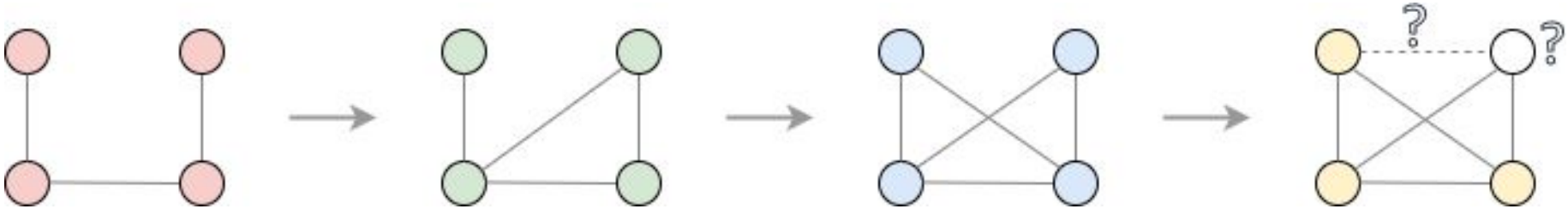
Beyond Spatio-Temporal Representations: Evolving Fourier Transform for Temporal Graphs

Anson Bastos, Kuldeep Singh, Abhishek Nadgeri, Manish Singh, Toyotaro Suzumura

ansonbastos@gmail.com
HERE Technologies, India
IIT Hyderabad, India

Dynamic Graph Representation Learning

- Consider the sequence of Graphs $\{G_t\}$ with N_t nodes and with E_t edges that evolves dynamically with time, then the aim is to learn some vector representation of G_t (graph at time t) capturing the historical interaction till t.



- Some of the application of Dynamic Graph Representation learning are:
  - **Link Prediction**: understand how pairs of nodes are connected to the graph.
  - **Edge Classification:** classify the evolving edges in the graph.
  - **Node Classification:** predict properties of nodes in the graph.

Challenges

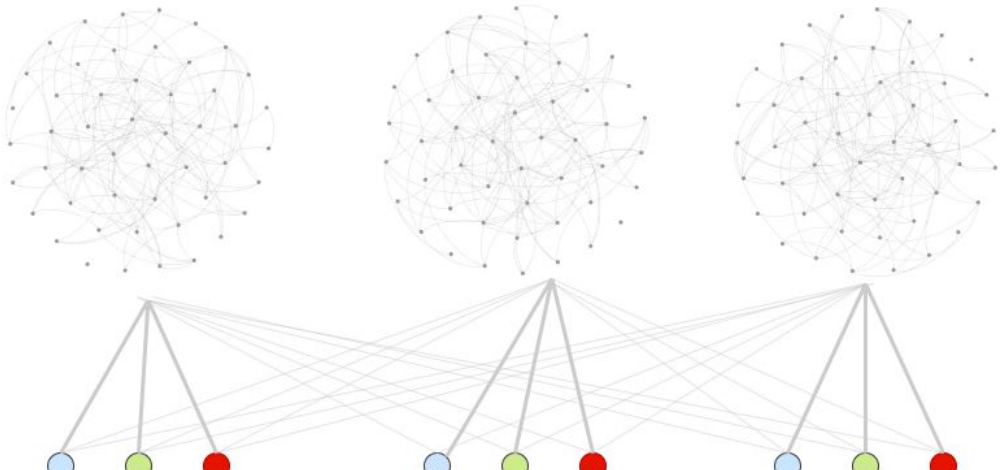
- Tremendous progress in the domain of Graph Representation Learning for static graphs.
- However, many real world graphs are dynamic in nature such as social network graphs, citation graphs, transaction networks, e-commerce preference graphs, epidemiological transmission graphs etc.
- It has been shown that the static graph methods do not work well in this setting of dynamic graphs
- It remains an open research question to suitably model the representation learning problem for dynamic graphs
    - Many attempts aim to capture the time varying properties using RNNs ahead of static Graph Neural Networks (GNNs)
    - Researchers have also tried using a autoencoders to learn the evolving features in an unsupervised manner
    - Recently researchers argue that learning the parameters of GNNs dynamically helps to effectively capture the evolving graph structure

Existing works and limitations

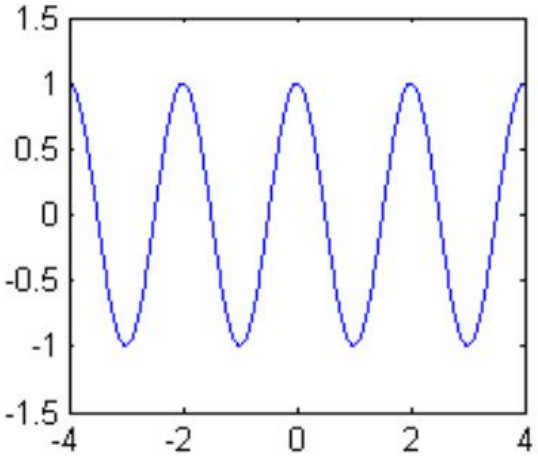
- Some major works in this direction are
  - DynGEM (Goyal et.al., IJCAI 2017)
    - Proposes to use auto encoders for learning over timesteps
  - dyngraph2vec (Goyal et.al., Knowledge-Based Systems 2019)
    - Autoencoder uses an RNN to incorporate past node information
  - EvolveGCN (Pareja et.al., AAAI 2018)
    - Learns the GCN parameters to evolve over time using an RNN
  - GAEN (Shi et.al., IJCAI 2021)
    - Learns parameters of a GAT with an RNN that focus on graph topology discrepancies
- The existing methods learn spatial features by **local neighborhood aggregation**, which essentially **only captures the low pass signals** and local interactions
- These works **ignore the global interactions** that may emerge due to dynamic nature of the graph

Motivation

- We argue that ***capturing global dependencies***, beyond local neighborhood aggregation will enhance representation learning in dynamic graphs
- For this we view the problem through the lens of spectral graph theory to ***learn representations after transformation in the dynamic graph*** **spectral space** to capture global interactions of the dynamic graph
    - For this we need to propose a novel Spectral Transform for Evolving Graph



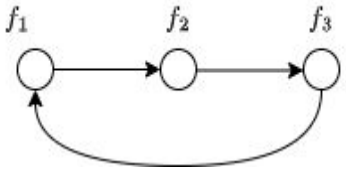
Classical Fourier Transform



$$F(\omega) = \int_{-\infty}^{\infty} f(t) \exp(-j\omega t) dt.$$

Source: Bentley, Paul M., and J. T. E. McDonnell. "Wavelet transforms: an introduction." *Electronics & communication engineering journal* 6.4 (1994): 175-186.

Graph Fourier Transform



0	1	1
1	0	1
1	1	0

$$\nabla f_2^- = f_2 - f_1 \qquad \nabla f_2^+ = f_3 - f_2$$

$$\nabla^2 f_2 = \langle \nabla, \nabla f_1 \rangle$$

$$\nabla^2 f_2 = \nabla f_2^+ - \nabla f_2^-$$

2	-1	-1
-1	2	-1
-1	-1	2

$$-\nabla^2 f_2 = 2f_2 - f_1 - f_3 \longrightarrow L = D - A$$

The laplacian of the ring graph is a circulant matrix.

The eigenvalues of L turn out to be the frequencies and the eigenvectors turn out to be the basis of the Fourier transform!
Generalizing this to the graph setting gives us the spectra and basis of the graph Fourier transform

Graph Fourier Transform

$$\mathcal{L}\chi_\ell = \lambda_\ell \chi_\ell$$

$$\hat{f}(\ell) = \langle \chi_\ell, f \rangle = \sum_{n=1}^{N} \chi_\ell^*(n) f(n) \longrightarrow \text{GFT}$$

$$f(n) = \sum_{\ell=0}^{N-1} \hat{f}(\ell) \chi_\ell(n) \longrightarrow \text{IGFT}$$

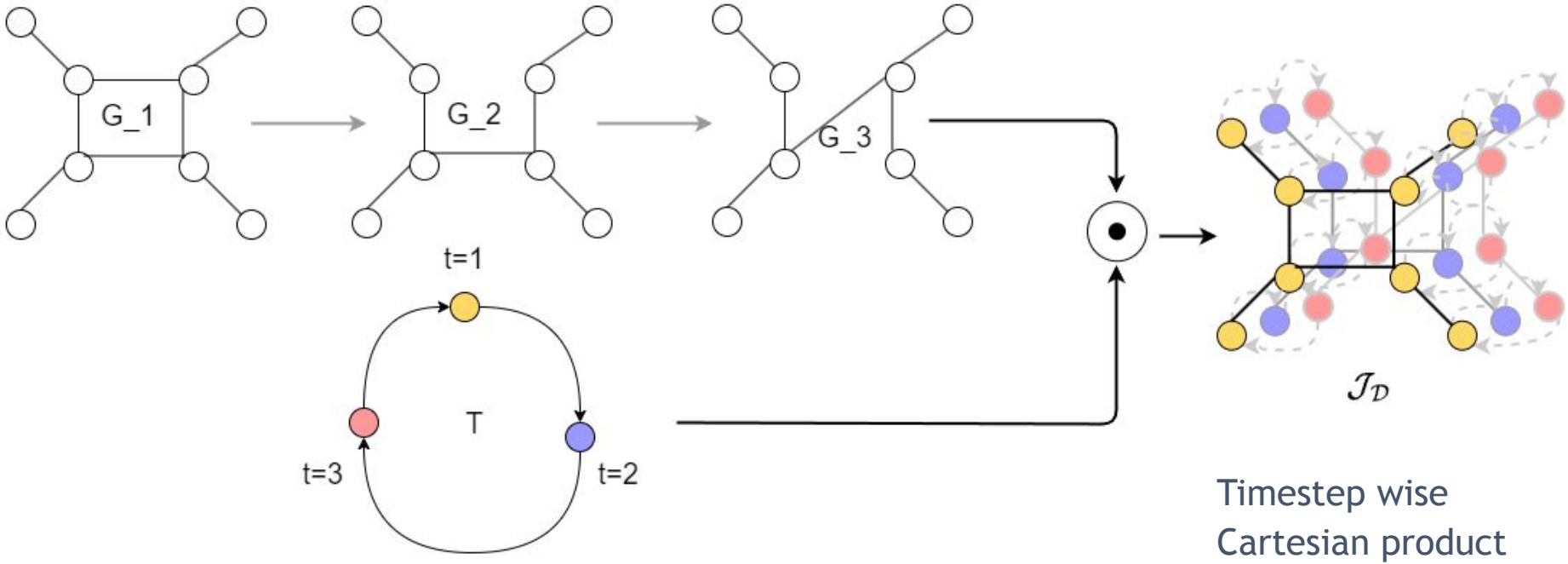
The eigenvalues of L turn out to be the frequencies and the eigenvectors turn out to be the basis of the Fourier transform!

Generalizing this to the graph setting gives us the spectra and basis of the graph Fourier transform

Evolving Graph Fourier Transform (EFT) ?

- Motivation: To make spectral GNN work for temporal graphs effectively and efficiently, there is a necessity for an invertible transform that collectively captures evolving spectra along the graph vertex and time domain
- GFT exists for static graphs but naive extension of GFT to dynamic graphs would lose the distinction between variation along temporal and vertex domains and incur an added computational cost by a multiplicative factor of $O(T^3)$
- Thus EFT aims to solve the following challenges over GFT:
  - EFT is computationally efficient compared to the direct eigendecomposition of the joint Laplacian.
  - Distinction between time and vertex domain frequency components with EFT provides interpretability to the transformed spectral domain.
- Research question:
  - *How to design a collective time-vertex transform over dynamic graphs?*
  - *What is the effectiveness of the collective time-vertex spectral filtering over dynamic graphs?*

Sketch of EFT Design

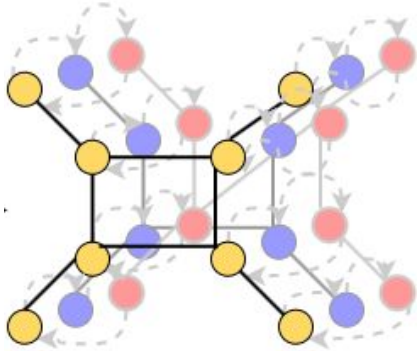


Timestep wise Cartesian product of the graphs

Sketch of EFT Design

Lemma 1. *(Variational Characterization of $\mathcal{J}_\mathcal{D}$) The 2-Dirichlet $S_2(X)$ of the signals X on $\mathcal{J}_\mathcal{D}$ is the quadratic form of the Laplacian $L_{\mathcal{J}_\mathcal{D}}$ of $\mathcal{J}_\mathcal{D}$ i.e.*

$$S_2(X) = \int_{i=0}^{NT} vec(X)(i) \int_{j=0}^{NT} L_{\mathcal{J}_\mathcal{D}}(i,j) vec(X)(j) didj = vec(X)^T L_{\mathcal{J}_\mathcal{D}} vec(X) \qquad (3)$$



$\mathcal{J}_\mathcal{D}$

- This implies that $L_{\mathcal{J}_\mathcal{D}} \succeq 0$ since $S_2(X) \geq 0$
- Thus we can be assured of the existence of the eigenvalue decomposition

Sketch of EFT Design - Optimization

$$f_{\max} = \max_{x, \|x\| \leq 1} \int_{i=0}^{NT} x(i) \int_{j=0}^{NT} L_{\mathcal{J}_{\mathcal{D}}}(i,j)x(j)didj = \max_{x, \|x\| \leq 1} \quad x^T L_{\mathcal{J}_{\mathcal{D}}}(i,j)x$$

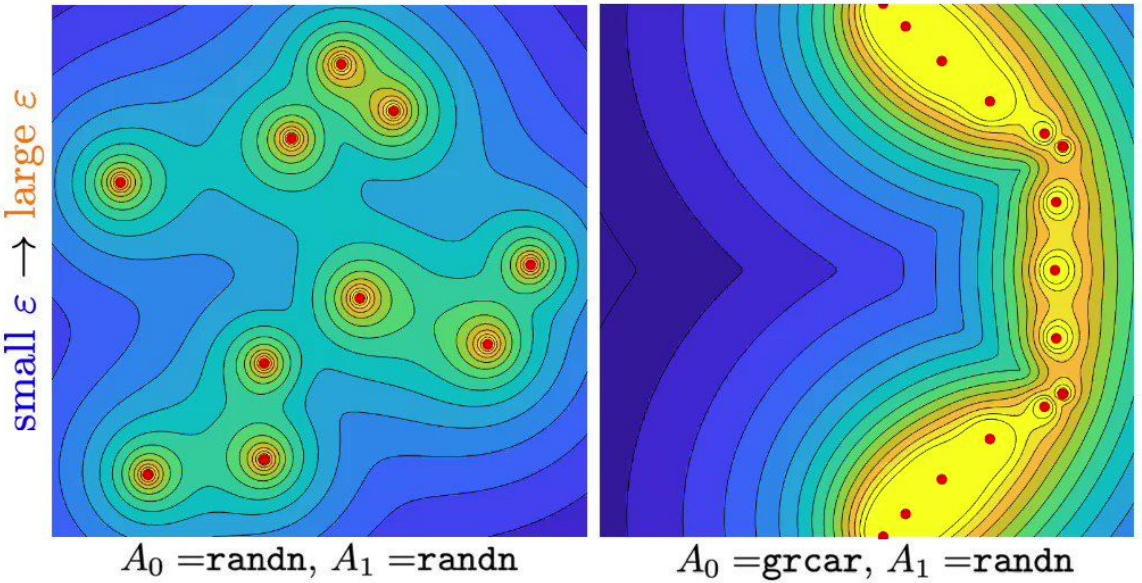
- The optimal solution x provides the basis for transforming a dynamic graph signal to obtain its maximum frequency component, denoted by fmax. We can obtain the next frequency values by optimizing equation 4 in orthogonal directions.
- However, this approach has an issue - the eigenvalue decomposition would have to be performed over a large number of nodes. In a real world setting of temporal graphs with T timesteps, this method would have a complexity of $O((NT)^3)$, which would be prohibitive considering large number of timesteps
- Thus, we relax the objective in above equation to include solutions in the pseudospectrum.

Sketch of EFT Design - Pseudospectrum

Pseudo-spectrum of $A \in \mathbb{C}^{n\times n}$:

$$\sigma_\varepsilon(A) \overset{\text{def.}}{=} \left\{ z \in \mathbb{C} \; ; \; \|(A - z\mathrm{Id})^{-1}\| > 1/\varepsilon \right\}$$

Animation of $\sigma_\varepsilon((1-t)A_0 + tA_1)$ for $0 \leqslant t \leqslant 1$:



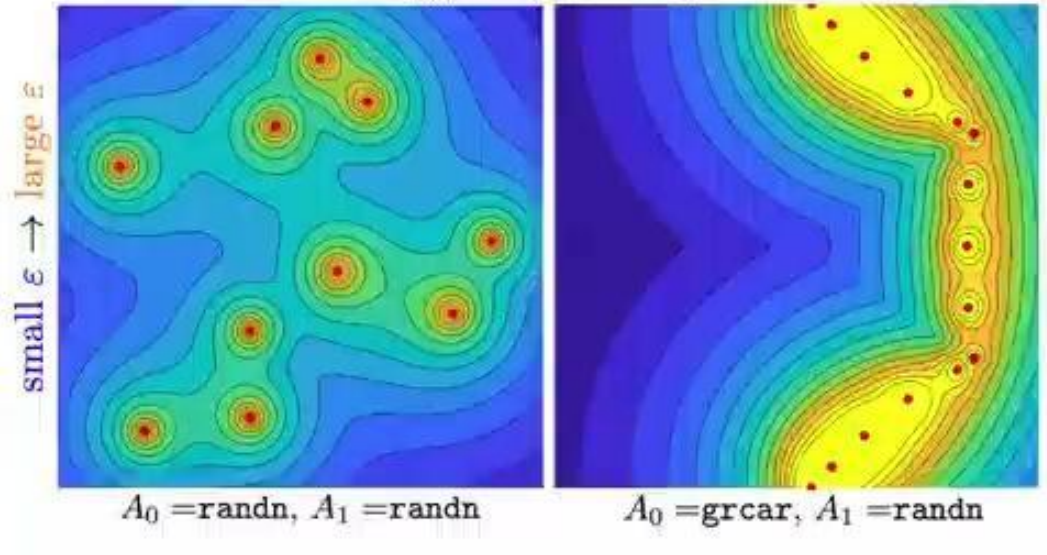
Source:https://twitter.com/gabrielpeyre/status/1694575259256360151

Sketch of EFT Design - Pseudospectrum

Pseudo-spectrum of $A \in \mathbb{C}^{n\times n}$:

$$\sigma_\varepsilon(A) \stackrel{\text{def.}}{=} \{z \in \mathbb{C}\,;\ \|(A - z\mathrm{Id})^{-1}\| > 1/\varepsilon\}$$

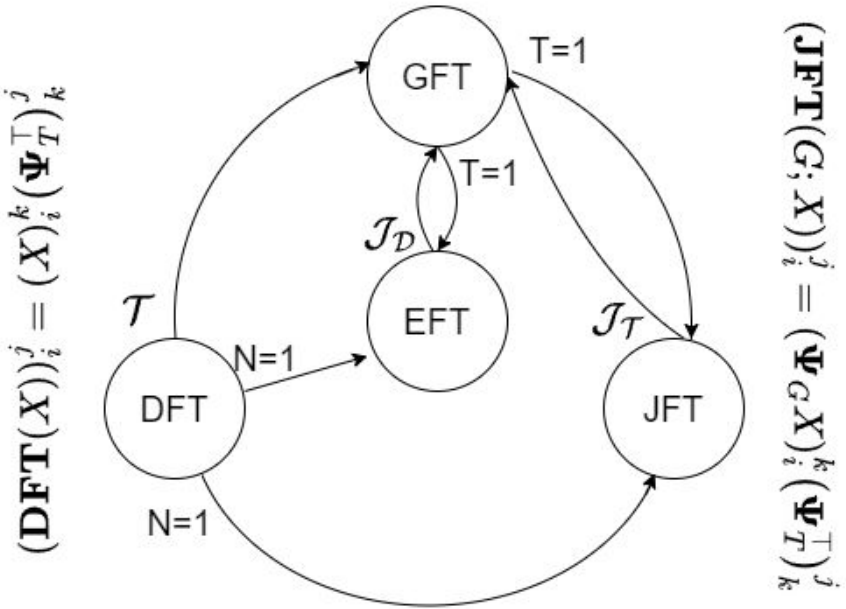
Animation of $\sigma_\varepsilon((1-t)A_0 + tA_1)$ for $0 \leqslant t \leqslant 1$:



Source:https://twitter.com/gabrielpeyre/status/1694575259256360151

Sketch of EFT Design

$$(\mathbf{GFT}(G;X))_i^j = (\mathbf{\Psi}_G X)_i^j$$



$$(\mathbf{DFT}(X))_i^j = (X)_i^k (\mathbf{\Psi}_T^\top)_k^j$$

$$(\mathbf{JFT}(G;X))_i^j = (\mathbf{\Psi}_G X)_i^k (\mathbf{\Psi}_T^\top)_k^j$$

$$(\mathbf{EFT}(\{G_t\};X))_i^j = (\mathbf{\Psi}_{G_t} X)_i^{kk} (\mathbf{\Psi}_T^\top)_k^j$$

- Figure shows equivalence between EFT and existing transformations (DFT, JFT, GFT).
- Each directed arrow (e.g, A to B), interprets as a transform simulation (transform A can be simulated by B using edge annotations)

EFT Properties

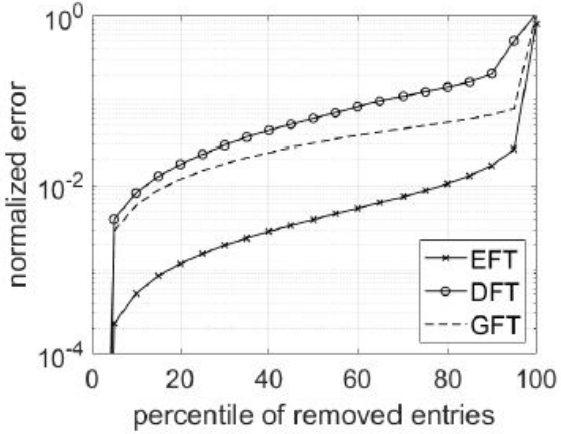
Property 1. *(Equivalence in special case) Consider* Ψ_T *to be the time Fourier transform and* Ψ_{G_t} *to be the Graph Fourier transform at time t. Let* $\Psi_{\mathcal{J}_\mathcal{D}}$ *be the Graph fourier transform of* $\mathcal{J}_\mathcal{D}$*. In the special case of* $G_{t_i} = G_{t_j} \forall i, j \in \{T\}$ *we have* $(\Psi_{\mathcal{J}_\mathcal{D}})_i^j = (\Psi_D)_i^j = (\Psi_T \otimes \{\Psi_{G_t}\})_i^{j\left\lfloor \frac{j}{N} \right\rfloor}$.

Property 2. *EFT is an invertible transform and the inverse is given by* $\mathbf{EFT}^{-1}(\hat{X})_i^j = \left(\Psi_G^{-1}\hat{X}\right)_i^{kk} \left(\Psi_T^{\top *}\right)_k^j$ *in matrix form and* $\mathbf{EFT}^{-1}(\hat{x})_{j*N+i} = \left(\Psi_T^* \otimes \Psi_G^{-1}\right)_{j*N+i}^{k\left\lfloor \frac{k}{N} \right\rfloor} \hat{x}_k$ *in vector form.*

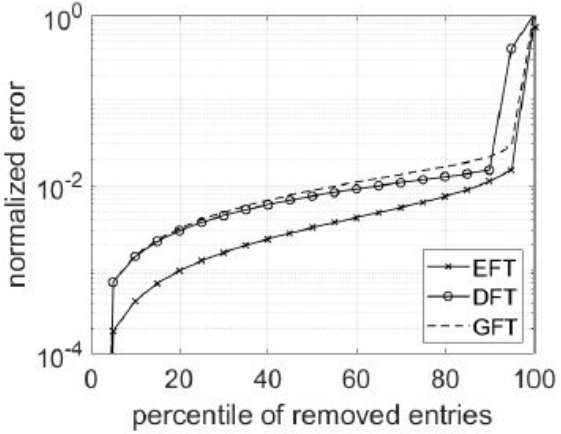
Property 3. *EFT is a unitary transform if and only if GFT is unitary at all timesteps considered i.e.* $\Psi_D \Psi_D^* = I_{NT}$ *iff* $\Psi_{G_t} \Psi_{G_t}^* = I_N, \forall t$.

Property 4. *EFT is invariant to the order of application of DFT or GFT on signal X.*

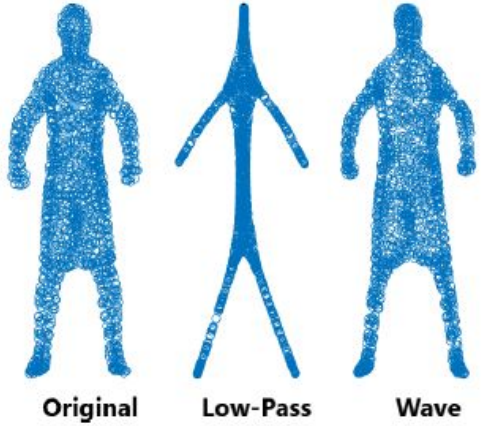
EFT Representations



(a) *Dog*



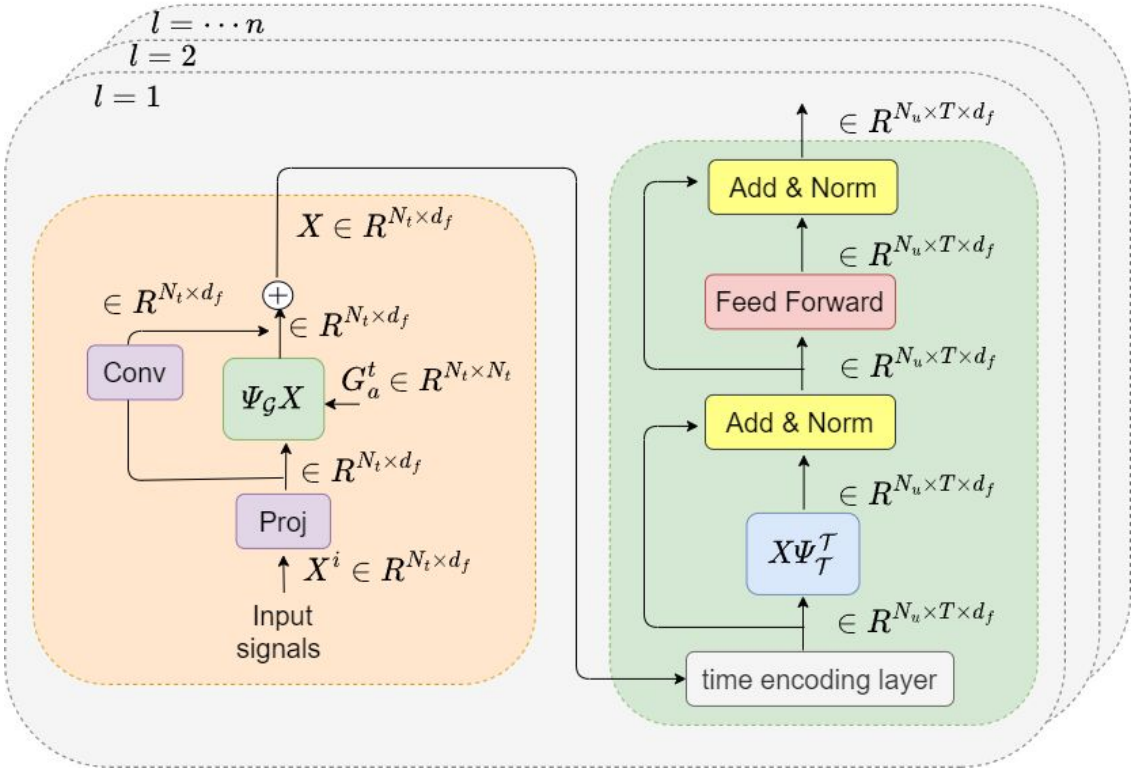
(b) *Dancer*



(c) *Dancer*

Representations on dynamic mesh datasets. Left (a,b): Reconstruction error on the datasets illustrating the compactness of EFT . Right (c): Illustration of filtering using EFT on the dynamic mesh of a Dancer.

Inducing EFT into a Neural Architecture



Since our idea is to perform collective filtering along the vertex and temporal domain in EFT, we need two modules :

- Ψ_{Gt}(vertex aspect)
- Ψ_{T}(temporal aspect)

Dataset Statistics

SR Datasets	Beauty	Games	CDs
# of Users	52,024	31,013	17,052
# of Items	57,289	23,715	35,118
# of Interactions	394,908	287,107	472,265
Average length	7.6	9.3	27.6
Density	0.01%	0.04%	0.08%

	# Nodes	# Edges	# Time Steps (Train / Val / Test)	Task
SBM	1,000	4,870,863	35 / 5 / 10	LP
UCI	1,899	59,835	62 / 9 / 17	LP
AS	6,474	13,895	70 / 10 / 20	LP
Elliptic	203,769	234,355	31 / 5 / 13	NC
Brain	5,000	1,955,488	10 / 1 / 1	NC

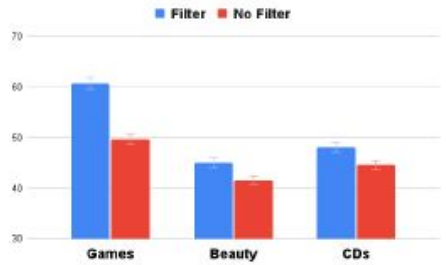
Results - Large SR Datasets

	GRU4Rec+	Caser	SASRec	HGN	TiSASRec	FMLPRec	SRGNN	HyperRec	DGSR	EFT-T
Recall@10										
Beauty	43.98	42.64	48.54	48.63	46.87	47.47	48.62	34.71	<u>52.40</u>	**53.23**
Games	67.15	68.83	73.98	71.42	71.85	73.62	73.49	71.24	<u>75.57</u>	**77.78**
CDs	67.84	61.65	71.32	71.42	71.00	72.41	69.63	71.02	<u>72.43</u>	**75.42**
NDCG@10										
Beauty	26.42	25.47	32.19	32.47	30.45	32.38	32.33	23.26	<u>35.90</u>	**37.10**
Games	45.64	45.93	53.60	49.34	50.19	51.26	53.35	48.96	<u>55.70</u>	**58.65**
CDs	44.52	45.85	49.23	49.34	48.97	53.31	48.95	47.16	<u>51.22</u>	**54.99**

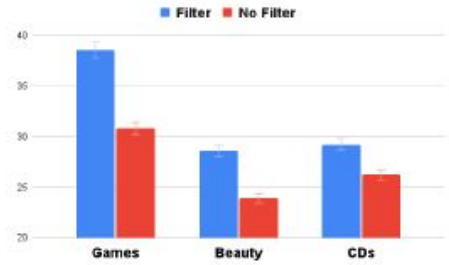
Results - Benchmark Dynamic Graph Datasets

Datasets	SBM		UCI		AS		Ell	Brn
Metrics	MAP	MRR	MAP	MRR	MAP	MRR	F1	F1
GCN	0.189	0.014	0.000	0.047	0.002	0.181	0.434	0.232
GAT	0.175	0.013	0.000	0.047	0.020	0.139	0.451	0.121
DynGEM	0.168	0.014	0.021	0.106	0.053	0.103	0.502	0.225
GCN-GRU	0.190	0.012	0.011	0.098	0.071	0.339	0.575	0.186
dg2vec v1	0.098	0.008	0.004	0.054	0.033	0.070	0.464	0.191
dg2vec v2	0.159	0.012	0.020	0.071	0.071	0.049	0.442	0.215
GAEN	0.1828	0.008	0.000	0.049	0.130	0.051	0.492	0.205
EGCN-H	0.195	0.014	0.013	0.090	0.153	0.363	0.391	0.225
EGCN-O	0.200	0.014	0.027	0.138	0.114	0.275	0.544	0.192
LED-GCN	0.196	0.015	0.032	0.163	0.193	0.469	0.471	0.261
LED-GAT	0.182	0.012	0.026	0.149	0.233	0.384	0.503	0.150
EFT-T	**0.250**	**0.024**	**0.055**	**0.181**	**0.672**	**0.689**	**0.616**	**0.308**

Effectiveness of joint filtering in the presence of semantic noise



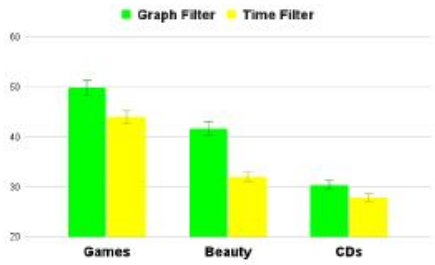
(a) *Recall*@10



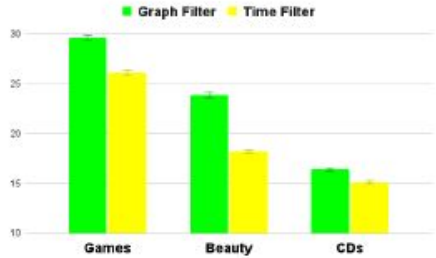
(b) *NDCG*@10

- We induce semantic noise into the system by adding a random vector to the node embeddings.
- The vector is sampled from a normal distribution with zero mean and variance equal to that of the node embeddings.
- Then, we run experiments on our model with learnable joint graph-time filters and without the filters.
- In the presence of noise, the performance of configuration with filters is much better than that without any filtering.
- This confirms that filtering helps with being robust to semantic noise

Effectiveness of filtering in the presence of structural noise



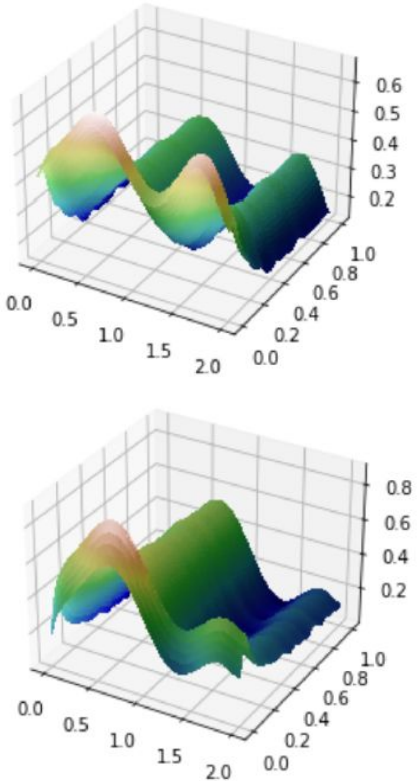
(a) *Recall*@10



(b) *NDCG*@10

- We test the hypothesis is that graph filters should be more robust to graph structure perturbations than time filters.
- Here, we induce structural noise into the system by adding random item nodes to a user node. This simulates the scenario in sequential recommendation where a user randomly clicks a few items unrelated to the primary intention.
- The embeddings of the noise node are initialized randomly from the same distribution as the item node embeddings. We randomly add 1-4 nodes per user sequence.
- We observe that on inducing structural noise, the performance of the configuration with graph filters is statistically better ($p < 0.01$ using a paired t-test) compared to the one with only time filters.
- The result confirms the hypothesis that graph filtering is more robust to structural noise, affirming the need for graph-based methods to ensure robustness to such perturbations in sequential recommendation.

Visualization of learnt filters



(c) *CDs*

- The x-axis shows the graph frequency range from 0-2 since we use the normalized laplacian. The y-axis represents the temporal frequency normalized by the Nyquist's frequency. The z-axis shows the magnitudes of the normalized frequency response.
- We observe that the response magnitude is predominant in the low-frequency region, indicating the datasets' nature.
- There are also a few modes of frequency distribution in the middle and higher-frequency (i.e., long-range interactions) regions.
- Illustrated behavior concludes a key finding: aggregation happens in Tracer from distant users besides local neighborhoods.

Summary and Conclusion

- We propose a novel transform to convert temporal graphs into the frequency domain
  - In order to do so, we introduce pseudospectrum relaxations to the variational objective obtaining a simplified transformation
  - EFT provides interpretability in the sense of decomposing the transformed spectral space into time and vertex domains
  - EFT is computationally efficient for real-world applications.
- We further demonstrate the practical effectiveness for temporal graphs.
- Our work opens up new possibilities for dynamic graph analysis and representation learning, and we encourage researchers to explore potential of EFT as a spectral representation of the evolving graph in downstream graph representation learning models.

Thank you

Questions? Please write to us

ansonbastos@gmail.com,
kuldeep.singh1@cerence.com ,
abhishek.nadgeri@rwth-aachen.de