Generalized Degrees for Scalable Discrete Time Dynamic Graph Generation

Kjartan van Driel[1,†] **Leonardo Niccolò Ialongo**[1,2] **Pablo Astudillo-Estévez**[3,4,1]

Stefan Thurner[1,5,6,‡]

Abstract

The evolution of many real-world systems is best described by dynamic graphs, whose statistical properties reflect the constraints of the system. When forecasting their dynamics, the goal is to generate a time series of graphs respecting these underlying constraints. Existing scalable dynamic graph learning methods, however, are designed for local tasks such as link prediction or node classification, and their independent, local predictions are ill-suited for graph generation. This limitation is particularly relevant for discrete time dynamic graphs, where coarse time resolution induces dependencies among edges within each snapshot. We propose using a generalized notion of degrees to model such dependencies directly, thereby shifting the focus from individual links to node dynamics. This approach bypasses the need to learn a sparse graph representation, and yields an inductive representation that enables the generation of large-scale discrete-time dynamic graphs.

1 Introduction

The evolution of many real-world systems is best described by dynamic graphs, which capture interactions and their temporal evolution. When forecasting their dynamics, the goal is not just to predict individual edges but to generate sequences of graphs that reproduce key higher-order properties. These properties govern crucial phenomena, such as systemic risk in socio-economic systems [11], epidemics [22], disruptions to supply chains [8], as well as financial crashes [1, 3].

Past years have seen a large interest in learning on dynamic graphs [15, 19], and various scalable methods have been developed for dynamic graphs, either as temporal extensions of graph convolutional networks [17, 24, 26, 29] or as novel architectures [7, 30]. Most of these methods, however, are optimized for discrimination tasks rather than for graph generation, and, as Chanpuriya et al. [5] demonstrate, realistic yet novel graphs cannot be generated when training relies on discriminative, edge-independent methods.

In this work, we focus on large-scale discrete-time dynamic graphs, where time is coarsened into intervals and all interactions within an interval are collapsed into a single snapshot. This coarse-graining introduces dependencies between edges [21], and such graphs are particularly common when considering administrative data, such as trade data, regulatory filings, as well as tax records, which are essential for modeling economic systemic risk [2, 8].

The large-scale discrete-time setting poses unique challenges. Unlike in continuous-time, one cannot rely on the sparsity and exact ordering of edges to apply edge independent models, as demonstrated

[1] Complexity Science Hub, [2] Supply Chain Intelligence Institute Austria, [3] Universidad San Francisco de Quito & School of Economics, [4] Institute for New Economic Thinking, University of Oxford, [5] Section for the Science of Complex Systems, Center for Medical Data Science, Medical University of Vienna, [6] Santa Fe Institute, † van-driel@csh.ac.at, ‡ thurner@csh.ac.at

van Driel et al., Generalized Degrees for Scalable Discrete Time Dynamic Graph Generation (Extended Abstract). Presented at the Fourth Learning on Graphs Conference (LoG 2025), Hybrid Event, December 10–12, 2025.

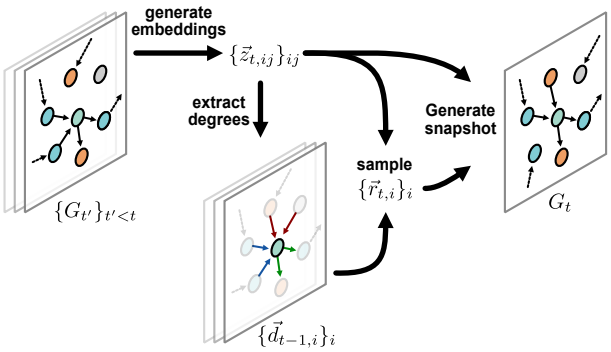

Figure 1: Conceptual illustration of the proposed framework, where the main innovation is the extraction and sampling of local structure using generalized degrees, which in turn provides a constraint for graph generation.

by Hosseini et al. [14]. At the same time, scalable methods that incorporate edge dependencies in static graphs do not directly extend to this regime [4, 28].

To address this gap, we propose a framework that models the evolution of local node topologies across snapshots, rather than individual links. This shift allows us to model important structural constraints explicitly while avoiding the need to learn sparse edge-level representations. Overall we develop an inductive representation that scales naturally to large graphs, enabling the generation of discrete-time dynamic graphs that preserve key structural properties.

2 Notation & Methods

We consider a time series of possibly attributed graphs $\{G_t\}_t$, with each graph defined as $G_t = (\mathcal{V}_t, \mathcal{E}_t, X_t, E_t)$, where $\mathcal{V}_t$ and $\mathcal{E}_t$ denote the node and edge set at time t respectively, $X_t \in \mathbb{R}^{|\mathcal{V}_t| \times d_x}$ is the node feature matrix, and $E_t \in \mathbb{R}^{|\mathcal{E}_t| \times d_e}$ is the edge feature matrix.

Generalized degrees extend the standard degree notion by distinguishing between different *classes* of edges, where each class represents a type of relationship. We formalize this idea as follows. First, we assume the existence of a map from the history of graphs $\{G_{t'}\}_{t' \leq t}$ to edge embeddings $\vec{z}_{t,ij} \in \mathbb{R}^{d_z}$. These embeddings may be observed directly from data, or defined latently as functions of the graph topology (e.g., via message passing networks), but crucially, we assume $\vec{z}_{t,ij}$ is defined for all node pairs. We then define a function $\pi : \mathbb{R}^{d_z} \to \{0, \ldots, k\}$ that partitions edges based on their embedding $\vec{z}_{t,ij}$.

Given a class $\gamma \in \{0, \ldots, k\}$, the γ-degree of node i at time t is defined as[1]:

$$d^{\gamma}_{t,i} = |\{\, j \mid \pi(\vec{z}_{t,ij}) = \gamma \,\}|. \tag{1}$$

To describe the local evolution of a node we consider how many of the edges that compose each γ-degree will be present in the next time step, as given by the γ-realization count:

$$r^{\gamma}_{t+1,i} = |\{\, j \mid (i,j) \in \mathcal{E}_{t+1},\ \pi(\vec{z}_{t,ij}) = \gamma \,\}|, \tag{2}$$

Temporal signature. In this work, we do not attempt to learn the partition π. Instead, our goal is to illustrate the utility of generalized degrees. To this end, we construct generalized degrees that capture the temporal evolution of a node's neighborhood by partitioning edges according to their direction and temporal signature. The temporal signature of an edge at time t is a bit-string of length τ recording its activity over the past τ timesteps; for example, when $\tau = 2$, the in, 01-degree $d^{\text{in},01}_{t,i}$ counts the number of incoming edges to node i with signature inactive at time $t-1$ and active at time t. We note that such partitions are inspired by, and generalize the work of Longa et al. [18].

[1]Undirected is chosen for ease of exhibition, for directed graphs, we assume two functions $\pi^{\text{in}}, \pi^{\text{out}}$, where then $d^{\gamma}_{t,i} = |\{\, j \mid \pi^{\text{out}}(\vec{z}_{t,ij}) = \gamma \,\} \cup \{\, j \mid \pi^{\text{in}}(\vec{z}_{t,ji}) = \gamma \,\}|$, so that one can distinguish the direction of the edge.

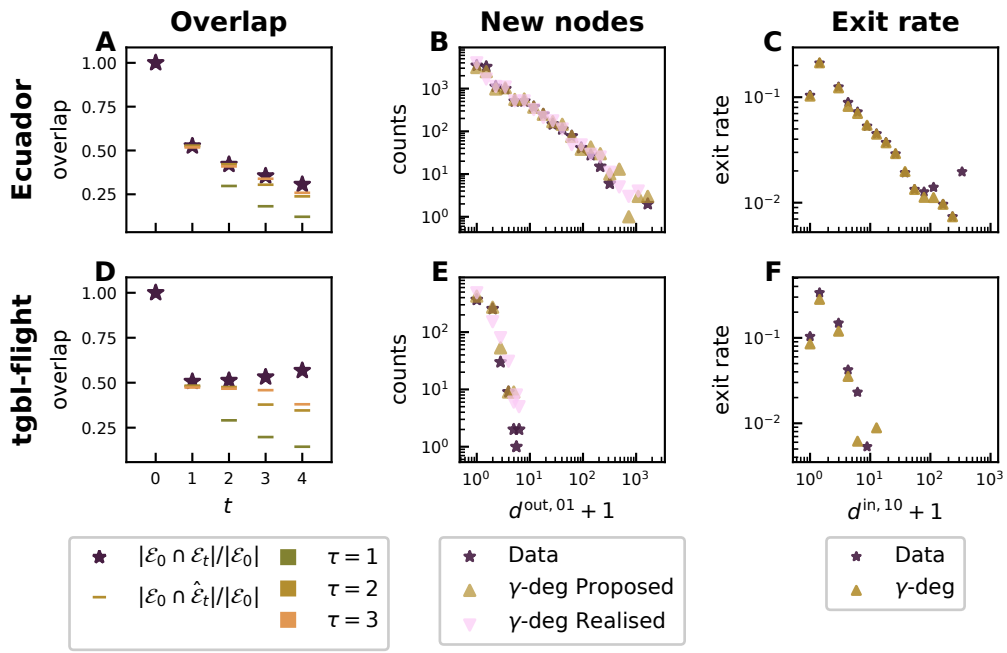

Figure 2: To highlight how our framework captures key properties of dynamic graphs, we generate an out of sample timeseries and compare key structural properties to the data. In particular we show: the edge overlap with respect to the data at time $t = 0$, the degree sequence of new nodes at $t = 1$, and the exit rate as a function of degree at time $t = 0$. Together, these properties capture whether the framework reproduces the right level of turnover among edges, new nodes, and exiting nodes. **A**,**D** show the overlap of generated edges $\hat{\mathcal{E}}_t$ to that of the data $\mathcal{E}_t$. **B**,**E** show the histogram of out-degree of entering nodes for the Data, the learned model (γ-deg Proposed), and the heuristic generation (γ-deg Realised). **C**,**F** show the exit rate for nodes given their in-10 degree.

Learning. For scalable learning, the key observation is that given the partition function π, one can model $p(\{\vec{r}_{t,i}\}_i \mid \{G_{t'}\}_{t'<t})$ without considering the likelihood over edges, thereby bypassing the need to learn a sparse matrix representation. We leave details of the learning procedure out of the main text (see Appendix B), but in short: we assume conditional independence between nodes given their previous degrees $\vec{d}_{t-1,i}$ so that the distribution factorizes as $p(\{\vec{r}_{t,i}\}_i \mid \{G_{t'}\}_{t'<t}) = \prod_i p(\vec{r}_{t,i} \mid \vec{d}_{t-1,i})$. To simplify the architecture, we learn the γ-realization ratio $\alpha^{\gamma}_{t,i} = \frac{r^{\gamma}_{t,i}}{d^{\gamma}_{t-1,i}}$, which is bounded within $[0, 1]$. We learn this distribution using a conditional RealNVP normalizing flow [9, 27], with a learned variational dequantizer [13] to handle discreteness, though the framework is not limited to this choice. For more details on our model choice see Appendix B.3.

Generation. The sampled realization counts describe the local evolution of individual nodes and can be viewed as constraints on the graph's topology. Crucially, satisfying such constraints introduces dependencies between edges, addressing the problem raised in the work of Chanpuriya et al. [5].

We propose a simple heuristic for directed discrete-time dynamic graphs that produces snapshots that approximately satisfy the constraints set by the realization counts. In brief, we assign each node a number of in-stubs matching its in-realization counts and then attach them, without replacement, to other nodes in proportion to their respective out-realization ratio, see Appendix D for details.

3 Experiments

We present two experiments to evaluate our framework. First, we demonstrate that it enables the generation of large dynamic graphs, learning the properties of entering and exiting nodes. Second,

we show that generalized degrees carry information beyond the temporal signatures of individual edges, using a link prediction task.

For the generation experiment, we use two large discrete-time dynamic graph datasets: **Ecuador** [2] and **tgbl-flight** [15]. **Ecuador** is a confidential, country-wide transaction graph provided by the Ecuadorian tax authorities (see Appendix A for details). For the link prediction experiment, we focus on the **tgbl-flight** benchmark.

3.1 Dynamic graph generation

To illustrate that our framework's ability to generate realistic discrete time dynamic graphs, we showcase its ability to reproduce three non-trivial structural properties: (i) overlap of edges with a reference set, (ii) the degree sequence of incoming nodes, and (iii) the exit rate as a function of degree. These properties together capture the extent to which the model reproduces turnover among edges, newly arriving nodes, and departing nodes, and highlight the main difference from discrimination tasks like link prediction. For both datasets, we train the model up to time t_L and generate a single out of sample time series extending four steps into the future. The results, shown in Figure 2, confirm that our learned distribution $p_\theta(\vec{r}_{t,i} \mid \vec{d}_{t-1,i})$ reproduces essential, and non-trivial properties of empirical dynamic graphs.

3.2 Link prediction

If the constraints implied by realization counts accurately capture the dynamics of the graph, then the corresponding realization ratios should provide good estimates of edge existence probabilities. In our setting, the generalized degrees arise from a partition of edges based on their temporal signature. We show that this captures information beyond the temporal signature of individual edges: predictions derived from our learned model, $p_\theta(\vec{r}_{t,i} \mid \vec{d}_{t-1,i})$, outperform those from f-Bank$_\tau$, a heuristic that simply records the frequency with which edges of temporal signature length τ at time t are active at the subsequent timestep $t+1$. We also compare against the EdgeBank heuristic [23], and state-of-the-art graph learning methods [10, 20, 24, 25, 31] on the large **tgbl-flight** benchmark [15]. The results are shown in Figure 3, with further details in Appendix C.

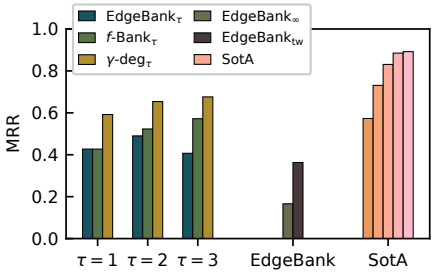

Figure 3: Link prediction performance on the **tgbl-flight** benchmark. Left: our method γ-deg as well as edge based benchmarks f-Bank and EdgeBank defined on the same time window τ. Right: Standard EdgeBank benchmark [23], as well as the current leaderboard for **tgbl-flight**: [10, 20, 24, 25, 31].

4 Conclusions & Outlook

We propose a framework that shifts the perspective from individual edges to that of node dynamics. Our representation of these dynamics — in terms of generalized degrees — allows us to explicitly model the evolution of local structural constraints. This reframes dynamic graph generation as a constrained link prediction problem. Our experiments suggest that the approach is promising: it scales to large graphs, reproduces non-trivial topological properties, and performs surprisingly well on link prediction.

The current partition is a design choice rather than a learned one. We chose a representation based on temporal signatures to demonstrate the value of our framework; however, this does not scale well for large τ. For future work, it will be crucial to cast the identification of the precise partition as a learning problem in itself. So far, our focus has been on learning the realization counts, yet enforcing these constraints during graph generation remains challenging. One promising direction is to combine our framework with scalable degree-guided diffusion methods [6, 28]. Finally, we acknowledge several simplifying assumptions in the current setup, most notably conditioning only on generalized degrees, which should be relaxed in future work.

Acknowledgments

We gratefully acknowledge financial support from the Austrian Federal Ministry for Economy, Energy, and Tourism (BMWET) and the Federal State of Upper Austria for supporting the Supply Chain Intelligence Institute Austria (ASCII), the Austrian Science Fund (FWF) for REMASS, doi: 10.55776/EFP5, the Austrian Central Bank, project Resilience of Economic Systems #18789, and the Austrian Research Promotion Agency FFG, project 924336, SYRIalert.

References

[1] Daron Acemoglu, Asuman Ozdaglar, and Alireza Tahbaz-Salehi. "Systemic Risk and Stability in Financial Networks". In: *American Economic Review* 105.2 (Feb. 2015), pp. 564–608. DOI: 10.1257/aer.20130456. 1

[2] Andrea Bacilieri et al. *Firm-Level Production Networks: What Do We (Really) Know?* July 8, 2025. DOI: 10.2139/ssrn.5344255. URL: https://papers.ssrn.com/abstract=5344255. 1, 4, 7

[3] Stefano Battiston et al. "DebtRank: Too Central to Fail? Financial Networks, the FED and Systemic Risk". In: *Scientific Reports* 2.1 (1 Aug. 2, 2012), p. 541. DOI: 10.1038/srep00541. URL: https://www.nature.com/articles/srep00541. 1

[4] Andreas Bergmeister et al. *Efficient and Scalable Graph Generation through Iterative Local Expansion*. May 14, 2024. DOI: 10.48550/arXiv.2312.11529. URL: http://arxiv.org/abs/2312.11529. 2

[5] Sudhanshu Chanpuriya et al. *On the Role of Edge Dependency in Graph Generative Models*. Dec. 6, 2023. DOI: 10.48550/arXiv.2312.03691. URL: http://arxiv.org/abs/2312.03691. 1, 3

[6] Xiaohui Chen et al. *Efficient and Degree-Guided Graph Generation via Discrete Diffusion Modeling*. May 31, 2023. DOI: 10.48550/arXiv.2305.04111. URL: http://arxiv.org/abs/2305.04111. 4

[7] Weilin Cong et al. *Do We Really Need Complicated Model Architectures For Temporal Networks?* Feb. 22, 2023. DOI: 10.48550/arXiv.2302.11636. URL: http://arxiv.org/abs/2302.11636. 1

[8] Christian Diem et al. "Quantifying Firm-Level Economic Systemic Risk from Nation-Wide Supply Networks". In: *Scientific Reports* 12.1 (1 May 11, 2022), p. 7719. DOI: 10.1038/s41598-022-11522-z. URL: https://www.nature.com/articles/s41598-022-11522-z. 1

[9] Laurent Dinh, Jascha Sohl-Dickstein, and Samy Bengio. *Density Estimation Using Real NVP*. Feb. 27, 2017. DOI: 10.48550/arXiv.1605.08803. URL: http://arxiv.org/abs/1605.08803. 3, 7

[10] Jian Gao, Jianshe Wu, and JingYi Ding. *HyperEvent:Learning Cohesive Events for Large-scale Dynamic Link Prediction*. July 16, 2025. DOI: 10.48550/arXiv.2507.11836. URL: http://arxiv.org/abs/2507.11836. 4

[11] Dirk Helbing. "Globally Networked Risks and How to Respond". In: *Nature* 497.7447 (May 2013), pp. 51–59. DOI: 10.1038/nature12047. URL: https://www.nature.com/articles/nature12047. 1

[12] Tennessee Hickling and Dennis Prangle. *Flexible Tails for Normalizing Flows*. June 22, 2024. DOI: 10.48550/arXiv.2406.16971. URL: http://arxiv.org/abs/2406.16971. 8

[13] Jonathan Ho et al. *Flow++: Improving Flow-Based Generative Models with Variational Dequantization and Architecture Design*. May 15, 2019. DOI: 10.48550/arXiv.1902.00275. URL: http://arxiv.org/abs/1902.00275. 3, 7

[14] Ryien Hosseini et al. "A Deep Probabilistic Framework for Continuous Time Dynamic Graph Generation". In: *Proceedings of the AAAI Conference on Artificial Intelligence* 39.16 (16 Apr. 11, 2025), pp. 17249–17257. DOI: 10.1609/aaai.v39i16.33896. URL: https://ojs.aaai.org/index.php/AAAI/article/view/33896. 2

[15] Shenyang Huang et al. "Temporal Graph Benchmark for Machine Learning on Temporal Graphs". In: *Advances in Neural Information Processing Systems* 36 (Dec. 15, 2023), pp. 2056–2073. URL: https://proceedings.neurips.cc/paper_files/paper/2023/hash/066b98e63313162f6562b35962671288-Abstract-Datasets_and_Benchmarks.html. 1, 4, 7

[16] Wouter Kool, Herke van Hoof, and Max Welling. *Stochastic Beams and Where to Find Them: The Gumbel-Top-k Trick for Sampling Sequences Without Replacement*. May 29, 2019. URL: http://arxiv.org/abs/1903.06059. 10

[17] Srijan Kumar, Xikun Zhang, and Jure Leskovec. "Predicting Dynamic Embedding Trajectory in Temporal Interaction Networks". In: *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. July 25, 2019, pp. 1269–1278. DOI: 10.1145/3292500.3330895. URL: http://arxiv.org/abs/1908.01207. 1

[18] A. Longa et al. "Generating Fine-Grained Surrogate Temporal Networks". In: *Communications Physics* 7.1 (Jan. 9, 2024), p. 22. DOI: 10.1038/s42005-023-01517-1. URL: https://www.nature.com/articles/s42005-023-01517-1. 2

[19] Antonio Longa et al. *Graph Neural Networks for Temporal Graphs: State of the Art, Open Challenges, and Opportunities*. July 8, 2023. DOI: 10.48550/arXiv.2302.01018. URL: http://arxiv.org/abs/2302.01018. 1

[20] Xiaodong Lu et al. *Improving Temporal Link Prediction via Temporal Walk Matrix Projection*. Oct. 5, 2024. DOI: 10.48550/arXiv.2410.04013. URL: http://arxiv.org/abs/2410.04013. 4

[21] Ashwin Paranjape, Austin R. Benson, and Jure Leskovec. "Motifs in Temporal Networks". In: *Proceedings of the Tenth ACM International Conference on Web Search and Data Mining*. Feb. 2, 2017, pp. 601–610. DOI: 10.1145/3018661.3018731. URL: http://arxiv.org/abs/1612.09259. 1

[22] Romualdo Pastor-Satorras et al. "Epidemic Processes in Complex Networks". In: *Reviews of Modern Physics* 87.3 (Aug. 31, 2015), pp. 925–979. DOI: 10.1103/RevModPhys.87.925. URL: https://link.aps.org/doi/10.1103/RevModPhys.87.925. 1

[23] Farimah Poursafaei et al. *Towards Better Evaluation for Dynamic Link Prediction*. Sept. 12, 2022. DOI: 10.48550/arXiv.2207.10128. URL: http://arxiv.org/abs/2207.10128. 4

[24] Emanuele Rossi et al. *Temporal Graph Networks for Deep Learning on Dynamic Graphs*. Oct. 9, 2020. DOI: 10.48550/arXiv.2006.10637. URL: http://arxiv.org/abs/2006.10637. 1, 4

[25] Rakshit Trivedi et al. "DyRep: Learning Representations over Dynamic Graphs". In: International Conference on Learning Representations. Sept. 27, 2018. URL: https://openreview.net/forum?id=HyePrhR5KX. 4

[26] Yanbang Wang et al. *Inductive Representation Learning in Temporal Networks via Causal Anonymous Walks*. Oct. 31, 2022. DOI: 10.48550/arXiv.2101.05974. URL: http://arxiv.org/abs/2101.05974. 1

[27] Christina Winkler et al. *Learning Likelihoods with Conditional Normalizing Flows*. Nov. 12, 2023. DOI: 10.48550/arXiv.1912.00042. URL: http://arxiv.org/abs/1912.00042. 3, 7

[28] Mingyang Wu, Xiaohui Chen, and Li-Ping Liu. *EDGE++: Improved Training and Sampling of EDGE*. Oct. 28, 2023. DOI: 10.48550/arXiv.2310.14441. URL: http://arxiv.org/abs/2310.14441. 2, 4

[29] Da Xu et al. *Inductive Representation Learning on Temporal Graphs*. Feb. 19, 2020. DOI: 10.48550/arXiv.2002.07962. URL: http://arxiv.org/abs/2002.07962. 1

[30] Le Yu et al. "Towards Better Dynamic Graph Learning: New Architecture and Unified Library". In: Thirty-Seventh Conference on Neural Information Processing Systems. Nov. 2, 2023. URL: https://openreview.net/forum?id=xHNzWHbklj. 1

[31] Xiaohui Zhang et al. *Efficient Neural Common Neighbor for Temporal Graph Link Prediction*. June 12, 2024. DOI: 10.48550/arXiv.2406.07926. URL: http://arxiv.org/abs/2406.07926. 4

A Datasets

Ecuador is firm to firm transaction dataset collected through VAT filings and it is provided by the Internal Revenues Service (IRS) of Ecuador. It comprises fine-grained information about all the transactions between firms within the country of Ecuador between 2007 and 2015. Basic statistics are shown in Table 1, for full details on **Ecuador** dataset we refer to appendix A.4 in the work of Bacilieri et al. [2].

Table 1: Graph dataset statistics.

Dataset	#Nodes	#Edges	#Timesteps
Ecuador[2]	328,640	24,678,048	9
tgbl-flight[15]	18,143	67,169,570	1,385

B Learning

B.1 Learning problem

Our objective is to learn the realization ratio rather than the realization count directly. The ratio is defined as

$$\alpha^{\gamma}_{t,i} = \frac{r^{\gamma}_{t,i}}{d^{\gamma}_{t-1,i}}, \tag{3}$$

where $r^{\gamma}_{t,i}$ is the realization count and $d^{\gamma}_{t-1,i}$ is the vector of generalized degrees at the previous timestep. Let $d_\alpha = 2^{\tau+1}$ denote the dimension of the vector $\vec{\alpha}_{t,i}$.

Both $\alpha^{\gamma}_{t,i}$ and $r^{\gamma}_{t,i}$ are discrete quantities, and so to apply normalizing flows and avoid mode collapse, we first dequantize them following [13]. Specifically, we introduce a dequantizing normalizing flow

$$q_\phi(\cdot \mid \vec{\alpha}_{t,i}, \vec{d}_{t-1,i}), \tag{4}$$

with support bounded by $[0,1]^{d_\alpha}$.

From this flow, we draw noise

$$\vec{\varepsilon} \sim q_\phi(\cdot \mid \vec{\alpha}_{t,i}, \vec{d}_{t-1,i}). \tag{5}$$

Using the sampled noise, we define a dequantized realization ratio:

$$\tilde{\alpha}^{\gamma}_{t,i} = \frac{r^{\gamma}_{t,i} + \varepsilon^{\gamma}}{d^{\gamma}_{t-1,i} + 1}, \tag{6}$$

and the learning objective for $(\vec{\tilde{\alpha}}_{t,i} \mid \vec{d}_{t-1,i})$ and $q_\phi(\vec{\varepsilon} \mid \vec{\alpha}_{t,i}, \vec{d}_{t-1,i})$ as in [13]:

$$\mathbb{E}_{\vec{r}_{t,i} \sim \textbf{Data}} \mathbb{E}_{\vec{\varepsilon} \sim q_\phi(\cdot \mid \vec{\alpha}_{t,i}, \vec{d}_{t-1,i})} \left[\log \frac{p(\vec{\tilde{\alpha}}_{t,i} \mid \vec{d}_{t-1,i})}{q_\phi(\vec{\varepsilon} \mid \vec{\alpha}_{t,i}, \vec{d}_{t-1,i})} \right] \tag{7}$$

Where we can recover the original discrete quantities as:

$$r^{\gamma}_{t,i} = \left\lfloor \tilde{\alpha}^{\gamma}_{t,i} \left(d^{\gamma}_{t-1,i} + 1 \right) \right\rfloor, \qquad \alpha^{\gamma}_{t,i} = \frac{\left\lfloor \tilde{\alpha}^{\gamma}_{t,i} \left(d^{\gamma}_{t-1,i} + 1 \right) \right\rfloor}{d^{\gamma}_{t-1,i} + 1}. \tag{8}$$

B.2 Architecture

For both the learned model $p(\vec{\alpha}_{t,i} \mid \vec{d}_{t-1,i})$, and the associated dequantizer $q_\phi(\vec{\varepsilon} \mid \vec{\alpha}_{t,i}, \vec{d}_{t-1,i})$, we use a conditional RealNVP normalizing flow [9, 27], where only add the conditional dependence in the coupling layers.

We bound the output by adding an additional logistic head to the forward flow:

$$x' = \epsilon + x \cdot (1 - 2\epsilon), \tag{9}$$

and then transform it using the logit function,

$$z = \log\left(\frac{x'}{1-x'}\right). \tag{10}$$

Furthermore we preprocess each element as $d^{\gamma}_{t-1,i} \mapsto \log(1 + d^{\gamma}_{t-1,i})$ and disregard the entries counting the edges corresponding to temporal signature $0 \ldots 0$. The hyperparameters are shown in table 2

Table 2: Hyperparameters for conditional RealNVP normalizing flow experiments

Parameter	**Value**
Data Dimensions	
Input dimension	$2^{\tau+1}$
Context dimension	$2^{\tau+1} - 1$
Main Model Architecture	
Number of flows	6
Hidden layers per flow	2
Hidden dimension	$2^{\tau+4}$
Dequantizer Architecture	
Number of flows	4
Hidden layers per flow	2
Hidden dimension	$2^{\tau+4}$
Context MLP	
Context net layers	8
Context net hidden dim	$8\,(2^{\tau+1} - 1)$
Training Parameters	
Learning rate	10^{-3}
Batch size	300
Logistic ϵ	10^{-6}
τ values tested	$\{2, 3, 4\}$

B.3 Motivation of architecture choice

Here we briefly discuss our motivation to use normalizing flows. As shown in Figure 4, the distribution of realization counts has several properties that make Normalizing Flows well suited for our exploratory work. First, the distribution is multi-modal, which is generally difficult to model and therefore motivates expressive generative models. Second, the data is both discrete and ordinal, which motivates the use of latent variable models with dequantization. While similar setups are possible with other expressive models such as diffusion models, the underlying dequantizer, to our knowledge, is still a normalizing flow. Our approach therefore avoids mixing different architectures for distribution modeling and dequantization.

We further note that it is known that realNVP is not well suited for capturing fat tailed distributions[12], but we leave these complications for future for work.

C Link prediction

To apply the learned model $p_\theta(\vec{\alpha}_{t,i} \mid \vec{d}_{t-1,i})$ for link prediction, we assign a score $s_{t,ij}$ to each triplet (i, j, t). We interpret $\alpha^{\gamma}_{t,i}$ as the sampled probability that node i retains an edge of type γ. To obtain stable estimates for both nodes i and j, we draw $n_{\text{sample}} = 10$ samples of $\vec{\alpha}_{t,i}$ and $\vec{\alpha}_{t,j}$, and then take the average: $\mathbb{E}_{\text{sample}}(\vec{\alpha}_{t,i})$ and $\mathbb{E}_{\text{sample}}(\vec{\alpha}_{t,j})$.

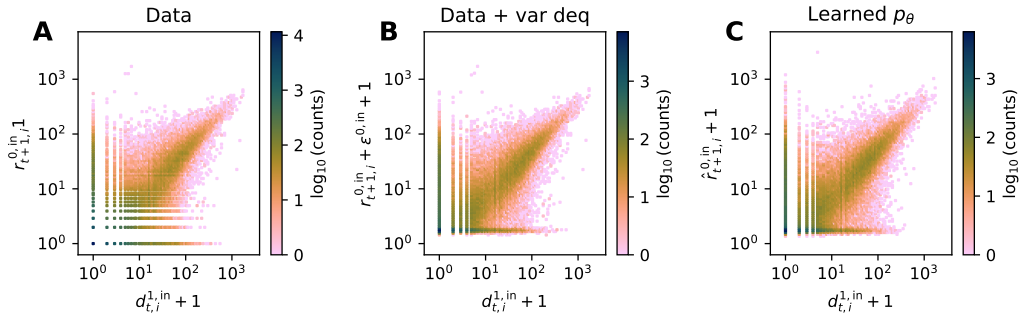

Figure 4: Conditional marginal of $r^{0,\text{in}}_{t+1}$ in the data without dequantization noise, with dequantization noise, and in the learned model, conditioned on the in-degree for $t = 2009$ in the **Ecuador** dataset. For each datapoint $(\vec{r}_{t+1,i}, \vec{d}_{t,i})$, we sample learned dequantization noise $\vec{\varepsilon} \sim q_\phi(\cdot \mid \vec{r}_{t+1,i}, \vec{d}_{t,i})$, as well as the model prediction $\hat{\vec{r}}_{t+1,i} \sim p_\theta(\cdot \mid \vec{d}_{t,i})$. Panel **A** shows the histogram of $(r^{0,\text{in}}_{t+1}, d^{1,\text{in}}_t)$; panel **B** shows the histogram of $(r^{0,\text{in}}_{t+1} + \varepsilon^{0,\text{in}}, d^{1,\text{in}}_t)$; and panel **C** shows the histogram of $(\hat{r}^{0,\text{in}}_{t+1}, d^{1,\text{in}}_t)$. The distribution exhibits a non-trivial, multimodal shape, motivating the use of expressive models.

The score is defined as

$$s(i,j,t) = \mathbb{E}_{\text{sample}}\left(\alpha_{t,i}^{\gamma^{\text{out}}}\right) \cdot \mathbb{E}_{\text{sample}}\left(\alpha_{t,j}^{\gamma^{\text{in}}}\right),$$

where $\gamma^{\text{out}} = \pi^{\text{out}}(\vec{e}_{ij,t})$ and $\gamma^{\text{in}} = \pi^{\text{in}}(\vec{e}_{ij,t})$.

D Graph generation models

We generate directed discrete-time dynamic graphs as follows: We train $p_\theta(\vec{\alpha}_{t,i} \mid \vec{d}_{t-1,i})$ on data from the earliest time in the dataset up to time t_s.

For **Ecuador**, $t_s = 2010$, and for **tgbl-flight**, $t_s = 60 \times 7 \times 86400$. Note that the **tgbl-flight** dataset is discretized in steps of $\Delta t = 86400$ (one day). In the procedure below, we treat $\Delta t = 1$ for ease of exhibition. We also extract the proportion f_{new} of new nodes at each timestep up to t_s.

Then to generate the network we apply Algorithm 1. This procedure exactly preserves the local structure of incoming edges, while approximately preserving the structure of outgoing edges.

Algorithm 1 Discrete-time dynamic graph generation heuristic.

Require: Time steps $\mathcal{T} = \{t_0, t_0 + 1, \dots\}$, new-node fraction f_{new}, graph history $\{G_{t'}\}_{t' < t_0}$.

for all $t \in \mathcal{T}$ **do**
  Add new nodes so that the fraction of new nodes equals f_{new}
  Initialize each new node i with $\vec{d}_{t_s,i} \leftarrow \vec{0}$
  for all nodes i **do**
    Extract $\vec{d}_{t,i}$ from $\{G_{t'}\}_{t' < t}$
    Sample $\vec{r}_{t+1,i} \sim p_\theta(\cdot \mid \vec{d}_{t,i})$
    Split $\vec{r}_{t+1,i}$ into $\vec{r}^{\,\text{in}}_{t+1,i}$ and $\vec{r}^{\,\text{out}}_{t+1,i}$
    Assign in-stubs $\{ r^{\text{in},\gamma}_{t+1,i} \}_\gamma$ to node i
  end for
  for all nodes i and classes γ **do**
    Connect out-stubs of i to nodes $j \neq i$ of class γ,
      without replacement with probability $\alpha^{\text{out},\gamma}_{t+1,i}$,
      using Kool, van Hoof, and Welling [16]
  end for
  Compute updated degrees $\vec{d}_{t+1}$
  Remove nodes i with $\vec{d}_{t+1} = \vec{0}$
end for