A Additional Algorithm Details

A.1 The Derivation of Approximate Von Neumann Entropy on Temporal Network

In the following, We commence by summarizing the approximation of the undirected graph von Neumann entropy presented by [27]. First, we introduce the von Neumann entropy can be computed from the normalized Laplacian spectrum as follows:

$$S_{VN}(G) = -\operatorname{Tr}(P\log P) = -\sum_{i=1}^{|V|}\frac{\lambda_i}{|V|}\log\frac{\lambda_i}{|V|}, \tag{19}$$

where $\lambda_1,\ldots,\lambda_{|V|}$ are the eigenvalues combinatorial Laplacian matrix. Scaling the normalized Laplacian matrix by the reciprocal of its trace, we obtain a density matrix $\frac{\hat{L}}{|V|}$, The eigenvalues of the density matrix is $\left(\frac{\hat{\lambda}_1}{|V|},\frac{\hat{\lambda}_2}{|V|},\ldots,\frac{\hat{\lambda}_{|V|}}{|V|}\right)$ and thus the von Neumann entropy of density matrix associated with the normalized Laplacian matrix of the graph is defined as

$$S_{VN}(G) = -\sum_{j=1}^{|V|}\frac{\hat{\lambda}_j}{|V|}\ln\frac{\hat{\lambda}_j}{|V|}. \tag{20}$$

The von Neumann entropy above relies on the computation of the normalized Laplacian spectrum, therefore its computational complexity is cubic in the number of nodes. The Taylor expansion for $\ln\frac{\lambda_i}{|V|}$ is

$$\left(\frac{\hat{\lambda}_j}{|V|}-1\right)-\frac{1}{2}\left(\frac{\hat{\lambda}_j}{|V|}-1\right)^2+\frac{1}{3}\left(\frac{\hat{\lambda}_j}{|V|}-1\right)^3-\frac{1}{4}\left(\frac{\hat{\lambda}_j}{|V|}-1\right)^4+\cdots. \tag{21}$$

If we keep the first item of the expansion and discard the remaining that contribute to a small amount, $\ln\frac{\hat{\lambda}_i}{|V|}$ is approximated using $\left(\frac{\hat{\lambda}_j}{|V|}-1\right)$. Then the entropy $S_{VN}(G)$ can be replaced by the quadratic entropy $\sum_j\frac{\hat{\lambda}_j}{|V|}\left(1-\frac{\hat{\lambda}_j}{|V|}\right)$, then we obtain

$$\begin{aligned} S_{VN}(G) &= -\sum_j\frac{\hat{\lambda}_j}{|V|}\ln\frac{\hat{\lambda}_j}{|V|}\simeq\sum_j\frac{\hat{\lambda}_j}{|V|}\left(1-\frac{\hat{\lambda}_j}{|V|}\right)\\ &= \frac{1}{|V|}\sum_j\lambda_j-\frac{1}{|V|^2}\sum_j\lambda_j^2. \end{aligned} \tag{22}$$

Using the fact that $\operatorname{Tr}\left[\hat{\mathbf{L}}^k\right]=\sum_j\hat{\lambda}_j^k$, the quadratic entropy can be rewritten as

$$S_{VN}(G) = \frac{\operatorname{Tr}[\hat{L}]}{|V|}-\frac{\operatorname{Tr}\left[\hat{L}^2\right]}{|V|^2}. \tag{23}$$

The normalized Laplacian matrix $\hat{L}$ has unit diagonal elements, therefore for the trace of the normalized Laplacian matrix we have

$$\operatorname{Tr}\left[\hat{L}\right]=|V|. \tag{24}$$

Similarly, for the trace of the square of the normalized Laplacian, we have

$$\begin{aligned} \operatorname{Tr}\left[\hat{L}^2\right] &= \sum_{u\in V}\sum_{v\in V}\hat{L}_{uv}\hat{L}_{uv}=\sum_{u\in V}\sum_{v\in V}\left(\hat{L}_{uv}\right)^2\\ &= \sum_{u,v\in V u=v}\left(\hat{L}_{uv}\right)^2+\sum_{\substack{u,v\in V\\ u\neq v}}\left(\hat{L}_{uv}\right)^2=|V|+\sum_{(u,v)\in e}\frac{1}{d_u d_v}. \end{aligned} \tag{25}$$

Table 3: Performance of AP(%) for link prediction. The best results in each column are highlighted in bold font and the second-best results are underlined.

Task	Methods	MathOverflow	Bitcoinalpha	Bitcoinotc	Wikipedia
Transductive	JODIE	84.95 ±0.43	90.32 ±0.19	91.50 ±0.19	92.95 ±2.27
	DyRep	80.97 ±0.25	79.42 ±2.23	78.95 ±2.76	94.63 ±0.20
	TGN	81.51 ±1.73	86.47 ±0.42	88.76 ±1.70	98.52 ±0.09
	TGAT	74.35 ±0.29	79.18 ±0.54	79.53 ±0.84	93.18 ±0.13
	CAW	61.40 ±0.28	71.27 ±0.87	79.29 ±0.76	<u>98.82 ±0.12</u>
	TDLG	82.87 ±0.16	91.19 ±0.24	92.24 ±0.25	87.25 ±0.15
	NeurTWs	<u>93.07 ±0.54</u>	<u>94.14 ±0.24</u>	<u>96.17 ±0.08</u>	96.01 ±0.52
	Ours	**98.60 ±0.26**	**99.06 ±0.20**	**98.83 ±0.47**	**99.04 ±0.26**
Inductive	JODIE	68.58 ±0.49	75.02 ±0.20	77.44 ±0.14	89.33 ±5.04
	DyRep	65.65 ±0.44	66.54 ±1.04	65.94 ±0.86	91.94 ±0.27
	TGN	67.04 ±1.42	70.52 ±1.06	79.74 ±1.21	97.83 ±0.16
	TGAT	62.77 ±0.64	67.09 ±0.88	68.32 ±1.84	94.18 ±0.43
	CAW	64.79 ±0.31	70.70 ±0.93	78.21 ±0.29	<u>99.11 ±0.13</u>
	TDLG	70.18 ±2.16	79.53 ±3.19	80.95 ±6.88	53.47 ±2.41
	NeurTWs	<u>92.68 ±0.40</u>	<u>94.16 ±0.27</u>	<u>96.44 ±0.34</u>	96.12 ±0.22
	Ours	**98.41 ±0.17**	**97.91 ±0.69**	**98.55 ±0.33**	**98.83 ±0.10**

Substituting Eq.24 and Eq.25 into Eq.23, the entropy becomes

$$S_{VN}(G) = \frac{\mathrm{Tr}[\hat{L}]}{|V|} - \frac{\mathrm{Tr}\left[\hat{L}^2\right]}{|V|^2} = \frac{|V|}{|V|} - \frac{|V|}{|V|^2} - \sum_{(u,v)\in e} \frac{1}{|V|^2 d_u d_v} = 1 - \frac{1}{|V|} - \frac{1}{|V|^2} \sum_{(u,v)\in e} \frac{1}{d_u d_v}. \tag{26}$$

We project the temporal network to the time-independent 2-D plane as an edge-weighted graph, resulting in a simplified depiction of the underlying network structure at a given time. As a result, The expression of the approximate entropy is quadratic in the number of nodes.

B Additional Experimental Results

B.1 Performances in Average Precision

Table 3 reports the detailed transductive and inductive link prediction results of AP.

B.2 Time Comparison

Fig. 6 compares the training times of ESSEN against the second-strongest baseline NTW. For fairness, we use the same batch size for both models and experiment in the same environment. Note that the running time of ESSEN is down quickly because the approximate thermodynamic quantities have been computed at the first epoch and use cache after that.

C Experimental Setting

C.1 Datasets

We introduce the datasets used in this paper as follows. In Fig. 7, we report the degree distribution on the 30th Day and 270th Day in datasets BitcoinOTC and MathOverflow. In order to align the timestamps, we shift the time so that they begin with zero. Additionally, we renumber the nodes to optimize space usage. Further details regarding the preprocessing steps undertaken to prepare these datasets for our method are discussed below.

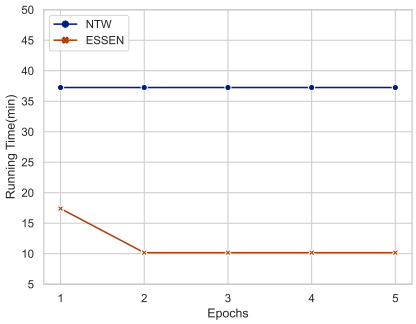
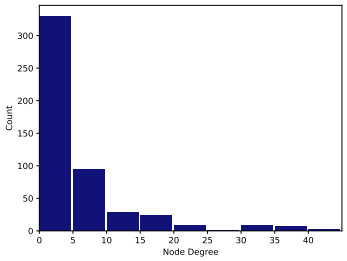
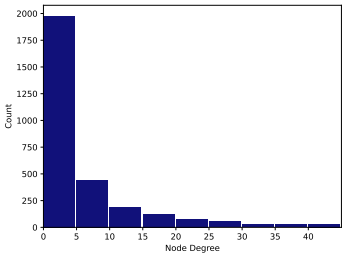


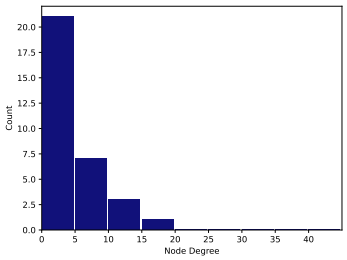
Figure 6: Time Comparison



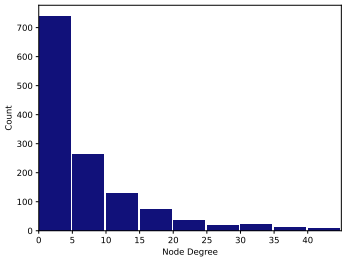
(a) MathOverflow: 30th Day



(b) MathOverflow: 270th Day



(c) BitcoinOTC: 30th Day



(d) BitcoinOTC: 270th Day

Figure 7: Degree distribution at the 30th Day and 270th Day in different networks.

- **MathOverflow dataset.**[1] It is a temporal network of interactions on the stack exchange website Math Overflow. The nodes represent users, and the edges represent the answers to questions. For example, a directed edge (u, v, t) represents user u answered user v's question at time t. Since edge features and node features are not provided, we use the all-zero vectors instead.
- **BitcoinOTC dataset.**[2] It is a who-trusts-whom network of people trade using Bitcoin on the BitcoinOTC platform. Each line records a trade from rater to rate and the rating ranges from -10 to +10 in step 1.
- **Bitcoin Alpha dataset.**[3] It is a similar network to the Bitcoin Alpha platform. Both of the datasets have no edge features or node features, so we initialize them as all-zero vectors.
- **Wikipedia dataset.**[4] It is a dataset of edited records from Wiki pages over a month. We use the top-edited pages and active users as nodes, and each row in our data represents a user editing a page. This dataset records user editing of pages over the course of a month. The

[1] https://snap.stanford.edu/data/sx-mathoverflow.html
[2] https://snap.stanford.edu/data/soc-sign-bitcoin-otc.html
[3] https://snap.stanford.edu/data/soc-sign-bitcoin-alpha.html
[4] http://snap.stanford.edu/jodie/wikipedia.csv

timestamp indicates the time when the user edited the page. As with the Reddit dataset, the features of these nodes were processed through LIWC. The user labels indicate if users are temporarily banned from editing.

C.2 Baselines

The introduction of baselines and their setting details are shown as follows. Baselines not specially mentioned use the default settings of the cited paper.

- **JODIE** uses two recurrent neural networks (RNNs) to learn trajectories of users and items, and updates the embedding when the interaction occurs. we set the number of epochs to 50 and the dimensions of node and time embedding to 100.
- **DyRep** is a temporal point process model capturing both topological evolution and nodes' activities. we set the number of epochs to 50 and the patience for early stopping to 5.
- **TGAT** utilizes a self-attention mechanism and presents a novel encoding method to learn graph embedding inductively. The batch size is 200 and the number of epochs is 50. We set the dimensions of node and time embedding to 100 and 20 neighbors are sampled in aggregation.
- **TGN** is a generic and efficient framework for deep learning on dynamic graphs for discrete representation. We set the number of runs to 10 in our experiments. The max number of sampling neighbors is set to 10 and two heads are used in the attention layer. The dropout probability is 0.1 and the learning rate is 0.0001.
- **CAW** utilizes a new anonymization strategy to represent a temporal network inductively. We set the dimension of the positional embedding to 108, batch size to 64, and bias to 1e-5. The maximum number of neighbors when sampling is 64.
- **TDLG** constructs line graphs to model edges directly instead of computing from node embedding. We discard the attributes of the Wikipedia dataset because the original module could not process data with attributes.
- **NeurTWs** improves the causal anonymous walks strategy in CAW and considers structural and tree traversal properties in the process of walking. The dimension of position embedding is 108 and 32 neighbors are sampled for each node. The temporal bias, spatial bias, and ee bias are set to 1e-5, 1, and 0 respectively.