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

Linhao Luo, Gholamreza Haffari, Shirui Pan

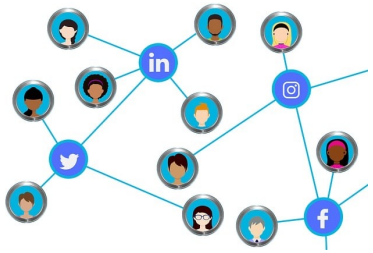
Monash University, and Griffith University

Presenter: Linhao Luo

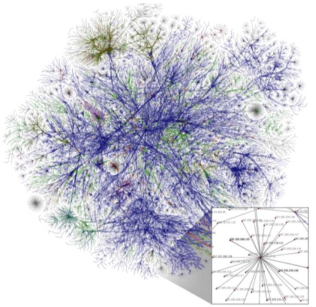
Background

Graph-structured data

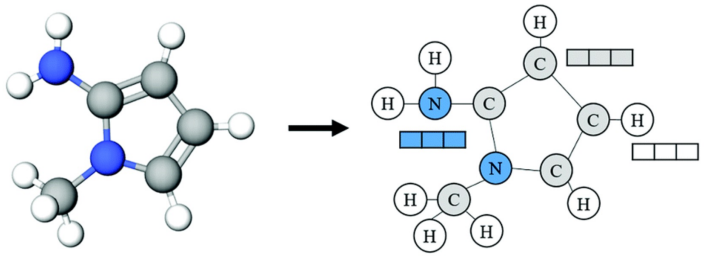
Graphs are ubiquitous in the real-world.



Social Networks



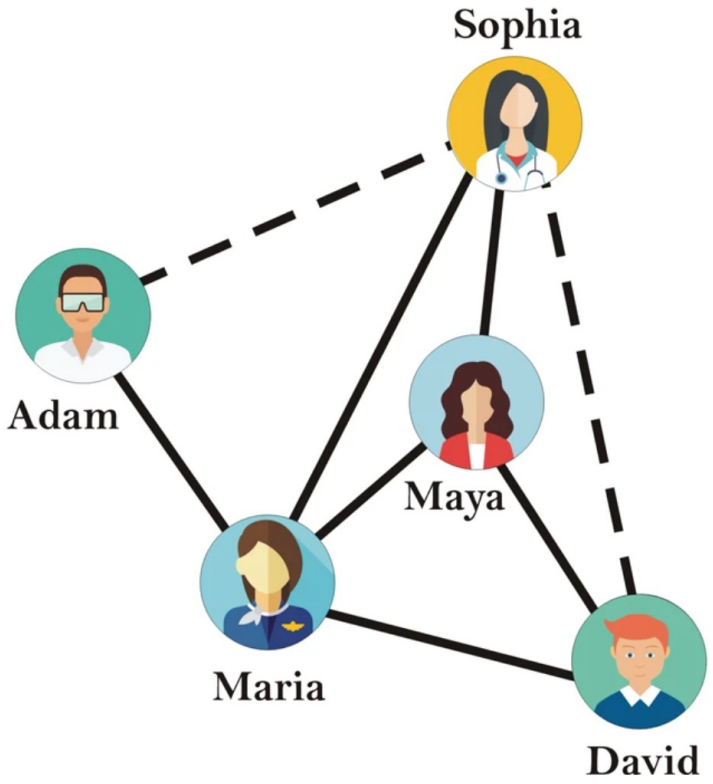
Internet



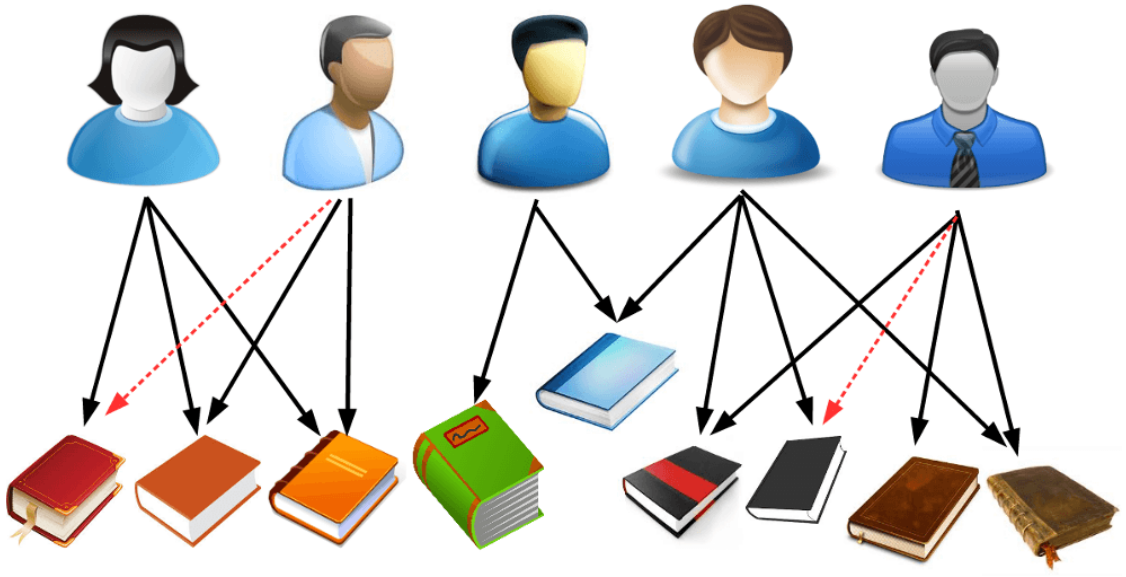
Molecular Graphs

Traffic network

Link Prediction on Graphs

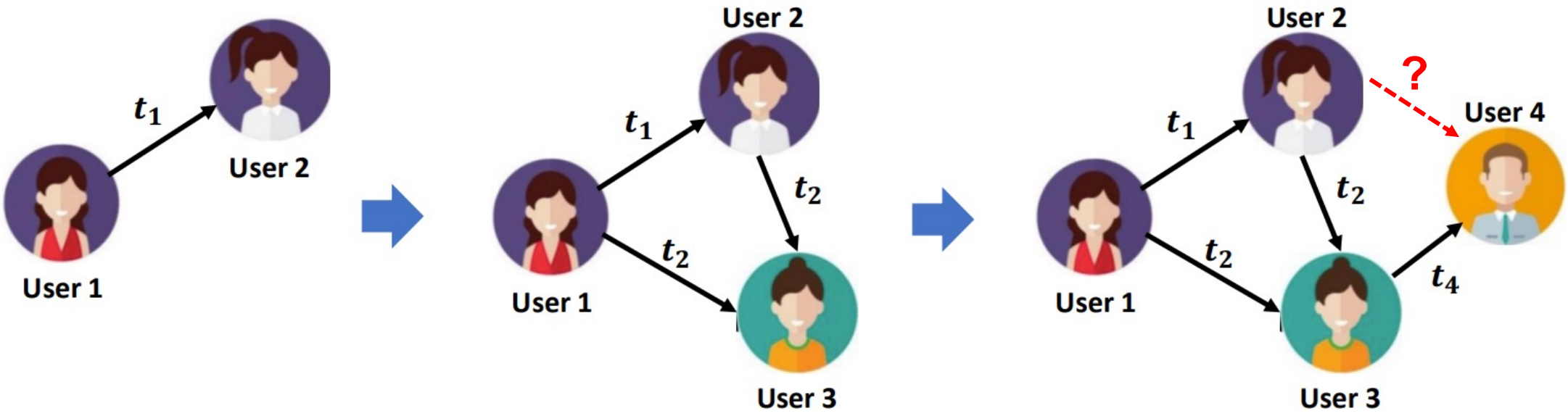


Social network analysis

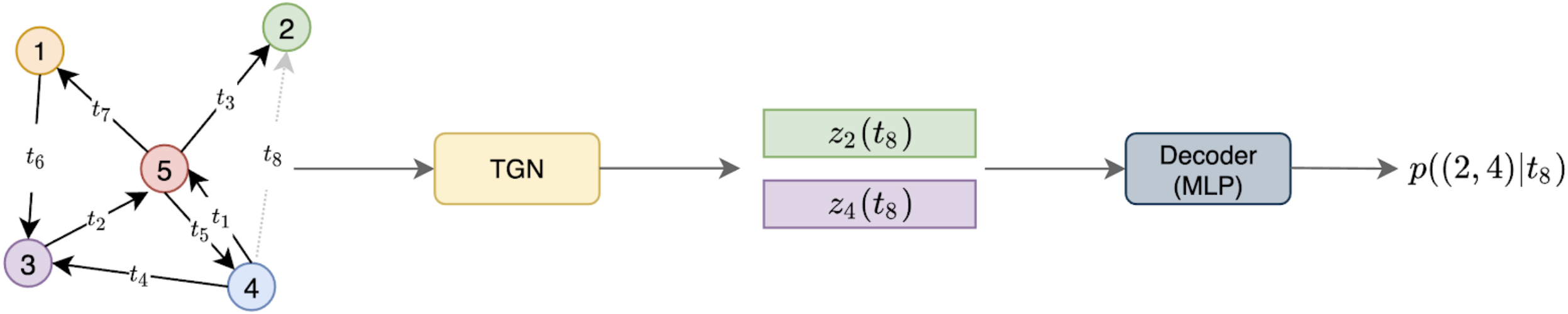


Recommender system

Link Prediction on Dynamic Graphs



Dynamic Graphs Neural Networks (DGNNs)



Emanuele Rossi, Ben Chamberlain, Fabrizio Frasca, Davide Eynard, Federico Monti, and Michael Bronstein. 2020. Temporal Graph Networks for Deep Learning on Dynamic Graphs. In ICML 2020 Workshop on Graph Representation Learning.

Link Prediction on Dynamic and Sparse Graphs

- Existing works require a **large proportion of historical data** for training.
- People want to deploy the model as quick as possible, **which leaves little time to collect the data.**
- Missing of interactions results in extremely **sparse graphs**, which would **incur great bias in the training data** and pose great challenges for training models.

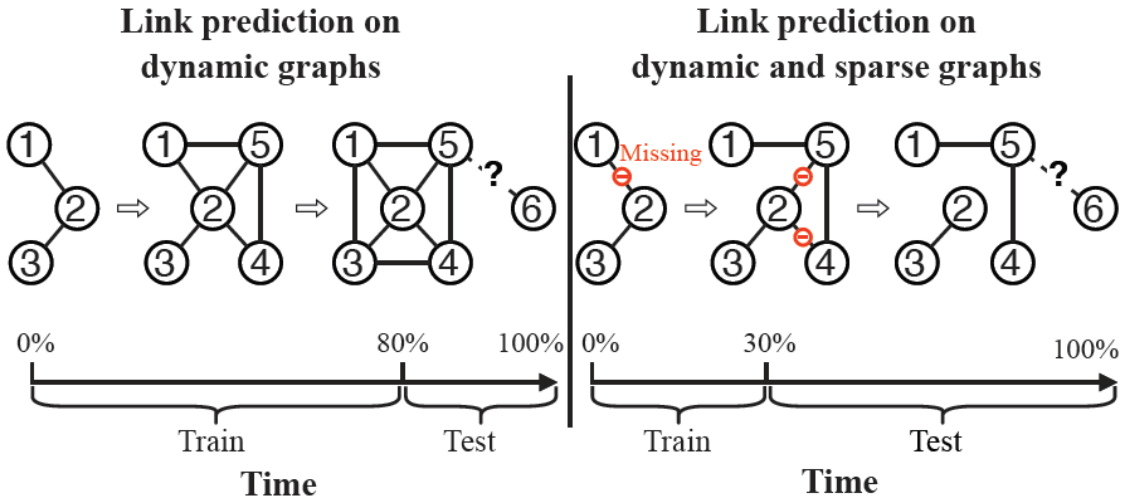


Figure 1: Comparison between link prediction on dynamic graphs and link prediction on dynamic and sparse graphs.

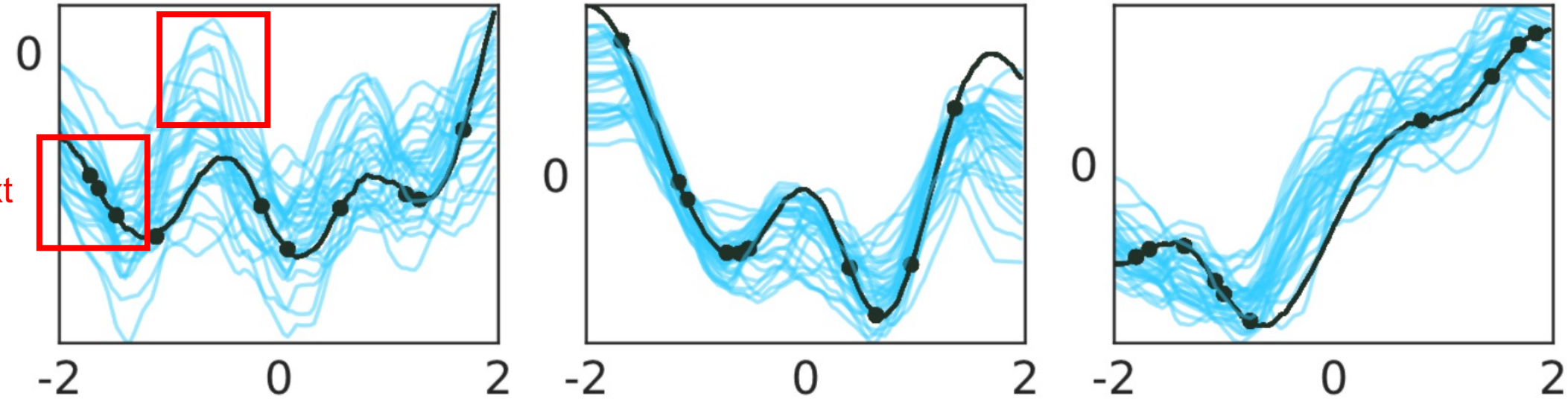
*How to enable effective link prediction on **dynamic** and **sparse** graphs remains a significant challenge in this area.*

Neural Processes (NPs)

- Neural Processes (NPs) is a stochastic method that defines **distributions** over **functions** and can **rapidly adapt** to new observations with **limited data**.

Functions sampled from distributions

Context data



Marta Garnelo, Jonathan Schwarz, Dan Rosenbaum, Fabio Viola, Danilo J Rezende, SM Eslami, and Yee Whye Teh. Neural processes. In ICML Workshop on Theoretical Foundations and Applications of Deep Generative Models, 2018b.

How Neural Process works?

- **Encoder: f_e**
  - Encode each $(x_C, y_C) \in C$, representing with r.

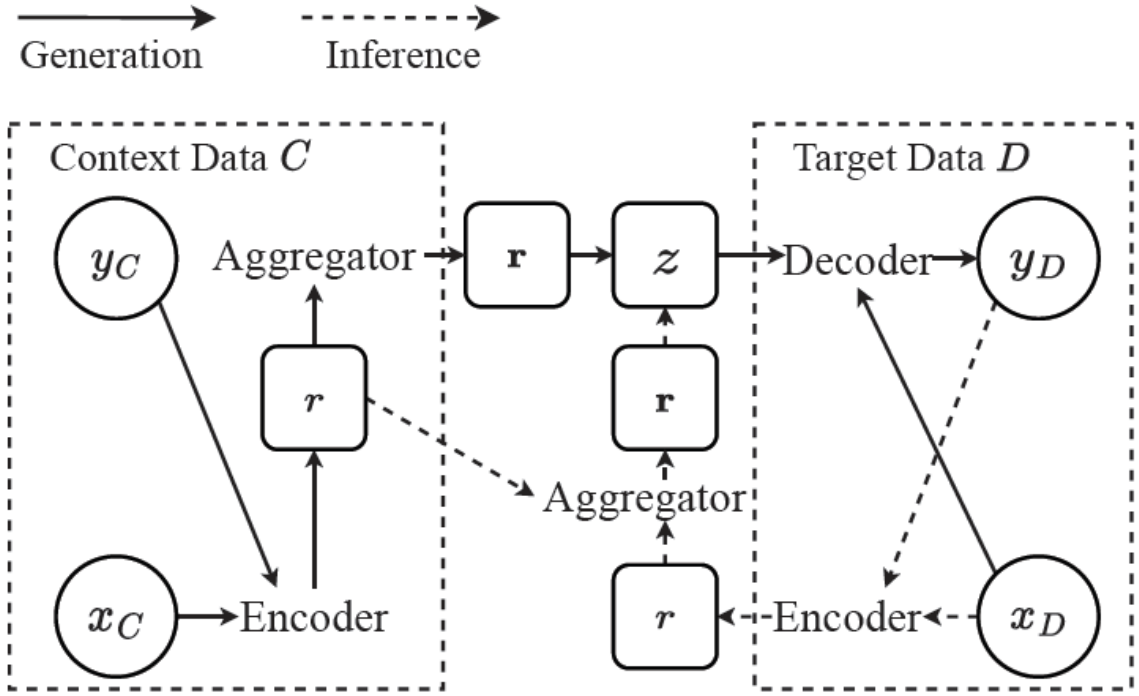
$$r = f_e(x_C, y_C)$$

- **Aggregator: MEAN**

  Summarizes all the r to define a distribution $\mathcal{N}(\mu(\mathbf{r}), \sigma(\mathbf{r}))$

$$\mathbf{r} = MEAN(r_i, \cdots)$$
$$z \sim \mathcal{N}(\mu(\mathbf{r}), \sigma(\mathbf{r}))$$

- **Decoder: g**
  - Sample$z \sim \mathcal{N}(\mu(\mathbf{r}), \sigma(\mathbf{r}))$
  - Predict: $y_D = g\ (x_D, z)$

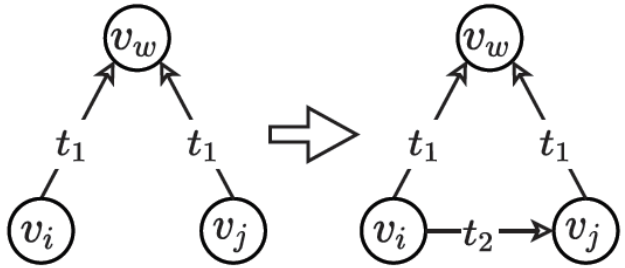


NPs aim to learn a distribution from context data (training) that minimizing the prediction loss on the target data (testing).

Neural Process for Dynamic Graph Link Prediction

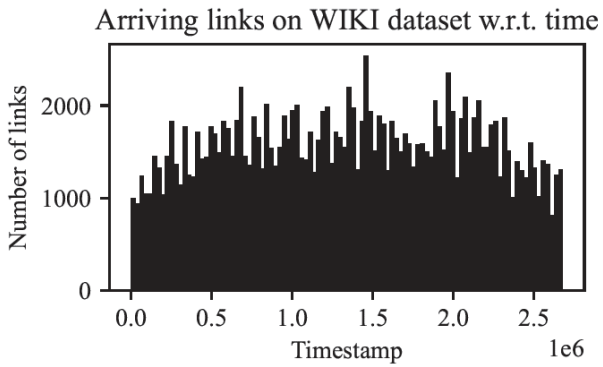
- **Sequential and structural dependence**

Inappropriate to use MEAN aggregator.



- **Irregular arriving links in dynamic graphs.**

Inappropriate to use static distribution.

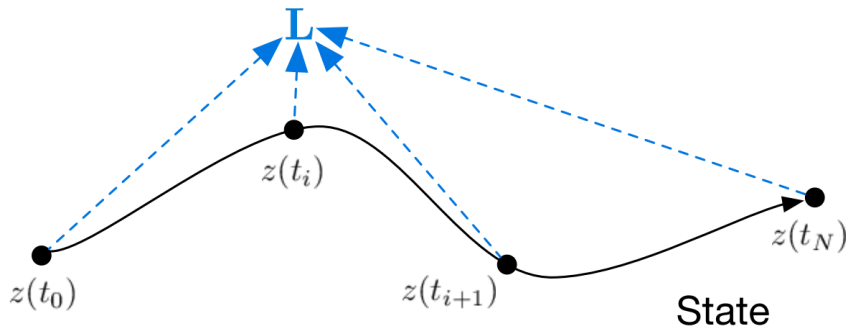


Neural Ordinary Differential Equation (Neural ODE)

Neural ODE defines the derivative of the hidden state with a neural network.

$$\frac{dz(t)}{dt} = f_{ode}(z(t), t),$$

$$z(t) = z(t_0) + \int_{t_0}^{t} f_{ode}(z(t_i), t)dt_i.$$



Ricky TQ Chen, Yulia Rubanova, Jesse Bettencourt, and David K Duvenaud. 2018. Neural ordinary differential equations. Advances in neural information processing systems 31 (2018).

Contributions

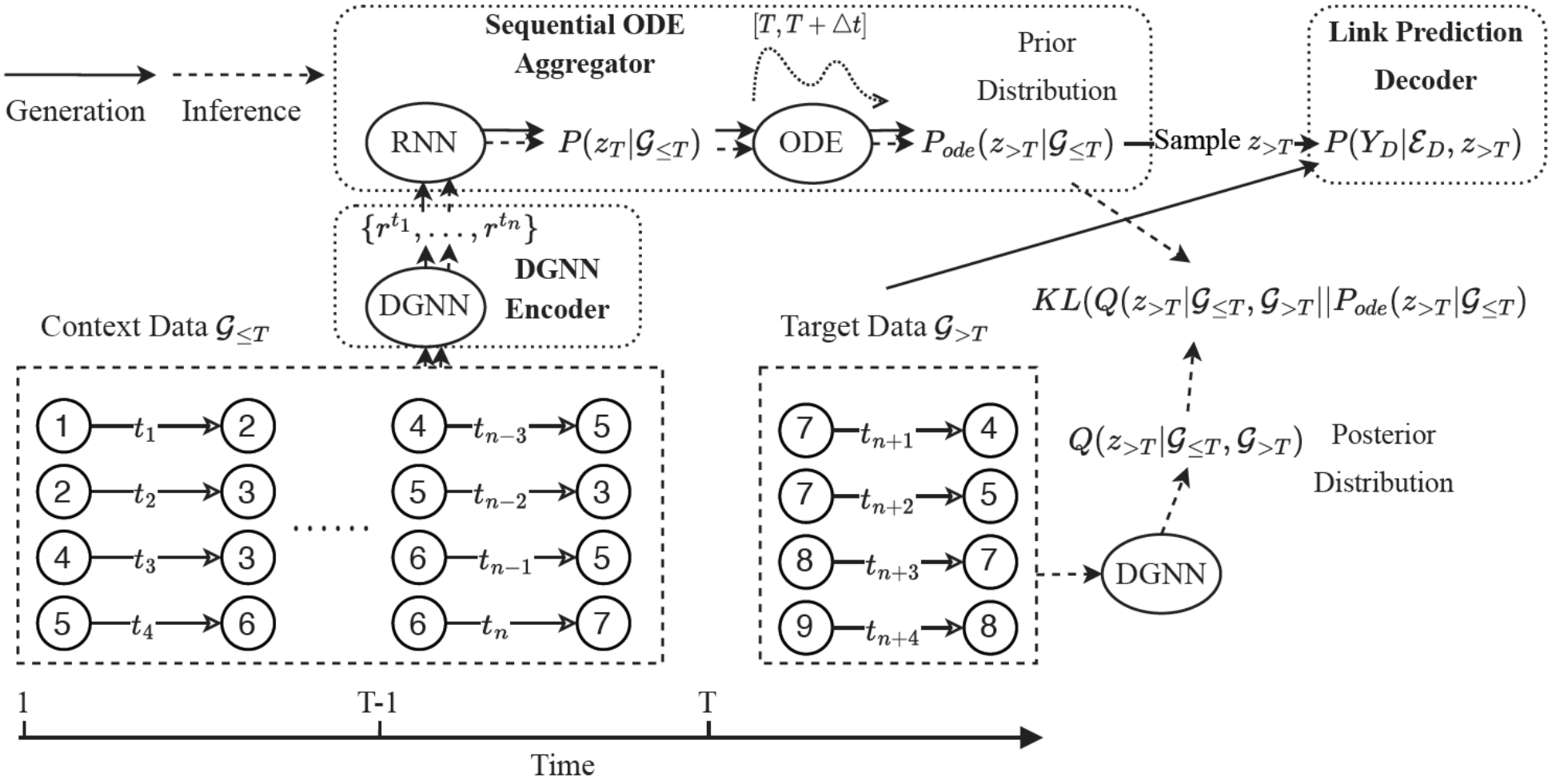
- We propose a novel class of NP, called Graph Sequential Neural ODE Process (GSNOP) for **link prediction on dynamic and sparse graphs.**

- 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**.

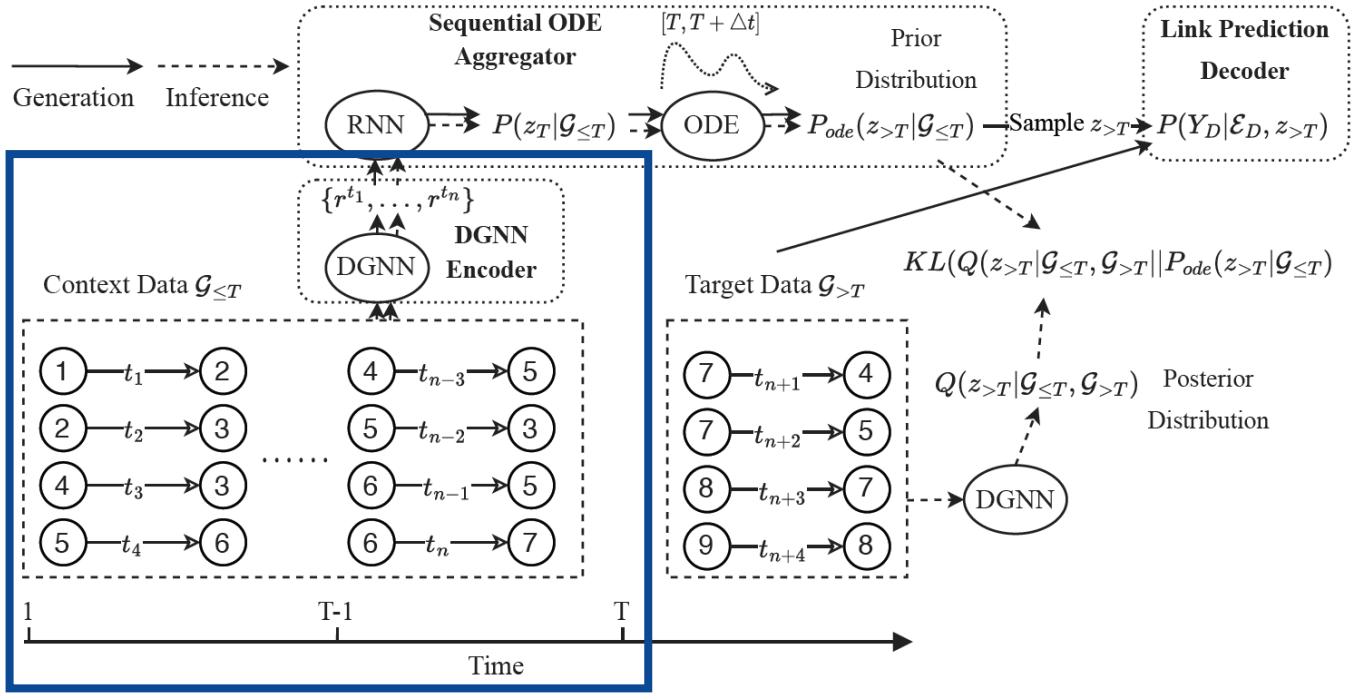
- GSNOP is **agonist to model structure** that can be incorporated with any existing dynamic graph neural network model.

Methodology

Graph Sequential Neural ODE Process (GSNOP)



Dynamic Graph Neural Network Encoder

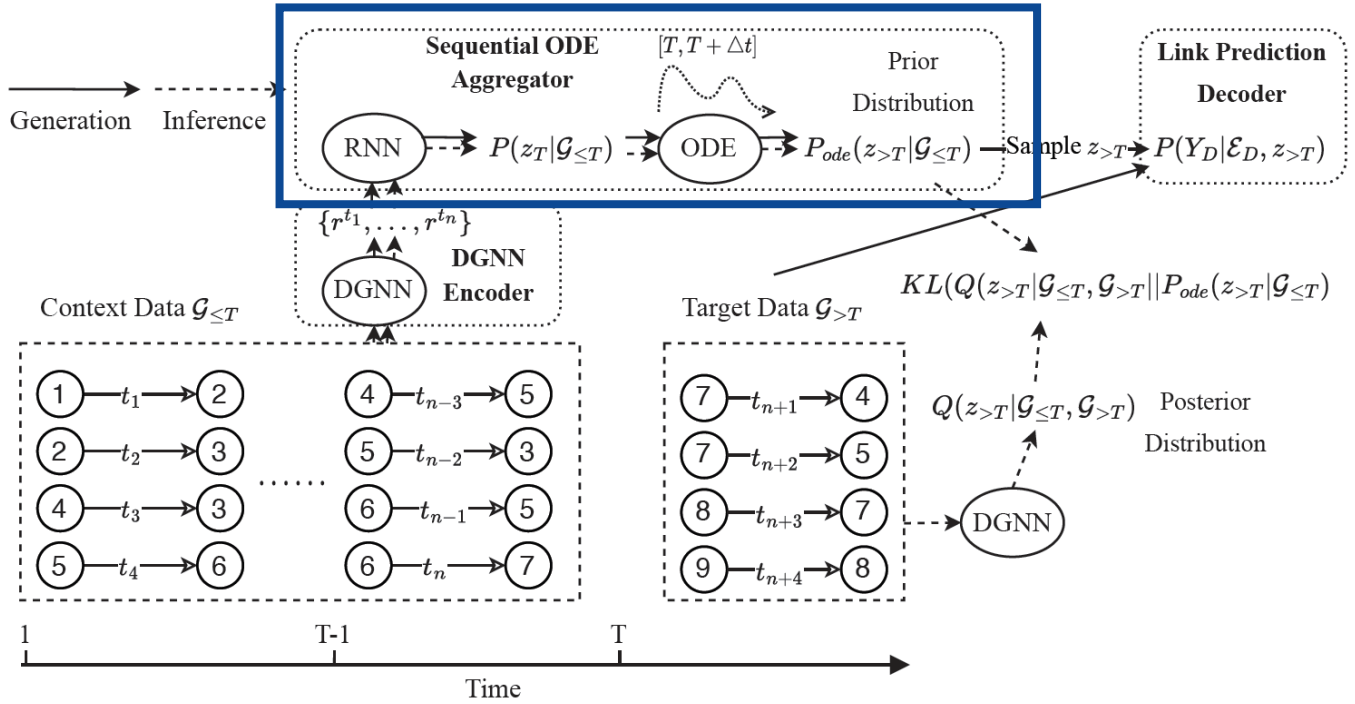


$$h^t = \text{DGNN}\left(h^{t-1}, agg\left(N_{<t}(v)\right)\right),$$

$$r^t = \text{MLP}\left(h_i^t || h_j^t || y\right) + t_{emb}, y = \begin{cases} 1, e_C(t) \in \mathcal{G}_{\leq T} \\ 0, e_C(t) \notin \mathcal{G}_{\leq T} \end{cases},$$

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**

$$\{\mathcal{P}_1, \ldots, \mathcal{P}_T\}.$$

$$\mathcal{P}_t = P(z_t|\mathcal{G}_{\leq t}) = P(z_t|\mathcal{P}_{t-1}, \mathcal{G}_t).$$

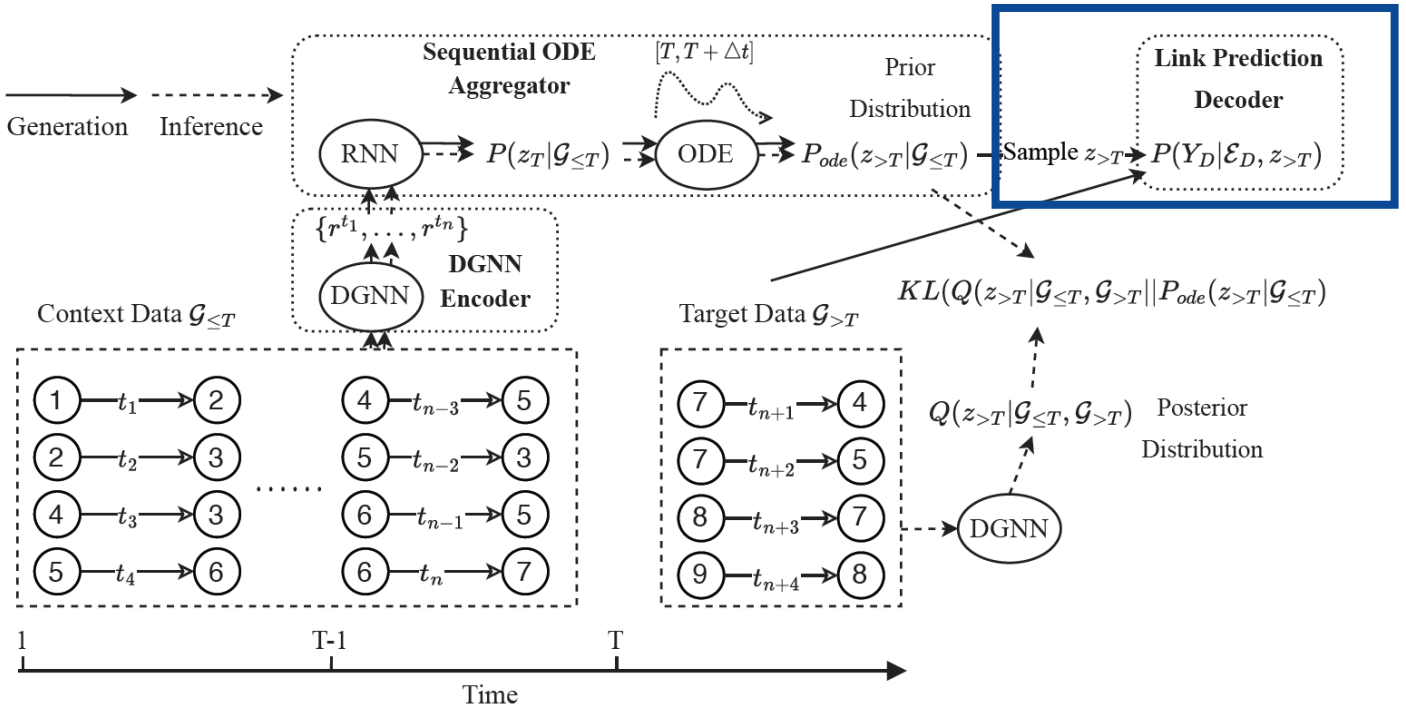
$$\mathbf{r}_t = \mathrm{RNN}(\mathbf{r}_{t-1}, \mathcal{G}_t), t > 1,$$

We apply the **Neural ODE** to learn the the **derivative** of the underlying distribution and **infer the distribution in the future** $\mathcal{N}(\mu(\mathbf{r}_{>T}), \sigma(\mathbf{r}_{>T}))$. Prior distribution

$$\mathbf{r}_{>T} = \mathbf{r}_T + \int_T^{T+\Delta t} f_{ode}(\mathbf{r}_t, t)dt,$$

$$\chi = \mathrm{MLP}(\mathbf{r}_{>T}),$$
$$\mu(\mathbf{r}_{>T}) = ReLU(\mathrm{MLP}(\chi)),$$
$$\sigma(\mathbf{r}_{>T}) = 0.1 + 0.9 * Sigmoid(\mathrm{MLP}(\chi)).$$

Link Prediction Decoder



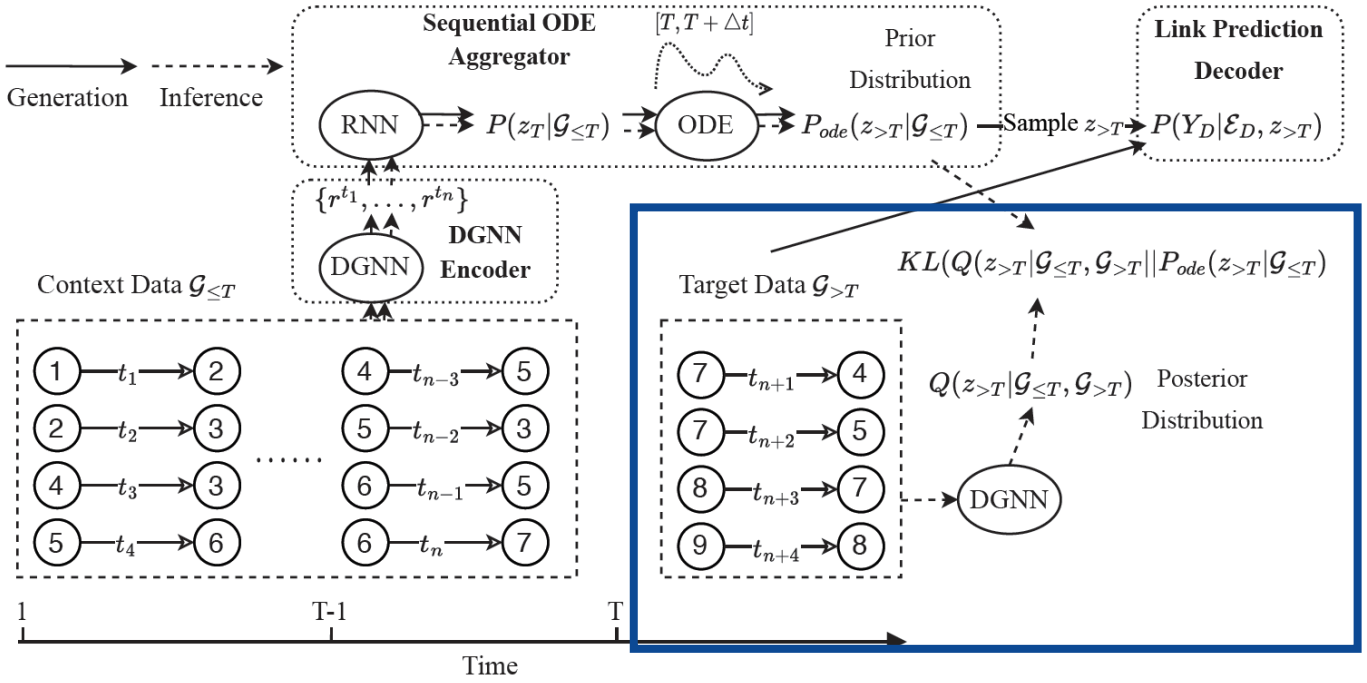
$$h_i^t = \text{DGNN}(v_i, N_{<t}(v_i)), h_j^t = \text{DGNN}(v_j, N_{<t}(v_i))$$

$$\text{Sample } z_{>T} \sim \mathcal{N}\big(\mu(\mathbf{r}_{>T}), \sigma(\mathbf{r}_{>T})\big)$$

$$\widetilde{h_i^t} = ReLU(\text{MLP}(h_j^t||z_{>T})), \widetilde{h_j^t} = ReLU(\text{MLP}(h_j^t||z_{>T})),$$

$$\hat{y} = Sigmoid\big(\text{MLP}(\widetilde{h_i^t}||\widetilde{h_j^t})\big),$$

Optimization



Evidence lower bound:

$$\mathcal{L}_{ELBO} = \mathbb{E}_{Q_\psi(z_{>T}|\mathcal{G}_{\leq T}, \mathcal{G}_{>T})}[logP_\phi(Y_D|\mathcal{E}_D, z_{>T})]$$

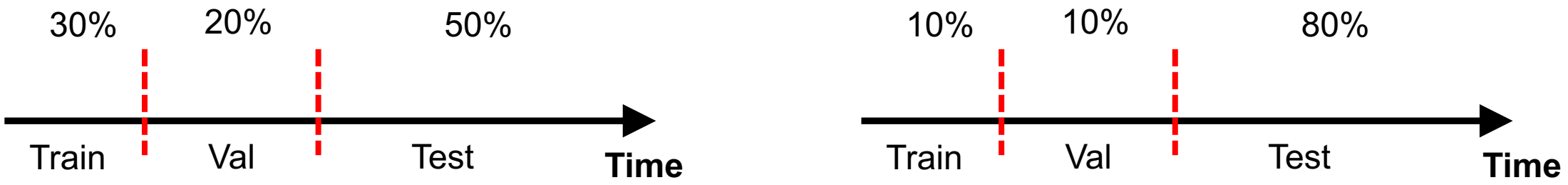
$$- KL\big(Q_\psi(z_{>T}|\mathcal{G}_{\leq T}, \mathcal{G}_{>T})||P_{ode}(z_{>T}|\mathcal{G}_{\leq T})\big).$$

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

Experiments

Experimental setting

Dataset	$\lvert\mathcal{V}_{all}\rvert$	$\lvert\mathcal{E}_{all}\rvert$	max(t)	d_e	Density (0.3)	Density (0.1)
WIKI	9,228	157,474	2.70E+06	172	1.03E-03	2.90E-04
REDDIT	10,985	672,447	2.70E+06	172	3.18E-03	1.01E-03
MOOC	7,047	411,749	2.60E+06	128	4.03E-03	1.52E-03



Split by time

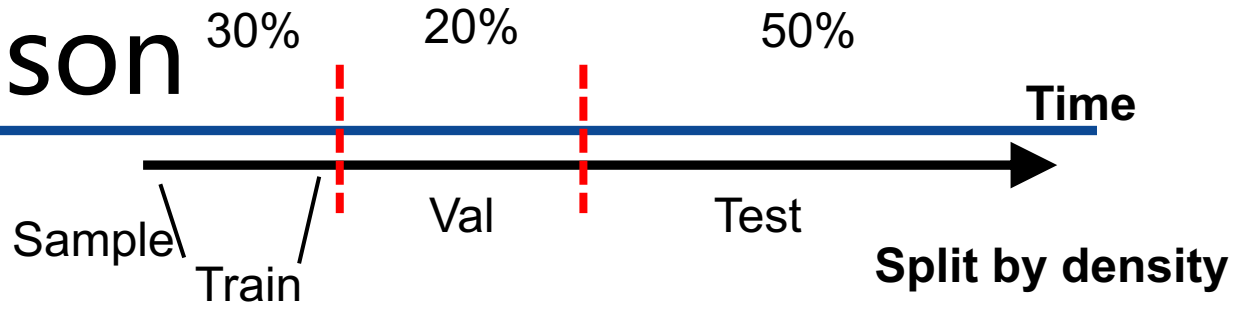
Performance comparison

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

Table 2: Experiments results on WIKI, REDDIT, and MOOC datasets.

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 comparison



- Performance on different densities

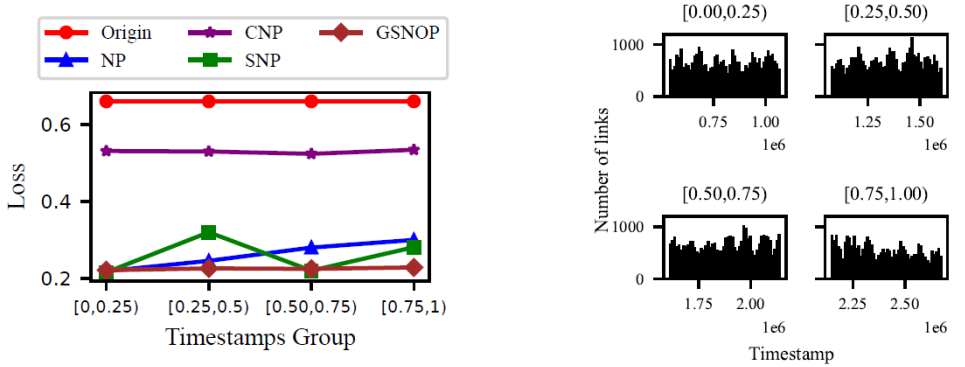
Dataset	WIKI_0.3									
Sample Ratio	100%		80%		50%		20%		10%	
Method	AP	MRR	AP	MRR	AP	MRR	AP	MRR	AP	MRR
JODIE	0.3329	0.5595	0.2804	0.5046	0.1464	0.3950	0.0300	0.1598	0.0238	0.1289
JODIE+GSNOP	**0.4258**	**0.6086**	**0.3177**	**0.5171**	**0.2666**	**0.4738**	**0.1845**	**0.4192**	**0.0663**	**0.2784**
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 comparison

Performance on different NP variants.

Table 4: Comparing with other neural process variants.

Method	WIKI_0.3										WIKI_0.1									
	Origin		NP		CNP		SNP		GSNOP		origin		NP		CNP		SNP		GSNOP	
	AP	MRR	AP	MRR	AP	MRR	AP	MRR	AP	MRR	AP	MRR	AP	MRR	AP	MRR	AP	MRR	AP	MRR
JODIE	0.3868	0.6241	0.4063	0.6118	0.3255	0.5943	0.4173	0.6023	**0.4258**	**0.6086**	0.1946	0.4741	0.2223	0.4699	0.1201	0.3609	0.2804	0.5149	**0.2979**	**0.5273**
DySAT	0.3757	0.6727	0.4226	0.6818	0.3494	0.6552	0.4199	0.6821	**0.5816**	**0.7716**	0.3152	0.6461	0.3151	0.6452	0.3482	0.6386	0.3874	0.6612	**0.3910**	**0.6628**
TGAT	0.3277	0.5171	0.3456	0.5147	0.2881	0.4680	0.3462	0.5173	**0.3701**	**0.5348**	0.2766	0.4638	0.3051	0.4704	0.2407	0.4474	0.2996	0.4646	**0.3366**	**0.4961**
TGN	0.5611	0.7789	0.6632	0.7837	0.6539	0.7766	0.6577	0.7809	**0.6633**	**0.7816**	0.5778	0.7750	0.6362	0.7617	0.5703	0.7347	0.6479	0.7656	**0.6568**	**0.7674**
APAN	0.2191	0.5085	0.0784	0.2701	0.0987	0.3195	0.1090	0.3503	**0.4570**	**0.6918**	0.1598	0.5251	0.0820	0.3115	0.0836	0.2837	0.1135	0.3995	**0.1996**	**0.5607**



a Prediction loss of each group.

b Arriving links of each group.

Figure 5: (a) Prediction loss and (b) arriving links of each group.

Thanks for your listening!