Graph Sequential Neural ODE Process for Link Prediction on Dynamic and Sparse Graphs

MONASH University
Griffith UNIVERSITY

Linhao Luo [1] Gholamreza Haffari [1] Shirui Pan [2]

[1]Monash University [2]Griffith University

wsdm

Github

Introduction

Existing approaches of link prediction on dynamic graphs typically require a significant amount of historical data. The missing links over time, which is a common phenomenon in graph data, further aggravates the issue and thus creates extremely sparse and dynamic graphs. How to enable effective link prediction on *dynamic* and *sparse* graphs remains a significant challenge in this area.

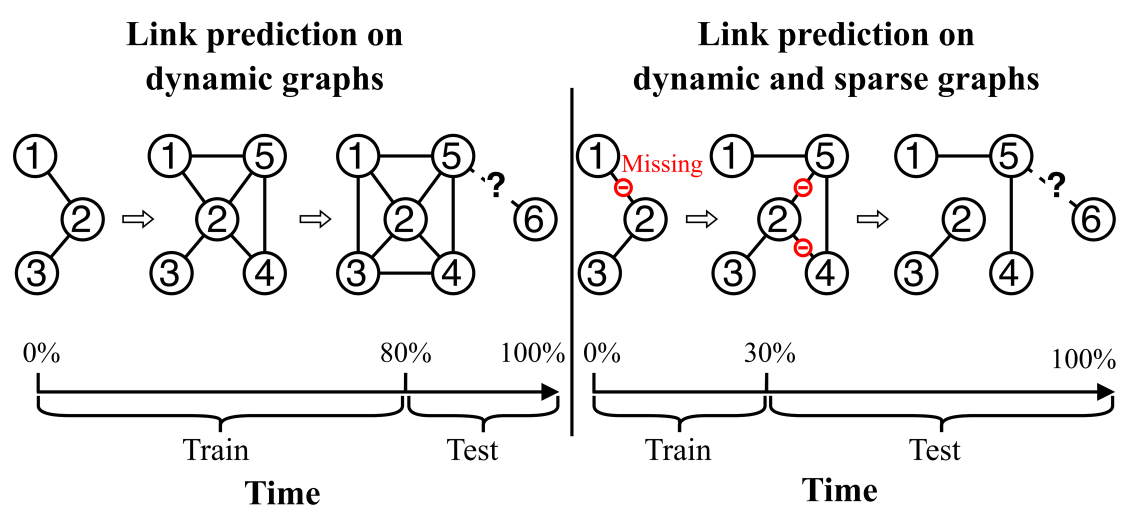

Figure 1. Comparison between link prediction on dynamic graphs and link prediction on dynamic and sparse graphs.

Neural Process (NPs)

Neural process (NP) [1] is a stochastic method that defines **distributions over functions** and can **rapidly adapt** to new observations with **limited data**.

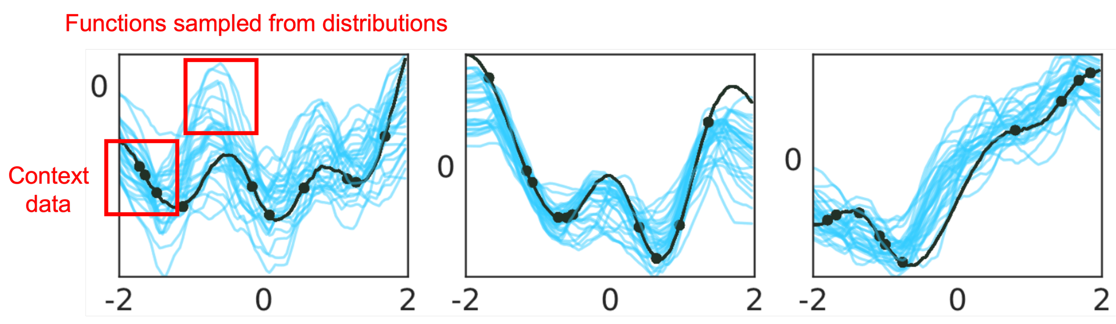

Figure 2. An example of nerual process for 1-D regression task.

NPs aim to learn a distribution from context data (training) that minimizing the prediction loss on the target data (testing).

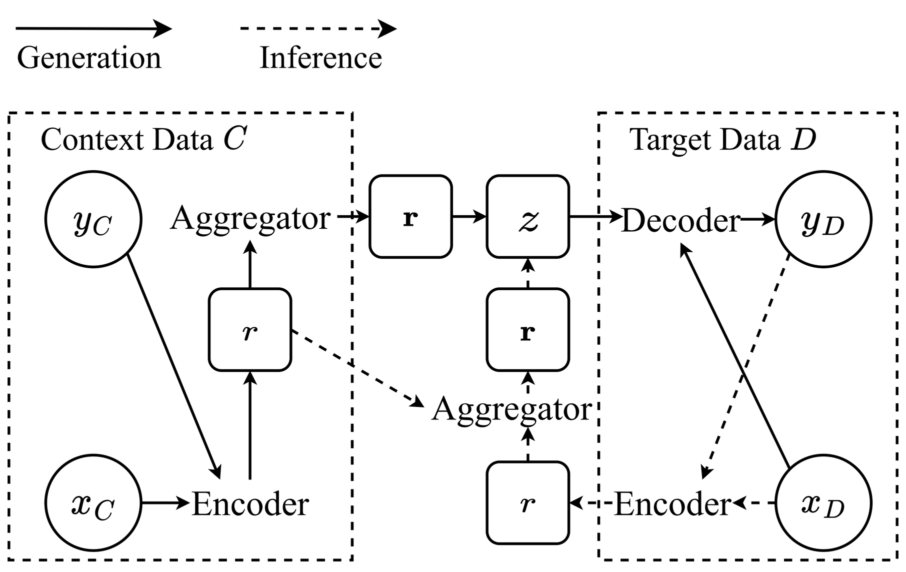

Figure 3. The framework of neural process. Circles denote the input and output data; round squares denote the latent variables.

- **Encoder** learns a low-dimension vector r for each pair (x_C, y_C) in the context data.
- **Aggregator** synthesizes all the r into a global representation $\mathbf{r}$ with an *average pooling*, which parametrizes the function distribution: $z \sim \mathcal{N}(\mu(\mathbf{r}), \sigma(\mathbf{r}))$.
- **Decoder** predicts labels y_D for target data x_D with a sampled z.

Limitiations and Challenges

Sequential and structural dependence

Conventional NPs use a simple average-pooling aggregator, which ignores the sequential and structural dependence between nodes.

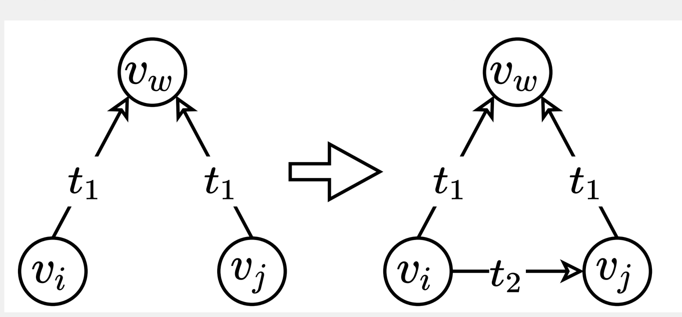

Figure 4. Sequential and structural dependence.

Irregular arriving links in dynamic graphs

Conventional NPs uses a static distribution through time. But links in dynamic graphs arrive irregularly.

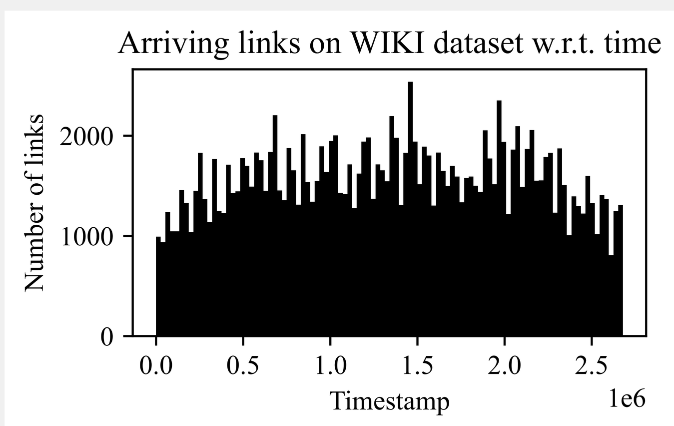

Figure 5. Irregular arriving links in dynamic graphs.

Contributions

1. We propose a novel class of NPs, called Graph Sequential Neural ODE Process (GSNOP) for **link prediction on dynamic and sparse graphs**.
2. We propose a **dynamic graph neural network encoder** and a **sequential ODE aggregator**, which inherits the merits of neural process and neural ODE to model a dynamic-changing stochastic process.
3. GSNOP is **agonist to model structure** that can be incorporated with any existing dynamic graph neural network model.

Graph Sequential Neural ODE Process (GSNOP)

GSNOP combines the merits of **neural process** and **neural ODE** to model the dynamic-changing stochastic process. It can be integrated with any dynamic graph neural network (DGNN) for dynamic graph link prediction.

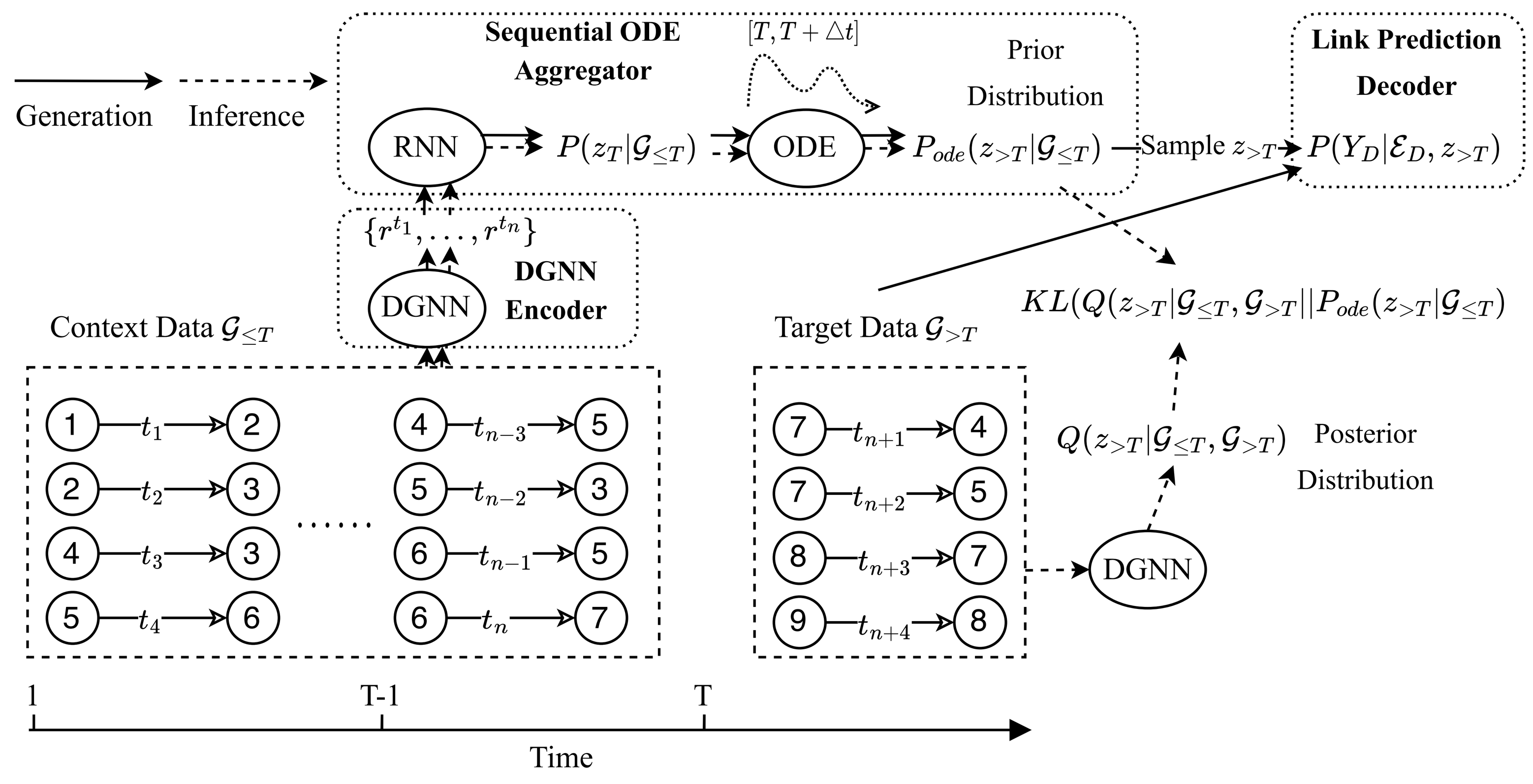

Figure 6. The framework of Graph Sequential Neural ODE Process (GSNOP) for link prediction on dynamic and sparse graphs.

Dynamic Graph Neural Network Encoder

$$h^t = \boxed{\text{DGNN}\left(h^{t-1}, agg(N_{<t}(v))\right)}, \quad r^t = \text{MLP}\left(h_i^t \| h_j^t \| y\right) + t_{emb}, y = \begin{cases} 1, e_C(t) \in \mathcal{G}_{\le T} \\ 0, e_C(t) \notin \mathcal{G}_{\le T} \end{cases},$$

Arbitrary DGNN model, e.g., TGN, TGAT, DySAT….

Sequential ODE Aggregator

We consider the dynamic graph link prediction function as a sequence of dynamic-changing stochastic processes $\{\mathcal{P}_1, \dots, \mathcal{P}_T\}$ and model it with RNN.

$$\mathcal{P}_t = P(z_t|\mathcal{G}_{\le t}) = P(z_t|\mathcal{P}_{t-1}, \mathcal{G}_t),$$
$$\mathbf{r}_t = \text{RNN}(\mathbf{r}_{t-1}, \mathcal{G}_t), t > 1.$$

We apply the Neural ODE to learn the the derivative of the underlying distribution and infer the distribution in the future.

$$\mathbf{r}_{>T} = \mathbf{r}_T + \int_T^{T+\Delta t} f_{ode}(\mathbf{r}_t, t)dt,$$
$$z \sim \mathcal{N}\left(\mu(\mathbf{r}_{>T}), \sigma(\mathbf{r}_{>T})\right).$$

Link Prediction Decoder

$$h_i^t = \text{DGNN}(v_i, N_{<t}(v_i)), h_j^t = \text{DGNN}(v_j, N_{<t}(v_i))$$
$$\text{Sample } z_{>T} \sim \mathcal{N}\left(\mu(\mathbf{r}_{>T}), \sigma(\mathbf{r}_{>T})\right)$$
$$\widetilde{h}_i^t = ReLU(\text{MLP}(h_j^t \| z_{>T})), \widetilde{h}_j^t = ReLU(\text{MLP}(h_j^t \| z_{>T})),$$
$$\hat{y} = Sigmoid\left(\text{MLP}(\widetilde{h}_i^t \| \widetilde{h}_j^t)\right),$$

Optimization

$$\mathcal{L}_{ELBO} = \mathbb{E}_{Q_\psi(z_{>T}|\mathcal{G}_{\le T}, \mathcal{G}_{>T})}[log P_\phi(Y_D|\mathcal{E}_D, z_{>T})] - KL\left(Q_\psi(z_{>T}|\mathcal{G}_{\le T}, \mathcal{G}_{>T}) \| P_{ode}(z_{>T}|\mathcal{G}_{\le T})\right).$$

By minimizing the KL divergence, we can encourage neural ODE to learn the distribution in the future.

Primary results

Performance on different DGNNs (JODIE, DySAT, TGAT, TGN, APAN)

Method	WIKI_0.3		REDDIT_0.3		MOOC_0.3		WIKI_0.1		REDDIT_0.1		MOOC_0.1	
	AP	MRR	AP	MRR	AP	MRR	AP	MRR	AP	MRR	AP	MRR
JODIE	0.3329	0.5595	0.6320	0.8101	0.6690	0.8802	0.1820	0.4476	0.3324	0.6401	0.7100	0.9063
JODIE+GSNOP	**0.4258**	**0.6086**	**0.8171**	**0.8922**	**0.6716**	**0.8853**	**0.2979**	**0.5273**	**0.4705**	**0.7260**	**0.7196**	**0.9066**
DySAT	0.3881	0.6680	0.5705	0.7887	0.6441	0.8807	0.3872	0.6601	0.5353	0.7853	0.6705	0.8968
DySAT+GSNOP	**0.5816**	**0.7716**	**0.7406**	**0.8443**	**0.6528**	**0.8811**	**0.3910**	**0.6628**	**0.5738**	**0.7887**	**0.6718**	**0.8971**
TGAT	0.3253	0.5229	0.8343	0.8797	0.5183	0.7828	0.2766	0.4638	0.4833	0.6523	0.1919	0.4228
TGAT+GSNOP	**0.3701**	**0.5348**	**0.8395**	**0.8820**	**0.5448**	**0.7905**	**0.3366**	**0.4961**	**0.5240**	**0.6807**	**0.1874**	**0.4581**
TGN	0.5541	0.7788	0.7621	0.8587	0.7200	0.9007	0.5676	0.7631	0.6749	0.8029	0.7677	0.9087
TGN+GSNOP	**0.6633**	**0.7816**	**0.8307**	**0.8935**	**0.7445**	**0.9117**	**0.6568**	**0.7674**	**0.7148**	**0.8263**	**0.7714**	**0.9174**
APAN	0.2431	0.5496	0.6166	0.7799	0.6601	0.8774	0.0504	0.1857	0.5601	0.7415	0.7016	0.8967
APAN+GSNOP	**0.4570**	**0.6918**	**0.7119**	**0.8560**	**0.6614**	**0.8790**	**0.1996**	**0.5607**	**0.6126**	**0.7606**	**0.7052**	**0.8982**

Performance on different graph sparsities

Dataset	WIKI_0.3									
Sample Ratio	100%		80%		50%		20%		10%	
Method	AP	MRR	AP	MRR	AP	MRR	AP	MRR	AP	MRR
JODIE	0.3329	0.5595	0.2804	0.5046	0.1464	0.3950	0.0300	0.1598	0.0238	0.1289
JODIE+GSNOP	**0.4258**	**0.6086**	**0.3177**	**0.5171**	**0.2666**	**0.4738**	**0.1845**	**0.4192**	**0.0663**	**0.2784**
DySAT	0.3881	0.6680	0.3856	0.6691	0.3873	0.6676	0.3870	0.6610	0.3121	0.6403
DySAT+GSNOP	**0.5816**	**0.7716**	**0.5539**	**0.6691**	**0.4349**	**0.4349**	**0.3900**	**0.6615**	**0.3159**	**0.6419**
TGAT	0.3253	0.5229	0.3274	0.5238	0.2717	0.4769	0.2573	0.4783	0.2536	0.4722
TGAT+GSNOP	**0.3701**	**0.5348**	**0.3594**	**0.5281**	**0.3630**	**0.5314**	**0.3371**	**0.4955**	**0.2882**	**0.4571**
TGN	0.5541	0.7788	0.5327	0.7560	0.5084	0.7341	0.4361	0.6809	0.4918	0.6831
TGN+GSNOP	**0.6633**	**0.7816**	**0.6395**	**0.7762**	**0.6179**	**0.7685**	**0.5535**	**0.7094**	**0.5566**	**0.6968**
APAN	0.2431	0.5496	0.2198	0.5374	0.1768	0.5179	0.0928	0.2524	0.0431	0.1969
APAN+GSNOP	**0.4570**	**0.6918**	**0.4363**	**0.6821**	**0.2068**	**0.5311**	**0.0735**	**0.2797**	**0.0588**	**0.2128**

Performance on different NP variants (NP, CNP, SNP)

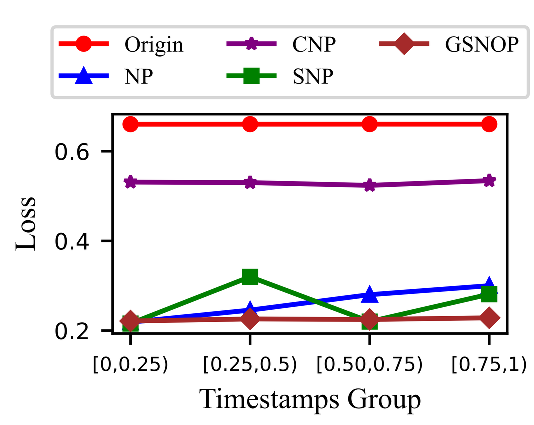

Figure 7. Prediction loss of each group..

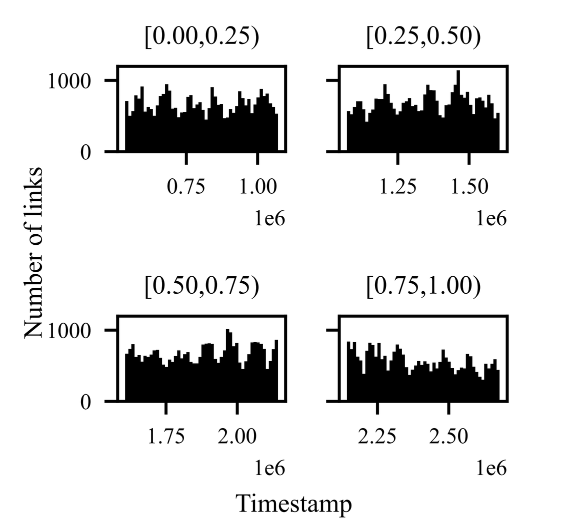

Figure 8. Arriving links of each group.

References

[1] Marta Garnelo, Jonathan Schwarz, Dan Rosenbaum, Fabio Viola, Danilo J Rezende, SM Eslami, and Yee Whye Teh. Neural processes. *arXiv preprint arXiv:1807.01622*, 2018.