.4798	0.4693	0.0127	0.1584	0.0476	0.4919
EnvolveGCN-O [6]	0.0279	0.4214	0.0242	0.4471	0.0159	0.1975	0.0549	0.6384
EnvolveGCN-H [6]	0.0280	0.4642	0.0106	0.5464	0.0168	0.1975	0.0448	0.6443
GCN–GAN [19]	0.0319	**0.2147**	0.0735	0.4309	*0.0018*	0.1033	0.1857	0.6743
SIVGRNN [2]	0.0218	0.2676	*0.0077*	*0.1458*	0.0021	*0.0279*	0.0727	*0.3227*
TVAE [21]	0.0527	0.4916	0.0127	0.4946	0.0064	0.1496	0.0429	0.4980
CInvNet (ours)	*0.0187*	*0.2257*	**0.0012**	**0.0141**	**0.0002**	**0.0265**	**0.0096**	**0.0911**

The best results are in boldface. Italic with underlined ones represent the second-best results.

to generation and prediction and we aim to improve upon state-of-the-art generative models. Among the generative models, TVAE performs better performance in terms of MAE and RMSE. As the likelihood-based model, TVAE is able to infer approximately the values of the latent variables that correspond to the data by optimizing the lower bound but can be comparatively challenging to optimize. SIVGRNN can outperform TVAE sightly in terms of EKL and MR. SIVGRNN is capable of inferring more flexible posteriors and better modeling of sparse dynamic systems, which makes better MR results achieved on most datasets compared with other models. It should be noted that although our proposed method does not achieve the best performance on the UCSB in Table 3, it still yields the second-best performance. In later experiments, we observed that by employing different distributions for the latent space within our conditional INNs framework, the performance of EKL and MR can be further improved compared to the current results. Furthermore, the proposed model outperforms all of the other methods on the remaining datasets.

Based on the above results, it can be observed that our proposals significantly improve the generalization performance of the competitive methods. More specifically, we find that our base proposal CInvNet outperforms the second-best comparative model by 48.25% MAE on UCSB (GCN–GAN *vs.* CInvNet), 78.54% MAE on KAIST (TVAE *vs.* CInvNet), 24.39% MAE on NumFabric (SIVGRNN *vs.* CInvNet) and 56.03% MAE on BJ-Taxi (TVAE *vs.* CInvNet). These results demonstrate that the link prediction data allows more powerful architectures to be trained more precisely and to achieve better performance. As expected, the flow-based generative models can learn better features to represent future links for prediction in the latent space.

5.5. Ablation experiments

To thoroughly evaluate the ideas proposed in this paper, we begin with an extensive ablation study on all datasets. Our goal is to elucidate the contributions of each component of our novel network architecture, including the effects of latent space learning and the long-tail distribution. Hence, we compare our overall model (CInvNet) with the following variants: (1) without (w/o) latent space learning, which eliminates the process of learning in the latent space; (2) w/o long-tail distribution, which only replaces the long-tail distribution with Gaussian distribution. Overall, the results of the ablation experiments are presented in Tables 4–7. The most important insights from the results of the ablation study are also summarized below. In the following section, we discuss each experiment in detail.

Effect of the proposed latent space learning method. Our ablation study results show that latent space learning plays an essential role in the network, as it tends to yield a useful latent space for downstream tasks. Without such latent space learning, performance can actually degrade (row 3). We find that the latent space learning with the base Gaussian distribution (w/o long-tail distribution) outperforms w/o latent space learning by 30.69%, 83.50%, 2.94%, 53.76% MAE (refer to rows 2 *vs.* 3 in Tables 4, 5, 6 and 7), respectively. This is analogous to the insight from generative models such as GANs [17] and VAEs [47]. On this basis, reversible generative models are able to infer exactly without approximation, leading to the exact log-likelihood of the data, instead of a lower bound.

Table 4
Ablation study on UCSB.

Methods	MAE	RMSE	EKL	MR
CInvNet	**0.0177**	**0.0102**	**0.0187**	0.2257
w/o long-tail distribution	0.0201	0.0103	0.0191	**0.2092**
w/o latent space learning	0.0290	0.0201	0.0421	0.2272

The best results are in boldface.

Table 5
Ablation study on KAIST.

Methods	MAE	RMSE	EKL	MR
CInvNet	**0.0106**	**0.0022**	**0.0012**	**0.0141**
w/o long-tail distribution	0.0119	0.0023	0.0026	0.0184
w/o latent space learning	0.0721	0.0487	0.0046	0.1424

The best results are in boldface.

Table 6
Ablation study on NumFabric.

Methods	MAE	RMSE	EKL	MR
CInvNet	**0.0031**	**0.0015**	**0.0002**	**0.0265**
w/o long-tail distribution	0.0033	0.0016	0.0006	0.0722
w/o latent space learning	0.0034	0.0017	0.0012	0.0969

The best results are in boldface.

Table 7
Ablation study on BJ-Taxi.

Methods	MAE	RMSE	EKL	MR
CInvNet	**0.0459**	0.0190	**0.0096**	**0.0911**
w/o long-tail distribution	0.0547	**0.0175**	0.0248	0.1777
w/o latent space learning	0.1183	0.0660	0.1700	0.2894

The best results are in boldface.

Effect of the long-tail distribution. The results in the tables show that latent space learning with the long-tail distribution accounts for 11.94%, 10.92%, 6.06% and 16.09% improvements in MAE, respectively (rows 1 *vs.* 2 in four tables), justified primarily by long-tail learning ability in the overall network. Similar performance gains are also observed for the other evaluation criteria. The four datasets exhibit superior fits after allowing for long tails, with a further improved fit using a long-tail distribution. We find that adding the long-tail distribution to the latent space learning improves the performance somewhat, but the extent of this improvement is limited compared with the effect of the proposed latent space learning method. In fact, to fully capture the properties of the data, long-tail behavior should not be ignored, necessitating a method such as CInvNet.

Table 8
Performance of the original model and models improved with the latent space learning method module on four datasets.

Dataset	UCSB				KAIST			
Methods/Metrics	MAE	MSE	EKL	MR	MAE	MSE	EKL	MR
EvolveGCN	0.0451	**0.0132**	0.0280	0.4642	0.0781	**0.0190**	0.0106	0.5464
EvolveGCN+LSL	**0.0295**	0.0195	**0.0265**	**0.2312**	**0.0599**	0.0229	**0.0408**	**0.1563**
GCN–GAN	0.0342	0.0194	0.0319	0.2147	0.1288	0.0535	0.0735	0.4309
GCN–GAN+LSL	**0.0287**	**0.0164**	**0.0188**	**0.2126**	**0.1245**	**0.0530**	**0.0513**	**0.3515**
DySAT	0.0541	0.0480	0.2215	0.3935	0.1422	0.1814	0.4798	0.469
DySAT+LSL	**0.0511**	**0.0474**	**0.0360**	**0.2931**	**0.0737**	**0.0441**	**0.0690**	**0.1074**
Encoder	0.0290	0.0201	0.0421	0.2272	0.0721	0.0487	0.0046	0.1424
CInvNet	**0.0201**	**0.0103**	**0.0191**	**0.2092**	**0.0119**	**0.0023**	**0.0026**	**0.0184**
Dataset	NumFabric				BJ-Taxi			
Methods/Metrics	MAE	MSE	EKL	MR	MAE	MSE	EKL	MR
EvolveGCN	0.0055	0.0016	0.0168	0.1975	0.1831	0.0554	0.0448	0.6443
EvolveGCN+LSL	**0.0052**	**0.0016**	**0.0001**	**0.0534**	**0.1165**	**0.0373**	**0.0374**	**0.3726**
GCN–GAN	0.0045	0.0016	0.0018	0.1033	0.1711	0.0576	0.1857	0.6743
GCN–GAN+LS	**0.0043**	**0.0015**	**0.0003**	**0.0535**	**0.1523**	**0.0563**	**0.0952**	**0.5794**
DySAT	**0.0081**	0.0067	0.0127	0.1584	0.2177	0.2384	0.0476	0.4919
DySAT+LS	0.0092	**0.0040**	**0.0003**	**0.0629**	**0.1696**	**0.1092**	**0.1194**	**0.2370**
Encoder	0.0034	0.0017	0.0012	0.0969	0.1183	0.0660	0.170	0.2894
CInvNet	**0.0033**	**0.0016**	**0.0006**	**0.0722**	**0.0547**	**0.0175**	**0.0248**	**0.1777**

The best results are highlighted in bold.

5.6. Analytic experiment

We further conduct several necessary experiments to pursue deeper insight into our model. To empirically demonstrate the key components of CInvNet from different perspectives, we propose three research questions to guide the following experiments:

- **RQ$_1$:** Can the proposed latent space learning method be effectively adapted to other basic encoding models?
- **RQ$_2$:** How do different types of distribution influence the performance of latent space learning?
- **RQ$_3$:** Does our proposed model achieve fast convergence in practice?

5.6.1. Latent space learning analysis (*RQ$_1$*)

As stated previously, the function $g(\mathbf{X})$ is designed to accommodate various architectures for conditioned historical contexts. Therefore, we explore different architectures for feature extraction and ensure their compatibility with subsequent INNs. To demonstrate that our proposed latent space learning method can still improve performance with other backbone feature extractors, we extensively conduct experiments on all datasets using various basic encoding models as our feature extractors which learn link representations from mature methods. First, we consider three link representations in EvolveGCN, GCN–GAN and DySAT as our basic encoding model, $g(\mathbf{X})$. These three basic feature extractors are selected since they have been widely used in evaluating the quality of dynamic node representations to predict the temporal evolution of system structures. We reimplement these three encoding models and then leverage their link representations as inputs for our proposed Latent Space Learning (LSL). Second, for comparison purposes, we also decouple the encoder from our overall network. Then the LSL is replaced with two feed-forward layers for prediction.

Table 8 summarizes the evaluation scores for the LSL analysis on the four datasets. Detailly, we find that methods with LSL are competitive with EvolveGCN and GCN–GAN on KAIST in terms of MSE. A similar observation is made for NumFabric in terms of MAE when compared with DySAT. Apart from that, experimental results in Table 8 clearly indicate that all methods with LSL achieve better scores than the original link representations in most cases, which is reasonable since the base models take advantage from our proposed LSL. Overall, the experimental results confirm that our LSL module has a certain generalization ability, which is applicable not only for our proposed encoder network but also for other basic models.

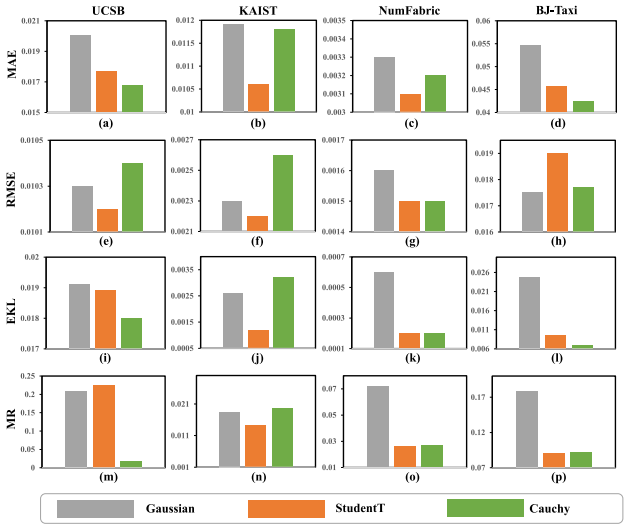


Fig. 6. The test MAE score of different types of distribution for our proposals on UCSB, KAIST, NumFabric and BJ-Taxi.

5.6.2. Comparison of different types of distributions (*RQ$_2$*)

Since the distribution in latent space plays a critical role in formulating flow-based generative models, we conducted an exploratory analysis to elucidate the influence of different types of distributions on the performance of learning in latent space.

First, it is common to set the base distribution as the standard Gaussian distribution (*i.e.*, $\mathbf{z} \sim \mathcal{N}(0, I)$). Second, we further explore an alternative long-tail distribution, the Cauchy distribution, in latent space learning. The Cauchy distribution has a heavier tail than the Student's t-distribution and has been proposed to deal with fat tails in variational inference in the recent literature [48]. For convenience, we use the naming convention StudentT to denote the Student's t-distribution. Fig. 6 shows the experimental results of the different types of distributions involved in the Gaussian, StudentT and Cauchy distributions on all datasets.

Using a long-tail distribution (StudentT or Cauchy distribution), we observe that one or the other of the two performs compares favorably with the standard Gaussian distribution in Fig. 6. From the results, there

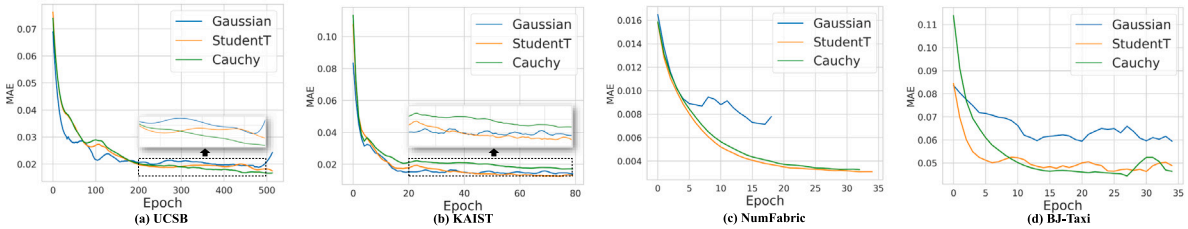


Fig. 7. Convergence analysis of different types of distribution on UCSB, KAIST, NumFabric and BJ-Taxi.

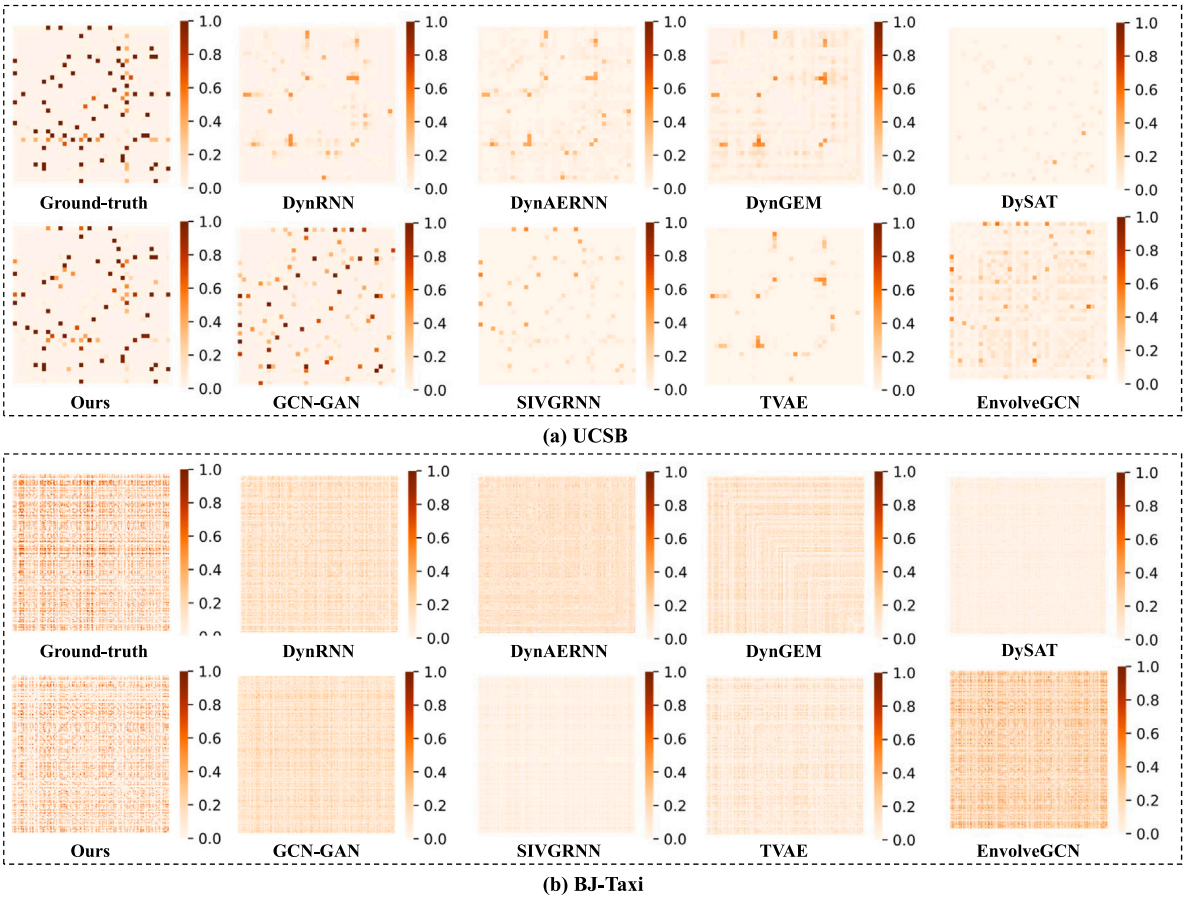


Fig. 8. Visualization of WLP generated by comparative methods and ours on UCSB (a) and BJ-Taxi (b).

is no clear winner between CInvNet with the Student's t-distribution or the Cauchy distribution, as each of these combinations performs better than the other on different datasets. This may be because different datasets exhibit different degrees of tailing effects. Note that when the tailed distribution is not suitable for the property of data, the model obviously harms prediction compared with the simple distribution. The Student's t-distribution distribution can address heavy-tail flows and the Cauchy distribution is able to accurately model fat-tail structure [48]. Hence, distribution should be selected appropriately in practice.

5.6.3. Convergence analysis (RQ_3)

To evaluate the convergence of our proposed model, we present the MAE values for four datasets during training, as shown in Fig. 7. The results reveal that utilizing a long-tail distribution in latent space learning is effective in reducing errors. However, the observed improvements are less significant for the first two datasets, as also highlighted in RQ_2. This might be attributed to the intrinsic properties of these datasets. For example, the non-Gaussian tail in the first two datasets is not obvious. Therefore, there is a small gap among the distributions. In contrast, NumFabric and BJ-Taxi may be more sensitive to the different degrees of tailed effects. We note that the Gaussian distribution has some defects such as a low convergence speed and falling into local minima. With a tailed distribution, the model converges faster and achieves lower MAE scores.

5.7. Visualization of WLP

For qualitative results, Fig. 8 further shows the advantageous properties of CInvNet (Ours) compared with the other methods. Here, we use heat maps to visualize the WLP generated by comparative methods and our method. Fig. 8 example cases on the smallest and the largest datasets. As expected from the results presented in Tables 2 and 3, our proposal performs particularly well on different cases.

In both cases, the following similar conclusions can be drawn. Intuitively, most comparative methods fail to predict the heavily weighted links on both datasets. Specifically, it can be observed that both DySAT and SIVGRNN suffer from underforecasting, since predicted values

Table 9
Average WLP results of DSBM1000 and DSBM2000.

Methods	DSBM1000				DSBM2000			
	MAE	RMSE	EKL	MR	MAE	RMSE	EKL	MR
DynRNN [14]	0.0171	0.0119	*0.0003*	0.2775	0.0132	*0.0067*	*0.0018*	0.3202
DynAERNN [14]	0.0193	0.0121	**0.0001**	0.2129	0.0188	0.0074	0.0027	0.2641
DynGEM [5]	0.0170	0.0120	**0.0001**	0.2183	0.0143	0.0076	**0.0012**	0.2750
EnvolveGCN-O [6]	0.0289	*0.0116*	0.0586	0.8493	0.0443	0.0120	0.0057	0.6124
EnvolveGCN-H [6]	0.0297	0.0119	0.0574	0.8467	0.0364	0.0129	0.0069	0.6124
SIVGRNN [2]	0.0169	0.0120	0.0010	*0.2111*	0.0134	0.0072	0.0030	0.3576
TVAE [21]	*0.0167*	*0.0116*	**0.0001**	0.4994	*0.0127*	0.0068	*0.0018*	*0.1997*
CInvNet (ours)	**0.0127**	**0.0089**	0.0007	**0.2085**	**0.0098**	**0.0055**	0.0023	**0.1587**

The best results are in boldface. Italic with underlined ones represent the second-best results.

are low. DynRNN, DynAERN, DynGEM and TVAE result in similar predictions, which are denser than ground-truth and blurry. The performances of GCN–GAN and EvolveGCN are relatively satisfactory. Our proposal could generate outstanding adjacency matrices which are similar to the ground-truths.

5.8. Experiments on larger dynamic networks

The effectiveness of CInvNet was demonstrated through the experiments in the previous sections. In this section, we also evaluate it on larger dynamic systems.

Two widely used larger datasets of the dynamic networks (*i.e.*, DSBM1000 and DSBM2000) [6,14,21] are considered to test our link prediction model. DSBM1000 and DSBM2000 follow the ones in [14, 21], which are generated from a dynamic stochastic block model with 1000 and 2000 nodes for simulating dynamic network structures and evolutions. For both datasets, four communities are set and a total of 100 snapshots are generated. In order to test the performance of the WLP model, 10–50 nodes of each community continuously increase their connection strength with other nodes.

We compare the performances of methods that can be directly used on larger datasets. The results on two larger datasets are listed in Table 9. Owing to limited computational resources, we made a trade-off between algorithm performance and runtime efficiency. To achieve this, community partition is employed to preprocess larger data. Once split into sub-communities, the proposed approach can perform efficient operations within each sub-community. From the results, we observe that SIVGRNN and TVAE generally obtain better results than the other comparative methods, whereas the proposed CInvNet achieves the best results in terms of almost all evaluation metrics. The current observations resemble the results from previous experiments. Meanwhile, the inferiority of CInvNet in EKL is demonstrated. For larger dynamic networks, this inferiority may arise because the community division operation of the model results in independent communities that lack weak continuity between communities. Perhaps, a running mode that operates directly on large-scale graphs may still be required to solve the problem of predicting community-level relationships.

6. Conclusions and future work

In conclusion, a new framework called CInvNet (**C**onditional **Inv**ertible neural **Net**work) is proposed for WLP in this study. This framework gains advantages from flow-based generative models that can generate fine-grained and diverse weighted topology data with high realism. Meanwhile, CInvNet explores pushing a tractable density with known tails instead of normal Gaussian base distribution in the latent space a pattern similar to that in a realistic weighted system. Our proposal was applied to four real-world WLP datasets. The results confirm that CInvNet substantially outperforms competing methods.

The use of conditional invertible neural networks for WLP is still in its initial stages, and many possible improvements should be explored. Future research will include the following: (1) a more sophisticated approach to deal with evolution in dynamic systems; (2) task-specific invertible transformations to further enhance the performance; (3) consideration of the sparseness of realistic data.

CRediT authorship contribution statement

Yajing Wu: Writing – review & editing, Writing – original draft, Project administration, Methodology, Investigation, Funding acquisition, Formal analysis, Data curation. **Chenyang Zhang:** Writing – original draft, Methodology, Investigation, Funding acquisition, Formal analysis, Data curation, Conceptualization. **Yongqiang Tang:** Writing – review & editing, Writing – original draft, Project administration, Methodology, Investigation, Funding acquisition, Formal analysis, Data curation, Conceptualization. **Xuebing Yang:** Writing – review & editing, Writing – original draft, Methodology, Investigation, Funding acquisition, Formal analysis, Data curation, Conceptualization. **Yanting Yin:** Writing – review & editing, Writing – original draft, Data curation, Conceptualization. **Wensheng Zhang:** Supervision, Project administration, Funding acquisition, Formal analysis, Conceptualization.

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

No data was used for the research described in the article.

Acknowledgments

The authors are thankful for the financial support by the National Key Research and Development Program of China (No. 2022YFF0903300), the National Natural Science Foundation of China (Grant Nos. 62206292, 62106266, 62206293 and U22B2048). This work was supported in part by Shandong Naoli Artificial Intelligence Tech. Co., Ltd, China.

References

[1] G. Xue, M. Zhong, J. Li, J. Chen, C. Zhai, R. Kong, Dynamic network embedding survey, Neurocomputing 472 (2022) 212–223.
[2] E. Hajiramezanali, A. Hasanzadeh, K. Narayanan, N. Duffield, M. Zhou, X. Qian, Variational graph recurrent neural networks, in: Proc. 33rd Int. Conf. Neural Inf. Process. Syst., 2019, pp. 10700–10710.
[3] L. Zhu, D. Guo, J. Yin, G.V. Steeg, A. Galstyan, Scalable temporal latent space inference for link prediction in dynamic social networks, IEEE Trans. Knowl. Data Eng. 28 (10) (2016) 2765–2777, http://dx.doi.org/10.1109/TKDE.2016.2591009.
[4] User behavior prediction model based on implicit links and multi-type rumor messages, Knowl.-Based Syst. 262 (2023) 110276.
[5] P. Goyal, N. Kamra, X. He, Y. Liu, DynGEM: Deep embedding method for dynamic graphs, 2018, arXiv preprint arXiv:1805.11273.

[6] A. Pareja, G. Domeniconi, J. Chen, T. Ma, T. Suzumura, H. Kanezashi, T. Kaler, T. Schardl, C. Leiserson, EvolveGCN: Evolving graph convolutional networks for dynamic graphs, in: Proc. 34th AAAI Conf. Artif. Intell., 2020, pp. 5363–5370.
[7] A. Sankar, Y. Wu, L. Gou, W. Zhang, H. Yang, DySAT: Deep neural representation learning on dynamic graphs via self-attention networks, in: Proc. 13th Int. Conf. Web Search Data Min., 2020, pp. 519–527.
[8] M.S. Granovetter, The strength of weak ties, Am. J. Sociol. 78 (6) (1973) 1360–1380.
[9] L. Lü, T. Zhou, Link prediction in weighted networks: The role of weak ties, Europhys. Lett. (EPL) 89 (1) (2010) 18001.
[10] B. Liu, S. Xu, T. Li, J. Xiao, X.-K. Xu, Quantifying the effects of topology and weight for link prediction in weighted complex networks, Entropy 20 (5) (2018) 363.
[11] L. Li, Y. Wen, S. Bai, P. Liu, Link prediction in weighted networks via motif predictor, Knowl.-Based Syst. 242 (2022) 108402.
[12] L. Zhou, Y. Yang, X. Ren, F. Wu, Y. Zhuang, Dynamic network embedding by modeling triadic closure process, in: Proc. 32nd AAAI Conf. Artif. Intell., vol. 32, (no. 1) 2018.
[13] Hierarchical attention link prediction neural network, Knowl.-Based Syst. 232 (2021) 107431.
[14] P. Goyal, S.R. Chhetri, A. Canedo, dyngraph2vec: Capturing network dynamics using dynamic graph representation learning, Knowl.-Based Syst. 187 (2020) 104816.
[15] S. Zhang, Z. Bu, DyCSC: Modeling the evolutionary process of dynamic networks based on cluster structure, 2022, arXiv preprint arXiv:2210.12690.
[16] M. Yang, J. Liu, L. Chen, Z. Zhao, X. Chen, Y. Shen, An advanced deep generative framework for temporal link prediction in dynamic networks, IEEE Trans. Cybern. 50 (12) (2020) 4946–4957.
[17] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, Y. Bengio, Generative adversarial nets, in: Proc. 28th Int. Conf. Neural Inf. Process. Syst., 2014, pp. 2672–2680.
[18] D.P. Kingma, M. Welling, et al., An introduction to variational autoencoders, Found. Trends Mach. Learn. 12 (4) (2019) 307–392.
[19] K. Lei, M. Qin, B. Bai, G. Zhang, M. Yang, GCN-GAN: A non-linear temporal link prediction model for weighted dynamic networks, in: Proc. IEEE Conf. on Comput. Commun. Soc., 2019, pp. 388–396.
[20] S. Tang, X. Xiao, Att-GAN: A deep learning model for dynamic network weighted link prediction, in: Proc. 3rd Int. Conf. Frontiers Technol. Inf. Comput., IEEE, 2021, pp. 15–20.
[21] P. Jiao, X. Guo, X. Jing, D. He, H. Wu, S. Pan, M. Gong, W. Wang, Temporal network embedding for link prediction via vae joint attention mechanism, IEEE Trans. Neural Netw. Learn. Syst. (2021) 1–14.
[22] I. Kobyzev, S.J. Prince, M.A. Brubaker, Normalizing flows: An introduction and review of current methods, IEEE Trans. Pattern Anal. Mach. Intell. 43 (11) (2021) 3964–3979.
[23] D.P. Kingma, P. Dhariwal, Glow: generative flow with invertible 1× 1 convolutions, in: Proc. 32nd Int. Conf. Neural Inf. Process. Syst., vol. 31, 2018, pp. 10236–10245.
[24] L. Liu, L. Tang, W. Zheng, Lossless image steganography based on invertible neural networks, Entropy 24 (12) (2022) 1762.
[25] L. Dinh, D. Krueger, Y. Bengio, Nice: Non-linear independent components estimation, in: Proc. 3rd Int. Conf. Learn. Represent. Workshop, 2015.
[26] G. Papamakarios, T. Pavlakou, I. Murray, Masked autoregressive flow for density estimation, in: Proc. 31st Int. Conf. Neural Inf. Process. Syst., 2017, pp. 2335–2344.
[27] L. Dinh, J. Sohl-Dickstein, S. Bengio, Density estimation using real NVP, in: Proc. 5th Int. Conf. Learn. Represent., 2017.
[28] C. Durkan, A. Bekasov, I. Murray, G. Papamakarios, Neural spline flows, in: Proc. 33rd Int. Conf. Neural Inf. Process. Syst., vol. 32, 2019.
[29] G. Papamakarios, E. Nalisnick, D.J. Rezende, S. Mohamed, B. Lakshminarayanan, Normalizing flows for probabilistic modeling and inference, J. Mach. Learn. Res. 22 (1) (2021) 2617–2680.
[30] C. Shi, M. Xu, Z. Zhu, W. Zhang, M. Zhang, J. Tang, GraphAF: A flow-based autoregressive model for molecular graph generation, in: Proc. 8th Int. Conf. Learn. Represent., 2020.
[31] C. Xu, X. Cheng, Y. Xie, Invertible neural networks for graph prediction, IEEE J. Sel. Areas Inf. Theory 3 (3) (2022) 454–467, http://dx.doi.org/10.1109/JSAIT.2022.3221864.
[32] M. Qin, D.-Y. Yeung, Temporal link prediction: A unified framework, taxonomy, and review, 2022, arXiv preprint arXiv:2210.08765.
[33] J. You, R. Ying, X. Ren, W. Hamilton, J. Leskovec, Graphrnn: Generating realistic graphs with deep auto-regressive models, in: Proc. 35th Int. Conf. Mach. Learn., PMLR, 2018, pp. 5708–5717.
[34] M. Popova, M. Shvets, J. Oliva, O. Isayev, MolecularRNN: Generating realistic molecular graphs with optimized properties, 2019, arXiv preprint arXiv:1905.13372.
[35] A. Grover, M. Dhar, S. Ermon, Flow-gan: Combining maximum likelihood and adversarial learning in generative models, in: Proc. 32nd AAAI Conf. Artif. Intell., vol. 32, (no. 1) 2018.
[36] A. Lugmayr, M. Danelljan, L. Van Gool, R. Timofte, SRFlow: Learning the super-resolution space with normalizing flow, in: Proc. 16th Eur. Conf. Comput. Vis., 2020, pp. 715–732.
[37] J. Liang, A. Lugmayr, K. Zhang, M. Danelljan, L. Van Gool, R. Timofte, Hierarchical conditional flow: A unified framework for image super-resolution and image rescaling, in: Proc. 18th Int. Conf. Comput. Vis., 2021, pp. 4076–4085.
[38] K. Madhawa, K. Ishiguro, K. Nakago, M. Abe, Graphnvp: An invertible flow model for generating molecular graphs, 2019, arXiv preprint arXiv:1905.11600.
[39] C. Zang, F. Wang, MoFlow: An invertible flow model for generating molecular graphs, in: Proc. 26th Int. Conf. Knowl. Discov. Data Min., 2020, pp. 617–626.
[40] D. Rezende, S. Mohamed, Variational inference with normalizing flows, in: Proc. 32nd Int. Conf. Mach. Learn., PMLR, 2015, pp. 1530–1538.
[41] X. Wang, K. Yu, S. Wu, J. Gu, Y. Liu, C. Dong, Y. Qiao, C. Change Loy, ESRGAN: Enhanced super-resolution generative adversarial networks, in: Proc. 14th Eur. Conf. Comput. Vis. Workshops, 2018.
[42] K. Ramachandran, I. Sheriff, E. Belding, K. Almeroth, Routing stability in static wireless mesh networks, in: Proc. 8th Int. Conf. on Passive and Active Network Measurement, Springer, 2007, pp. 73–82.
[43] K. Lee, S. Hong, S.J. Kim, I. Rhee, S. Chong, Slaw: A new mobility model for human walks, in: Proc. IEEE Conf. on Comput. Commun. Soc., INFOCOM, 2009, pp. 855–863.
[44] K. Nagaraj, D. Bharadia, H. Mao, S. Chinchali, M. Alizadeh, S. Katti, NUMFabric: Fast and flexible bandwidth allocation in datacenters, in: Proc. ACM SIGCOMM 2016 Conf., 2016, pp. 188–201.
[45] J. Yuan, Y. Zheng, X. Xie, G. Sun, Driving with knowledge from the physical world, in: Proc. 17th Int. Conf. Knowl. Discov. Data Min., 2011, pp. 316–324.
[46] T.N. Kipf, M. Welling, Semi-supervised classification with graph convolutional networks, in: Proc. 5th. Int. Conf. Learn. Represent., 2017, pp. 1–14.
[47] D.P. Kingma, M. Welling, Auto-encoding variational Bayes, in: Proc. 2nd Int. Conf. Learn. Represent., 2013.
[48] F. Liang, M. Mahoney, L. Hodgkinson, Fat–tailed variational inference with anisotropic tail adaptive flows, in: Proc. 39th Int. Conf. Mach. Learn., PMLR, 2022, pp. 13257–13270.