# EasyDGL: Encode, Train and Interpret for Continuous-time Dynamic Graph Learning (supplementary material)

## Appendix A Proof of Theorem 1

We provide the proof details of our main result, presented in Theorem 1. The main procedure follows the seminar work [2] and we note that the challenge lies in Lemma. 2 which we believe is new and different from results in [4], [5].

**Theorem 1 (Error Bound).** *For $\widetilde{\mathbf{L}} = \widetilde{\mathbf{U}}\widetilde{\mathbf{\Sigma}}\widetilde{\mathbf{U}}^\top$ in Algorithm 1, and $\zeta = 1 + \sqrt{\frac{r}{p-1}} + \frac{e\sqrt{r+p}}{p}\sqrt{s-r}$,*

$$\mathbf{E} \parallel \mathbf{L} - \widetilde{\mathbf{L}} \parallel_2 \leq \zeta^{1/q} \parallel \mathbf{L} - \widetilde{\mathbf{L}}_r \parallel_2 + \left(1 + \zeta^{1/q}\right) \frac{n}{\sqrt{s}} \mathbf{L}^*_{i,i},$$

*where $\mathbf{L}^*_{i,i} = \max_i \mathbf{L}_{i,i}$ and $\mathbf{L}_r$ is the best rank-r approximation.*

### A.1 Preliminaries and Existing Results

For the sake of completeness, we present the definitions and existing results used in the proof.

**Definition 1.** (**Orthogonal Projector**, [1]). A matrix $\mathbf{P}$ is called an orthogonal projector if $\mathbf{P} = \mathbf{P}^\top = \mathbf{P}^2$.

**Lemma 1.** *As shown in [6], the matrices $\mathbf{G} \in \mathbb{R}^{n\times n}$ and $\mathbf{F} \in \mathbb{R}^{n\times n}$ satisfy,*

$$\max_{1\leq i\leq n} |\sigma_i(\mathbf{G}) - \sigma_i(\mathbf{F})| \leq \quad \parallel \mathbf{G} - \mathbf{F} \parallel_2 \tag{1}$$

$$\sum_{i=1}^{n} \Big(\sigma_i(\mathbf{G}) - \sigma_i(\mathbf{F})\Big)^2 \leq \quad \parallel \mathbf{G} - \mathbf{F} \parallel_F^2 . \tag{2}$$

**Theorem 2.** *(**Proposition 1, [2]**). Given a real l-by-l matrix $\mathbf{A}$ with eigenvalues $\sigma_1 \geq \ldots \geq \sigma_l$, choose a target rank $k$ and an oversampling parameter $p \geq 2$, where $k + p \leq l$. Draw an $l$-by-$(k + p)$ standard Gaussian matrix $\Omega$, then construct the sample matrix $\mathbf{A}^q\mathbf{\Omega}$ where $q \geq 1$. The orthonormal basis $\mathbf{Q}$ of matrix $\mathbf{A}^q\mathbf{\Omega}$ (i.e., $\mathbf{A}^q\mathbf{\Omega} = \mathbf{Q}\mathbf{Q}^\top\mathbf{A}^q\mathbf{\Omega}$) satisfies*

$$\mathbb{E} \parallel (\mathbf{I} - \mathbf{Q}\mathbf{Q}^\top)\mathbf{A} \parallel_2 \leq \zeta^{1/q}\sigma_{k+1}(\mathbf{A}), \tag{3}$$

*where $\zeta = 1 + \sqrt{\frac{k}{p-1}} + \frac{e\sqrt{k+p}}{p}\sqrt{l-k}$.*

**Theorem 3.** *(**Theorem 10.5, [2]**). Suppose that $\mathbf{A}$ is a real l-by-l matrix with eigenvalues $\sigma_1 \geq \ldots \geq \sigma_l$. Choose a target rank $k$ and an oversampling parameter $p \geq 2$, where $k + p \leq l$. Draw an $l \times (k + p)$ standard Gaussian matrix $\Omega$, and construct the sample matrix $\mathbf{A\Omega}$. Then,*

$$\mathbb{E} \parallel (\mathbf{I} - \mathbf{Q}\mathbf{Q}^\top)\mathbf{A} \parallel_F \leq \left(1 + \frac{k}{p-1}\right)^{1/2} \left(\sum_{i>k} \sigma_i^2\right)^{1/2}, \tag{4}$$

*where $\mathbf{Q}$ is the orthonormal basis of the range of matrix $\mathbf{A\Omega}$ such that $\mathbf{A\Omega} = \mathbf{Q}\mathbf{Q}^\top\mathbf{A\Omega}$.*

**Theorem 4.** *(**Corollary 2, [4]**). Suppose that $\mathbf{X}$ is a real d-by-n matrix. Choose a set $\mathcal{S}$ of size $l$ at random without replacement from $\{1, 2, \ldots, n\}$, and let $\mathbf{H}$ equals the columns of $\mathbf{X}$ corresponding to indices in $\mathcal{S}$. Let $\mathbf{H}\mathbf{H}^\top$ be an approximation to $\mathbf{X}\mathbf{X}^\top$, then*

$$\mathbb{E} \parallel \mathbf{X}\mathbf{X}^\top - \kappa\mathbf{H}\mathbf{H}^\top \parallel_F \leq \frac{n}{\sqrt{l}} \max_i \parallel \mathbf{X}_{*,i} \parallel^2, \tag{5}$$

*where $\kappa = \frac{n}{l}$ is a non-zero scaling parameter, and $\parallel \mathbf{X}_{*,i} \parallel$ is the Euclidean norm of the $i^{\text{th}}$ column of matrix $\mathbf{X}$.*

### A.2 Proof of Theorem 1

*Proof.* Since the graph Laplacian matrix $\mathbf{L}$ is a symmetric positive semidefinite matrix, we can write it as:

$$\mathbf{L} = \mathbf{X}^\top\mathbf{X},$$

where $\mathbf{X} \in \mathbb{R}^{d\times n}$ and $d$ is the rank of matrix $\mathbf{L}$.

Let $\mathbf{S} = \{0, 1\}^{n\times s}$ be a column sampling matrix where $\mathbf{S}_{i,j}$ equals to 1 if the $i^{\text{th}}$ column of $\mathbf{L}$ is chosen in the $j^{\text{th}}$ random trial and equals to 0 otherwise. Then, $\mathbf{C} = \mathbf{X}^\top\mathbf{H}$ and $\mathbf{A} = \mathbf{H}^\top\mathbf{H}$ where $\mathbf{H} = \mathbf{XS}$.

We take $\mathbf{R} = \mathbf{H}\mathbf{A}^{-1/2}\mathbf{Q}$, then

$$\begin{aligned}\widetilde{\mathbf{L}} &= \mathbf{C}\mathbf{A}^{-1/2}\widetilde{\mathbf{V}}\widetilde{\mathbf{V}}^\top\mathbf{A}^{-1/2}\mathbf{X}^\top \\ &= \mathbf{X}^\top\mathbf{H}\mathbf{A}^{-1/2}\mathbf{Q}\mathbf{Q}^\top\mathbf{A}^{-1/2}\mathbf{H}^\top\mathbf{X} \\ &= \mathbf{X}^\top\mathbf{P}_{\mathbf{R}}\mathbf{X} = \mathbf{X}^\top\mathbf{U}_{\mathbf{R}}\mathbf{U}_{\mathbf{R}}{}^\top\mathbf{X},\end{aligned}$$

where $\mathbf{P}_{\mathbf{R}} = \mathbf{H}\mathbf{A}^{-1/2}\mathbf{Q}\mathbf{Q}^\top\mathbf{A}^{-1/2}\mathbf{H}^\top$ is an orthogonal projector and $\mathbf{U}_{\mathbf{R}}$ is the orthonormal basis of matrix $\mathbf{R}$.

To bound the approximate error, we have

$$\begin{aligned}&\parallel \mathbf{L} - \widetilde{\mathbf{L}} \parallel_2 \\ &= \parallel \mathbf{X}^\top\mathbf{X} - \mathbf{X}^\top\mathbf{U}_{\mathbf{R}}\mathbf{U}_{\mathbf{R}}{}^\top\mathbf{X} \parallel_2 \\ &\overset{(a)}{=} \parallel \mathbf{X}^\top\mathbf{X} - (\mathbf{P}_{\mathbf{R}}\mathbf{X})^\top\mathbf{P}_{\mathbf{R}}\mathbf{X} \parallel_2 \\ &\overset{(b)}{=} \parallel \mathbf{X} - \mathbf{U}_{\mathbf{R}}\mathbf{U}_{\mathbf{R}}{}^\top\mathbf{X} \parallel_2^2 \\ &= \parallel \mathbf{X} - \mathbf{X}\mathbf{U}_{\mathbf{R}}\mathbf{U}_{\mathbf{R}}{}^\top \parallel_2^2 \\ &= \parallel \mathbf{X}\mathbf{X}^\top - \mathbf{X}\mathbf{U}_{\mathbf{R}}\mathbf{U}_{\mathbf{R}}{}^\top\mathbf{X}^\top \parallel_2,\end{aligned}$$

where (a) holds due to the orthogonal project $\mathbf{P}_{\mathbf{R}}$ satisfying $\mathbf{P}_{\mathbf{R}} = \mathbf{P}_{\mathbf{R}}{}^\top = \mathbf{P}_{\mathbf{R}}{}^2$; (b) holds due to $\parallel \mathbf{AB} \parallel_2 = \parallel \mathbf{BA} \parallel_2$ for any $\mathbf{A} \in \mathbb{R}^{m\times n}$ and $\mathbf{B} \in \mathbb{R}^{n\times m}$;

Using Lemma 2 in Sec. A.3,

$$\begin{aligned}&\| \mathbf{X}\mathbf{X}^\top - \mathbf{X}\mathbf{U}_\mathbf{R}\mathbf{U}_\mathbf{R}{}^\top\mathbf{X}^\top \|_2^2\\ \le& \| \mathbf{X}\mathbf{X}^\top - \kappa\mathbf{R}(\mathbf{H}\mathbf{Q})^\top\mathbf{H}\mathbf{Q}\mathbf{R}^\top \|_2\\ \le& \| \mathbf{X}\mathbf{X}^\top - \kappa\mathbf{H}^\top\mathbf{H} \|_2\\ &\qquad + \kappa \| \mathbf{H}\mathbf{H}^\top - \mathbf{R}(\mathbf{H}\mathbf{Q})^\top\mathbf{H}\mathbf{Q}\mathbf{R}^\top \|_2, \end{aligned} \tag{6}$$

where the last step holds due to the triangle inequality.

Since $\mathbf{A}^{-1/2}\mathbf{H}^\top\mathbf{H}\mathbf{A}^{-1/2} = \mathbf{I}_s$, it gives

$$\begin{aligned}&\| \mathbf{H}\mathbf{H}^\top - \mathbf{R}(\mathbf{H}\mathbf{Q})^\top\mathbf{H}\mathbf{Q}\mathbf{R}^\top \|_2\\ =& \| \mathbf{H}\mathbf{H}^\top - \mathbf{H}\mathbf{A}^{-1/2}\mathbf{Q}(\mathbf{H}\mathbf{Q})^\top\mathbf{H}\mathbf{Q}\mathbf{Q}^\top\mathbf{A}^{-1/2}\mathbf{H}^\top \|_2\\ =& \| \mathbf{H}\mathbf{A}^{-\mathbf{1/2}}(\mathbf{A}^{\mathbf{1/2}}\mathbf{H}^\top - \mathbf{Q}\mathbf{Q}^\top\mathbf{H}^\top\mathbf{H}\mathbf{A}^{-1/2}\mathbf{H}^\top) \|_2\\ =& \| \mathbf{H}^\top\mathbf{H} - \mathbf{Q}\mathbf{Q}^\top\mathbf{H}^\top\mathbf{H} \|_2 = \| (\mathbf{I} - \mathbf{Q}\mathbf{Q}^\top)\mathbf{A} \|_2 .\end{aligned}$$

Using Theorem 2, we can bound the expected error,

$$\begin{aligned}&\kappa\mathbb{E} \| \mathbf{H}\mathbf{H}^\top - \mathbf{R}(\mathbf{H}\mathbf{Q})^\top\mathbf{H}\mathbf{Q}\mathbf{R}^\top \|_2\\ =&\kappa\mathbb{E} \| (\mathbf{I} - \mathbf{Q}\mathbf{Q}^\top)\mathbf{A} \|_2\\ \le&\zeta^{1/q}\sigma_{r+1}(\kappa\mathbf{A}) = \zeta^{1/q}\sigma_{r+1}(\kappa\mathbf{H}\mathbf{H}^\top)\\ \le&\zeta^{1/q}\sigma_{r+1}(\mathbf{X}\mathbf{X}^\top) + \zeta^{1/q} \| \mathbf{X}\mathbf{X}^\top - \kappa\mathbf{H}\mathbf{H}^\top \|_2,\end{aligned} \tag{7}$$

where the last inequality holds because of $\sigma_{r+1}(\kappa\mathbf{H}\mathbf{H}^\top) - \sigma_{r+1}(\mathbf{X}\mathbf{X}^\top) \le \max_i |\sigma_i(\mathbf{X}\mathbf{X}^\top) - \sigma_i(\kappa\mathbf{H}\mathbf{H}^\top)|$ and Lemma 1.

Combining Eq. (6) and (7), we conclude our result

$$\begin{aligned}&\mathbb{E} \| \mathbf{L} - \widetilde{\mathbf{L}} \|_2\\ \le& \zeta^{1/q}\sigma_{r+1}(\mathbf{X}\mathbf{X}^\top) + (1 + \zeta^{1/q}) \| \mathbf{X}\mathbf{X}^\top - \kappa\mathbf{H}\mathbf{H}^\top \|_2\\ \le& \zeta^{1/q} \| \mathbf{L} - \mathbf{L}_r \|_2 + (1 + \zeta^{1/q})\frac{n}{\sqrt{s}}\mathbf{L}^*_{i,i},\end{aligned}$$

where the last step is due to $\| \mathbf{X}\mathbf{X}^\top - \kappa\mathbf{H}\mathbf{H}^\top \|_2 \le \| \mathbf{X}\mathbf{X}^\top - \kappa\mathbf{H}\mathbf{H}^\top \|_F$ and Theorem 4. □

### A.3 Proof of Lemma 2

**Lemma 2.** *Given $\mathbf{X} \in \mathbb{R}^{d\times n}$, let $\mathbf{U}_\mathbf{R}$ be the orthonormal basis of the range of matrix $\mathbf{R} \in \mathbb{R}^{d\times s}$. Then for any $\mathbf{H}\mathbf{Q} \in \mathbb{R}^{s\times s}$,*

$$\begin{aligned}&\| \mathbf{X}\mathbf{X}^\top - \mathbf{X}\mathbf{U}_\mathbf{R}(\mathbf{X}\mathbf{U}_\mathbf{R})^\top \|_2\\ &\qquad \le \| \mathbf{X}\mathbf{X}^\top - \kappa\mathbf{R}(\mathbf{H}\mathbf{Q})^\top\mathbf{H}\mathbf{Q}\mathbf{R}^\top \|_2 .\end{aligned}$$

*where $\kappa = \frac{n}{s}$ is a non-zero scaling parameter.*

*Proof.* Let $\mathbf{P}_\mathbf{R} = \mathbf{U}_\mathbf{R}\mathbf{U}_\mathbf{R}^\top$. On using the property of orthogonal projector (i.e., $\mathbf{P}_\mathbf{R} = \mathbf{P}_\mathbf{R}^\top = \mathbf{P}_\mathbf{R}{}^2$), we have

$$\begin{aligned}&\| \mathbf{X}\mathbf{X}^\top - \mathbf{X}\mathbf{U}_\mathbf{R}(\mathbf{X}\mathbf{U}_\mathbf{R})^\top \|_2\\ =& \| \mathbf{X}\mathbf{X}^\top - \mathbf{X}\mathbf{P}_\mathbf{R}(\mathbf{X}\mathbf{P}_\mathbf{R})^\top \|_2 \qquad (8)\\ =& \| \mathbf{X} - \mathbf{P}_\mathbf{R}\mathbf{X} \|_2^2 = \max_{\|\mathbf{v}\|=1} \| \mathbf{v}^\top(\mathbf{X} - \mathbf{P}_\mathbf{R}\mathbf{X}) \|^2 .\end{aligned}$$

We then decompose the vector $\mathbf{v}$ as $\mathbf{v} = \alpha\mathbf{y} + \beta\mathbf{z}$, where $\mathbf{y} \in \mathrm{ran}(\mathbf{R}), \mathbf{z} \in \mathrm{ran}^\perp(\mathbf{R})$ and $\alpha^2 + \beta^2 = 1$. It is clear to see that $\mathbf{y}^\top\mathbf{P}_\mathbf{R} = \mathbf{y}^\top$, and $\mathbf{z}^\top\mathbf{P}_\mathbf{R} = 0$. Thereby,

$$\begin{aligned}&\| \mathbf{X} - \mathbf{P}_\mathbf{R}\mathbf{X} \|_2\\ \le& \max_{\mathbf{y}\in\mathrm{ran}(\mathbf{R}),\|\mathbf{y}\|=1} \| \mathbf{y}^\top(\mathbf{X} - \mathbf{P}_\mathbf{R}\mathbf{X}) \|\\ &\qquad + \max_{\mathbf{y}\in\mathrm{ran}^\perp(\mathbf{R}),\|\mathbf{z}\|=1} \| \mathbf{z}^\top(\mathbf{X} - \mathbf{P}_\mathbf{R}\mathbf{X}) \|\\ \le& \max_{\mathbf{z}\in\mathrm{ran}^\perp(\mathbf{R}),\|\mathbf{z}\|=1} \| \mathbf{z}^\top\mathbf{X} \| .\end{aligned} \tag{9}$$

For $\mathbf{z} \in \mathrm{ran}^\perp(\mathbf{R})$, $\mathbf{z}^\top\mathbf{R}(\mathbf{H}\mathbf{Q})^\top\mathbf{H}\mathbf{Q}\mathbf{R}^\top\mathbf{z} = 0$. Then,

$$\begin{aligned}\| \mathbf{z}^\top\mathbf{X} \|^2 &= \mathbf{z}^\top\mathbf{X}\mathbf{X}^\top\mathbf{z}\\ &= \mathbf{z}^\top(\mathbf{X}\mathbf{X}^\top - \kappa\mathbf{R}(\mathbf{H}\mathbf{Q})^\top\mathbf{H}\mathbf{Q}\mathbf{R}^\top)\mathbf{z}\\ &\le \max_{\|\mathbf{z}\|=1} \mathbf{z}^\top(\mathbf{X}\mathbf{X}^\top - \kappa\mathbf{R}(\mathbf{H}\mathbf{Q})^\top\mathbf{H}\mathbf{Q}\mathbf{R}^\top)\mathbf{z}\\ &= \| \mathbf{X}\mathbf{X}^\top - \kappa\mathbf{R}(\mathbf{H}\mathbf{Q})^\top\mathbf{H}\mathbf{Q}\mathbf{R}^\top \|_2 .\end{aligned} \tag{10}$$

Combining Eq. (8-10) concludes the lemma. □

## Appendix B Implementation Details

In this section, we present the details of our implementation in order for reproducibility. All experiments are conducted on the machines with Xeon 3175X CPU, 128G memory and RTX8000 GPU with 48 GB memory. The configurations and packages are listed below:

- Ubuntu 16.04
- CUDA 10.2
- Python 3.7
- Tensorflow 1.15.3
- Pytorch 1.10
- DGL 0.8.2
- NumPy 1.19.0 with MKL Intel

### B.1 EasyDGL Architectures for Three Tasks on Graph

#### *B.1.1 Dynamic Link Prediction:*

- Use maximum sequence length to 30 with the masked probability 0.2.
- Two-layer Attention-Intensity-Attention with two heads.
- Use *ReLU* as the activation.
- Use inner product between user embedding and item embedding as ranking score.

#### *B.1.2 Dynamic Node Classification*

- Randomly mask graph nodes with probability 0.2.
- Two-layer *GATConv* and one-layer Attention-Intensity-Attention block with two heads.
- Use *ReLU* as the activation.
- Use one-layer Linear for multi-class prediction.

#### *B.1.3 Traffic Forecasting*

- Randomly mask graph nodes with probability 0.2.
- Two-layer *SAGEConv* and one-layer Attention-Intensity-Attention with eight heads.
- Use *ReLU* as the activation.
- Use one-layer Linear for prediction.

### B.2 Baseline Architectures for Dynamic Link Prediction

**As mentioned, we follow IDCF [7] to build typical GNN architectures. Here we introduce the details for them.**

**GAT.** We use the *GATConv* layer available in DGL for implementation. The detailed architecture description is as below:

- A sequence of one-layer *GATConv* with four heads.
- Add self-loop and use batch normalization for graph convolution in each layer.

- Use *tanh* as the activation.
- Use inner product between user embedding and item embedding as ranking score.

**GraphSAGE.** We use the *SAGEConv* layer available in DGL for implementation. The detailed architecture description is as below:

- A sequence of two-layer *SAGEConv*.
- Add self-loop and use batch normalization for graph convolution in each layer.
- Use *ReLU* as the activation.
- Use inner product between user embedding and item embedding as ranking score.

**GCN.** We use the *SGConv* layer available in DGL for implementation. The detailed architecture description is as below:

- One-layer *SGConv* with two hops.
- Add self-loop and use batch normalization for graph convolution in each layer.
- Use *ReLU* as the activation.
- Use inner product between user embedding and item embedding as ranking score.

**ChebyNet.** We use the *ChebConv* layer available in DGL for implementation. The detailed architecture description is as below:

- One-layer *ChebConv* with two hops.
- Add self-loop and use batch normalization for graph convolution in each layer.
- Use *ReLU* as the activation.
- Use inner product between user embedding and item embedding as ranking score.

**ARMA.** We use the *ARMAConv* layer available in DGL for implementation. The detailed architecture description is as below:

- One-layer *ARMAConv* with two hops.
- Add self-loop and use batch normalization for graph convolution in each layer.
- Use *tanh* as the activation.
- Use inner product between user embedding and item embedding as ranking score.

**We also summarize the implementation details of the compared sequential and temporal baselines as follows.**

**GRU4REC.**[1] We use the software provided by the authors for experiments. The detailed architecture description is as below:

- A sequence of two GRU cells.
- Use maximum sequence length to 30.
- Use inner product between user embedding and item embedding as ranking score.

**SASREC.**[2] We use the software provided by the authors for experiments. The detailed architecture description is as below:

- A sequence of two-block Transformer with four heads on Koubei, eight heads on Tmall and one head on Netflix.
- Use maximum sequence length to 30.
- Use inner product between user embedding and item embedding as ranking score.

**GREC.**[3] We use the software provided by the authors for experiments. The detailed architecture description is as below:

- A sequence of six-layer dilated CNN with degree $1, 2, 2, 4, 4, 8$.
- Use maximum sequence length to 30 with the masked probability $0.2$.
- Use inner product between user embedding and item embedding as ranking score.

**S2PNM.**[4] We use the software provided by the authors for experiments. The detailed architecture description is as below:

- A sequence of one-block GRU-Transformer.
- Use maximum sequence length to 30.
- Use inner product between user embedding and item embedding as ranking score.

**BERT4REC.**[5] We use the software provided by the authors for experiments. The detailed architecture description is as below:

- A sequence of three-block Transformer with eight heads.
- Use maximum sequence length to 30 with the masked probability $0.2$.
- Use inner product between user embedding and item embedding as ranking score.

**DyREP.**[6] We use the software provided by the third party for experiments. The detailed architecture description is as below:

- A sequence of one Attention-RNN Layer.
- Use maximum sequence length to 30.
- Use linear layer of user embedding and item embedding with softplus activation as ranking score.

**TGAT.**[7] We use the software provided by the authors for experiments. The detailed architecture description is as below:

- A sequence of three-block Transformer with time sinusoidal embeddings.
- Use maximum sequence length to 30.
- Use inner product between user embedding and item embedding as ranking score.

**TiSASREC.**[8] We use the software provided by the authors for experiments. The detailed architecture description is as below:

- A sequence of three-block Transformer with one heads with time embedding.
- Use maximum sequence length to 30.
- Use inner product between user embedding and item embedding as ranking score.

**TGREC.**[9] We use the software provided by the authors for experiments. The detailed architecture description is as below:

- A sequence of three-block Transformer with time sinusoidal embeddings.
- Use maximum sequence length to 30.

1. https://github.com/hidasib/GRU4Rec
2. https://github.com/kang205/SASRec
3. https://github.com/fajieyuan/WWW2020-grec
4. https://github.com/cchao0116/S2PNM-TKDE2022
5. https://github.com/FeiSun/BERT4Rec
6. https://github.com/uoguelph-mlrg/LDG
7. https://github.com/StatsDLMathsRecomSys/Inductive-representation-learning-on-temporal-graphs
8. https://github.com/JiachengLi1995/TiSASRec
9. https://github.com/DyGRec/TGSRec

- Use inner product between user embedding and item embedding as ranking score.

**TimelyREC.**[10] We use the software provided by the authors for experiments. The detailed architecture description is as below:

- A sequence of one-block Attention-Attention.
- Use maximum sequence length to 30.
- Use inner product between user embedding and item embedding as ranking score.

**CTSMA.**[11] We use the software provided by the authors for experiments. The detailed architecture description is as below:

- A sequence of two-block Transformer with four heads.
- Use maximum sequence length to 30.
- Use inner product between user embedding and item embedding as ranking score.

## B.3 Baseline Architectures for Dynamic Node Classification

The configurations for static graph models are identical to the dynamic link prediction task except the decoder module that is replaced by one-layer *Linear*.

**In the following, we present the architectures for new dynamic graph models.**

**DySAT.**[12] We use the software provided by the authors for experiments. The detailed architecture description is as below:

- Use past five graph snapshots as input.
- A sequence of three-layer *GATConv* and one-layer TemporalAttention with two heads.
- Use *ReLU* as the activation.
- Use one-layer Linear for multi-class prediction.

**EvolveGCN.**[13] We use the software provided by the authors for experiments. The detailed architecture description is as below:

- Use past five graph snapshots as input.
- One-layer gated GRUCell to update the parameters of two-layer *GCNConv*.
- Use *ReLU* as the activation.
- Use one-layer Linear for multi-class prediction.

**JODIE.**[14] We use the software provided by the authors for experiments. The detailed architecture description is as below:

- Use past five graph snapshots as input.
- One-layer gated GRUCell to update the hidden states read out from two-layer TemporalTransformer.
- Use *ReLU* as the activation.
- Use one-layer Linear for multi-class prediction.

**TGN.**[15] We use the software provided by the authors for experiments. The detailed architecture description is as below:

- Use past five graph snapshots as input with sinusoidal time embedding.
- Three-layer TemporalTransformer with five heads.
- One-layer time-decayed Recurrent unit to sequentially update node embeddings over time.
- Use *ReLU* as the activation.
- Use one-layer Linear for multi-class prediction.

## B.4 Baseline Architectures for Traffic Forecasting

The configurations for static graph models are identical to the dynamic link prediction task except the decoder module that is replaced by one-layer *Linear*.

**In the following, we present the architectures for new dynamic graph models.**

**DCRNN.** We use the software provided in DGL library. The detailed architecture description is as below:

- Use past twelve graph snapshots as input.
- One-layer *ChebConv* with two hops.
- One-layer *GraphRNN* to sequential update the node embeddings over time.
- Use *ReLU* as the activation.
- Use one-layer Linear for prediction.

**GaAN.** We use the software provided in DGL library. The detailed architecture description is as below:

- Use past twelve graph snapshots as input.
- One-layer gated graph attention that considers the edge weights.
- One-layer *GraphRNN* to sequential update the node embeddings over time.
- Use *ReLU* as the activation.
- Use one-layer Linear for prediction.

**STGCN.** We use the software provided in DGL library. The detailed architecture description is as below:

- Use past twelve graph snapshots as input.
- A sequence of TNTSTNTST, where T, N, S represents temporal convolutional neural network, *LayerNorm* and spatial graph convolutional neural network, respectively.
- Use *ReLU* as the activation.
- Use one-layer Linear for prediction.

**DSTAGNN.**[16] We use the software provided by the authors for experiments. The detailed architecture description is as below:

- Use past twelve graph snapshots as input.
- Four-layer DSTAGNN with four heads each of which uses *ChebConv* with three hops.
- Use *ReLU* as the activation.
- Use *Conv2d* and *Linear* for prediction.

## B.5 Choice of Hyper-parameters

Regarding the choice of optimizer, we use Adam [3] if not specified and the number of epochs is 200. We search by grid the embedding size ranging in $\{64, 128, \ldots, 512\}$, learning rate $\{1e–5, 1e–4, \ldots, 1e–1\}$, dropout rate $\{0.1, 0.2, \ldots, 0.7\}$, batch size $\{64, 128, \ldots, 512\}$ and $\ell_2$ regularizer $\{1e-5, 1e-4, \ldots, 1e–1\}$. We also study the influence of the neighborhood hops ranging from 1 to 3, the number of graph filtering blocks from 1 up to 4 and the number of heads in $\{1, 2, \ldots, 8\}$.

We warn that we set the embedding size to 32 for fair comparisons when evaluating the performance on the Ellicit

10. https://github.com/Junsu-Cho/TimelyRec
11. https://github.com/cchao0116/CTSMA-ICML21
12. https://github.com/aravindsankar28/DySAT
13. https://github.com/IBM/EvolveGCN
14. https://github.com/claws-lab/jodie
15. https://github.com/twitter-research/tgn
16. https://github.com/SYLan2019/DSTAGNN

and META-LA datasets. This is because that TGN, DCRNN and DSTAGNN take days to complete the training if the embedding size is greater than 64.

With regard to EasyDGL, we search by grid the masking rate in $\{10\%, 20\%, \dots, 50\%\}$ where in majority of cases $20\%$ outputs the best results. We also search the best parameter for the TPPLE term in $\{1e-7, 1e-6, \dots, 1e-3\}$.

## Appendix C Dataset Processing

We present more dataset details in this section.

### C.1 Time Scaling

We use one hour, one day and one week to scale the time data on the Netflix, Tmall and Koubei datasets, respectively. The choice of time unit is determined by the averaged time between two consecutive events for each user. We warn that again the time is discrete on the Elliptic data and the traffic speed readings on META-LA are record every five minutes. For both of these two datasets, we apply no modifications to the time data.

### C.2 Random Seed

We use five random seeds to yield different data splits, i.e., 12345, 54321, 56789, 98765 and 7401.

### C.3 Normalization on the META-LA data

We calculate the mean and the standard deviation of the training readings on each road (node). When making predictions, we use these quantities to scale down the input readings and scale up the output readings. We found by experiments that this treatment can significantly reduce the RMSE and MAPE errors.

## References

[1] P. Drineas, M. W. Mahoney, and N. Cristianini. On the nyström method for approximating a gram matrix for improved kernel-based learning. *Journal of Machine Learning Research*, 6(12), 2005.

[2] N. Halko, P.-G. Martinsson, and J. A. Tropp. Finding structure with randomness: Probabilistic algorithms for constructing approximate matrix decompositions. *SIAM review*, 53(2):217–288, 2011.

[3] D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. In *Proceedings of the International Conference on Learning Representations (ICLR '15)*, 2015.

[4] S. Kumar, M. Mohri, and A. Talwalkar. Sampling methods for the nyström method. *Journal of Machine Learning Research*, 13(1):981–1006, 2012.

[5] M. Li, J. T.-Y. Kwok, and B. Lü. Making large-scale nyström approximation possible. In *Proceedings of the International Conference on Machine Learning (ICML '10)*, page 631, 2010.

[6] G. W. Stewart. *Matrix perturbation theory*. Boston: Academic Press, 1990.

[7] Q. Wu, H. Zhang, X. Gao, J. Yan, and H. Zha. Towards open-world recommendation: An inductive model-based collaborative filtering approach. In *Proceedings of the International Conference on Machine Learning (ICML '21)*, pages 11329–11339, 2021.