Learnable Spectral Wavelets on Dynamic Graphs to Capture Global Interactions

Anson Bastos[1], **Abhishek Nadgeri**[2], **Kuldeep Singh**[3], **Toyotaro Suzumura**[4], **Manish Singh**[1]
[1] Indian Institute of Technology Hyderabad, India
[2] RWTH Aachen, Germany
[3] Zerotha Research and Cerence Gmbh, Germany
[4] The University of Tokyo, Japan
cs20resch11002@iith.ac.in, abhishek.nadgeri@rwth-aachen.de, kuldeep.singh1@cerence.com, suzumura@acm.org, msingh@cse.iith.ac.in

Abstract

Learning on evolving(dynamic) graphs has caught the attention of researchers as static methods exhibit limited performance in this setting. The existing methods for dynamic graphs learn spatial features by local neighborhood aggregation, which essentially only captures the low pass signals and local interactions. In this work, we go beyond current approaches to incorporate global features for effectively learning representations of a dynamically evolving graph. We propose to do so by capturing the spectrum of the dynamic graph. Since static methods to learn the graph spectrum would not consider the history of the evolution of the spectrum as the graph evolves with time, we propose a novel approach to learn the graph wavelets to capture this evolving spectra. Further, we propose a framework that integrates the dynamically captured spectra in the form of these learnable wavelets into spatial features for incorporating local and global interactions. Experiments on eight standard datasets show that our method significantly outperforms related methods on various tasks for dynamic graphs.

Introduction

Recently there has been tremendous progress in the domain of Graph Representation Learning (Khoshraftar and An 2022). The aim here is to develop novel methods to learn the features of graphs in a vector space. Such approaches have successful applications in the domain of image recognition (Han et al. 2022), computational chemistry (Ying et al. 2021), drug discovery (Stärk et al. 2022) and Natural Language processing (Wu et al. 2021a). Although effective, the underlying graphs are static in nature.

In many real-world scenarios, graphs are dynamic, for example, social networks, citation graphs, bank transactions, etc. For such cases, various approaches have been developed (see survey by Kazemi et al. (2020)). Broadly, these methods aim to learn the evolving nature of graphs through spatial features relying on local neighborhood aggregation (local dependencies) (Pareja et al. 2020; Goyal, Chhetri, and Canedo 2020). For example, researchers (Pareja et al. 2020; Shi et al. 2021) have resorted to using GNNs along with RNNs to capture the dynamic evolving nature of graphs. With initial successes, these methods are inherently limited to capturing local interactions, missing out on other important information. For example, in the case of a dynamic money transaction graph, having a skewed representation of fraud users among genuine users, there would exist links between these fraud and genuine users, thus, giving rise to high-frequency signals. In the case of local neighborhood aggregation (attending to low-frequency signals), the majority node(genuine) pattern will cause the fraudulent pattern to diminish. Thus, capturing global properties becomes necessary. Here, the global information would help identify the fraud pattern eventually assisting in identifying the criminal. Similarly, in the citation graph, the local properties will help to understand the category of paper. In contrast, global properties will help to understand the amount of interdisciplinary research across research domains. Hence, learning the global dependencies is crucial for dynamic graphs which is a relatively unexplored area in its scientific literature.

In this paper we aim to encompass global dependencies in dynamic graphs, beyond local neighborhood aggregation, and view it through the lens of spectral graph theory (Hammond, Vandergheynst, and Gribonval 2019). For the same, we introduce a novel concept of *learnable spectral graph wavelets* to capture global interactions of dynamic graph. The concept of learnable wavelet has following motivation: (i) Computing the spectra using full eigen decomposition is computationally expensive and requires $\mathcal{O}(N^3)$ time. Moreover, this gets even more computationally costly by a factor of the number of timesteps(T) considered for dynamic graphs that evolve over every timestep. Spectral graph wavelets can be computed efficiently in $\mathcal{O}(N + |\mathcal{E}|)$ as we shall see in the following sections. (ii) Wavelets are sparser as compared to the graph Fourier transform adding to the computational benefits (Tremblay and Borgnat 2014). (iii) Wavelets give a sense of *localization* in the *vertex domain* of the graphs thus enabling interpretability of the convolutions while also being able to capture the global properties of the graph by changing the scale parameter (c.f., Figure 1). iv) Focusing on dynamic learnable wavelets helps in capturing global properties in that evolving spectra as the graph changes with time, where static and non-learnable wavelet methods (Xu et al. 2019a) show empirical limitations.

Furthermore, build on the recent success of capturing neighborhood features for evolving graphs, we propose to learn *homogeneous representation* of spatial and spectral

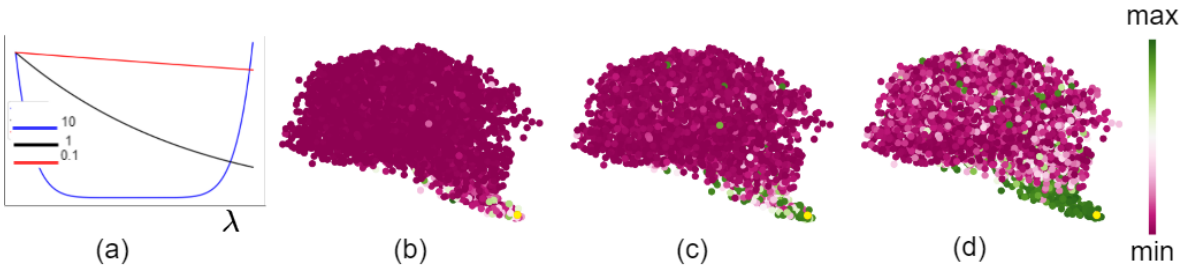


Figure 1: For Brain dataset (Xu et al. 2019b) at a given timestep, the figure (a) shows the learned filter functions at different scales by our proposed framework resembling band reject, low pass and all pass filters. The three diagrams to the right show the corresponding wavelets for a node (colored in yellow in figures (b,c,d) at lowest tip of the Brain) in the brain stem region. The graph nodes indicate the Regions of Interest(ROI) in brain. As scales change, the ROIs that the wavelet focuses on change from concentrated(local interactions) to diffused(global interactions). Moreover, the wavelets respect the brain structure and focus on the concerned region(brain stem in this case) thus mitigating noise due to interactions from unrelated regions.

features to apprehend both local and global dependencies. Our approach is very intuitive as it keeps the proven local information intact whilst adding the global properties through learnable graph wavelet approach. Similar approaches have been proven in computer vision (Srinivas et al. 2021) and NLP (Prasad and Kan 2019) where such restrictive inductive bias work well on learning local properties but miss out on global interactions. Also, the local aggregation leads to problems such as over smoothing and may not work well on heterophilic graphs (Wu et al. 2021b). Thus, using our methods will help to alleviate the drawbacks of the existing popular methods on dynamic graphs.

Our key contributions are two-fold: 1) a novel approach to *learning spectral wavelets* on dynamic graphs for capturing global dependencies (with its theoretical foundations), 2) a novel framework named DEFT that combines spectral features obtained using learnable wavelets into spatial feature of the evolving graphs. For effective use in downstream tasks, DEFT integrates the spatial and spectral features into *homogeneous representations* which allows capturing shift invariance (Oppenheim and Schafer 1975) among the node features that could arise from the temporal nature.

Related Works

There has been considerable work on static graphs from the spatial and spectral domain perspective. Some works such as (Kipf and Welling 2017; Hamilton, Ying, and Leskovec 2017; Veličković et al. 2018) focus on effectively learning spatial properties. Similarly, efforts such as (Levie et al. 2018; Balcilar et al. 2020; Bastos et al. 2022) have looked at graphs from a spectral perspective. GWNN (Xu et al. 2019a) has proposed to use spectral graph wavelets on static graphs. However, it obtains static wavelets using heat kernel-based filters. Unlike GWNN, we learn spectral graph wavelets for dynamically evolving graphs. We observe in the experiment section that learnable wavelets perform significantly better for dynamic graphs compared to static wavelet methods.

One straight-forward way to use methods developed for static graphs on evolving dynamic graphs is to use RNN modules such as GRU and LSTM in addition to GNN modules to capture the evolving graph dynamism. This idea has been explored in works such as (Seo et al. 2016; Narayan and Roe 2018; Manessi, Rozza, and Manzo 2020). However, these models suffer a performance drop if new nodes are introduced in the graph; the GNNs may not be able to adapt to the evolving graph structure. Thus, EvolveGCN (Pareja et al. 2020) has introduced to use RNNs to learn the parameters of the evolving GNN. The GNN, a GCN (Kipf and Welling 2017) in this case, is thus used ahead of the RNN module that captures the graph dynamism and offers promising results on dynamic graphs. EvolveGCN has limitation that it generalizes to unseen nodes in future timesteps, which is not always the case in real-world scenarios. Xu et al. (2019c) proposed similar approach by learning parameters of a GAT with an RNN that focus on graph topology discrepancies.

Furthermore, autoencoder-based methods have been introduced, such as in (Goyal et al. 2017; Goyal, Chhetri, and Canedo 2020; Xu, Singh, and Karniadakis 2022) that focus on reconstructing the graphs in future timesteps as an objective. Xiang, Huang, and Wang (2022) proposed a meta-learning framework in which the objective is to predict the graph at future timestep. However, these works focus only on learning spatial properties to capture local dependencies and ignores the global dependencies that may emerge due to dynamic nature of the graph.

Preliminaries

Consider a graph with vertices and edges as $(\mathcal{V}, \mathcal{E})$ and adjacency matrix A. The laplacian($L = D - A$) can be decomposed into its orthogonal basis, namely the eigenvectors(U) and eigenvalues(Λ) as:$L = U\Lambda U^*$. Let $X \in R^{N\times d}$ be the signal on the nodes of the graph. The Fourier Transform $\hat{X}$ of X is then given as: $\hat{X} = U^*X$.

Spectral graph wavelet transforms (Hammond, Vandergheynst, and Gribonval 2011) are obtained by functions of the laplacian L. Consider a linear self-adjoint operator($g(L)$) that acts on each component in the spectral space. We define a parameter s for the scaling in the spectral space. The spectral graph wavelet at any given vertex n is defined as the impulse(δ_n) response of the wavelet operator

at vertex n:

$$\psi_{s,n}(m) = \sum_{k=1}^{N} g(s\lambda_k) U_k^*(n) U_k(m)$$

The n-th wavelet coefficients at scale s represented as $W_f(s,n)$ can be obtained by taking the inner product of the function f in the *vertex domain* with these wavelets as

$$W_f(s,n) = \langle \psi_{s,n}, f \rangle = \sum_{k=1}^{N} g(s\lambda_k) \hat{f}(k) U_k(m) \quad (1)$$

In our work, we propose to learn the wavelet coefficients for dynamic graphs, where the exact form of the scaling function $g(s\lambda_k)$ is parameterized.

Method

Figure 2 illustrates the proposed DEFT (Dynamic wavElets For global inTeractions) framework that comprises the following modules: 1) Spectral component: focuses on global features of the graph in the form of learnable spectral graph wavelets. 2) Spatial component: the necessity of this component is to mitigate the unsmooth spectrum phenomenon i.e. the node features gets correlated with eigenvectors corresponding to high frequencies causing information loss (Yang et al. 2022). To resolve this, spatial module focus on the local neighborhood of the graph in addition to that captured by the wavelets. 3) Integration module: Finally, for a *homogeneous representation* of the global properties captured by the spectral component and the local properties learned by the spatial component, we propose to learn invariant representations and use these in an aggregation module.

Spectral Module

We aim to capture global features without requiring the full eigen decomposition of the laplacian. Here, we propose to approximate the wavelet operator using some function. We use the Chebyshev polynomials to be consistent with the literature (Hammond, Vandergheynst, and Gribonval 2011). It is known that for any function h with domain $y \in [-1,1]$ there exists a convergent Chebyshev series:

$$h(y) = \sum_{k=0}^{\infty} c_k T_k(y)$$

with the Chebyshev polynomials satisfying the recurrence relation $T_k(y) = 2yT_{k-1}(y) - T_{k-2}(y), T_0 = 1, T_1 = y$ and the coefficients c_k are given by:

$$c_k = \frac{2}{\pi} \int_{-1}^{1} \frac{T_k(y)h(y)}{\sqrt{1-y^2}} dy = \frac{2}{\pi} \int_0^{\pi} cos(k\theta) h(cos(\theta)) d\theta$$

In order to approximate the function $g(s_j x)$ for each scale j, we need to bring the domain x in $[-1,1]$. Noting that $0 \leq x \leq \lambda_{max}$ for the laplacian L, we perform the transform $y = \frac{x-a}{a}, a = \frac{\lambda_{max}}{2}$. We now define $\overline{T_k}(x) = T_k(y) = T_k(\frac{x-a}{a})$ and the approximation for g looks as below

$$g(s_j x) = \sum_{k=0}^{\infty} c_{j,k} \overline{T_k}(x) \quad (2)$$

with the coefficients given by,

$$c_{j,k} = \frac{2}{\pi} \int_0^{\pi} cos(k\theta) g(s_j(a(cos(\theta)+1))) d\theta$$

We truncate the polynomial to M terms, which is the filter order. The coefficients $c_{j,k}$ which are analytical coefficients of the filter function as desired are approximated using functions parameterized by GNNs and MLPs, as we shall see next. f_c^s is the parameterized form of it, obtained in spectral module at scale s. A GNN is used to perform message passing over the input graph along with the node features v at layer l for neighborhood $\mathcal{N}$.

$$v_{im}^l = A_f(v_j^l | v_j \in \mathcal{N}(v_i)), \quad v_i^{l+1} = U_f(v_i^l, v_{im}^l)$$

Here, A_f, U_f are the aggregation and update functions, respectively. The update function could contain a non-linearity such as leaky ReLU for better expressivity of the learned function. Since we intend to learn the filter coefficients $f_c \in R^M$ for the concerned graph(G), we apply a pooling layer to get an intermediate vector representation($v_G \in R^{d_1}$) from the output of the GNN. The pooling layer converts a set of vectors(one for each node of the graph) to a single vector representation. For the final filter coefficients f_c, we apply a two-layer MLP with activation(σ) to v_g

$$f_c = W_2 \ \sigma(W_1 v_g) \quad (3)$$

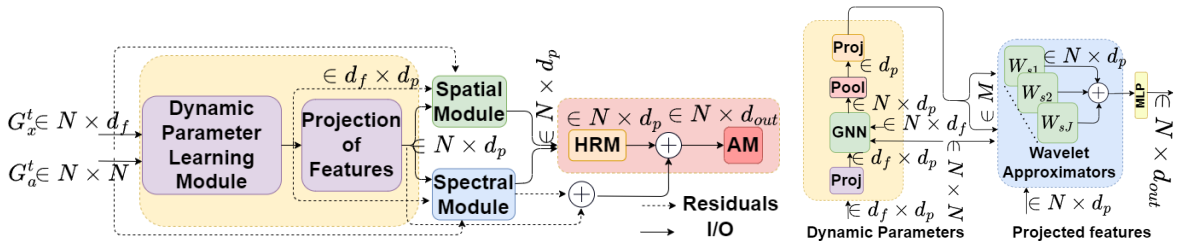
where $W_1 \in R^{d_2 \times d_1}, W_2 \in R^{d_2 \times M}$ are learnable weights. Since the two-layer MLP is a universal approximator (Hornik 1991) we can be assured of the existence of a function in this space that learns the desired mapping to the filter coefficients. In principle, any message passing GNN can be used to perform the update and aggregation steps. This process can be repeated with multiple GNNs for learning multiple filter functions. As we consider dynamic graphs, we would like to evolve the parameters of the GNN with time (Dynamic Parameter learning Module of Figure 2, common for both spectral and spatial components). Inspired from (Pareja et al. 2020), we use an RNN module for generating the parameters for the GNN in layer l at time t:

$$W_t^l = RNN(H_t^l, W_{t-1}^l) \quad (4)$$

where W_t^l and H_t^l are the hidden state and input at layer l and time t of the RNN. In the below pseudo code, we outline our method to evolve the Spectral Module($ESpectral$) for dynamically learning filter coefficients per timestep

function $f_{c_t} = ESpectral(A_t, H_t^{(l)}, W_{t-1}^{(l)})$
  $W_t^{(l)} = RNN(W_{t-1}^{(l)})$
  $H_t^{(l+1)} = GNN(A_t, H_t^{(l)}, W_t^{(l)})$
  $v_{g_t} = Pool(A_t, H_t^{(l+1)})$
  $f_{c_t} = W_2 \ \sigma(W_1 v_{g_t})$
end function

After learning GNN parameters, we need to learn filter coefficients for evolving graph. Learning the filter coefficients f_c^s at a given timestep, we can obtain the wavelet operator $g(L)$ at scale $s = 1$ using equation 2. For operators at a given scale



(a) (b)

Figure 2: Figure (a) shows the DEFT architecture with tensor dimensions on the arrows. The yellow outer box shows the modules that learn the weight parameters of the GNNs in an evolving manner. These parameters are then given to the spectral and spatial modules to learn the corresponding features. The learned features then propagates to the *homogeneous representation* module (HRM) followed by the aggregation module(AM) which together forms the integration module (red outer box) of our framework. Figure (b) explains the components of the spectral module. Similar to image (a), the yellow outer box learns the filter functions for the wavelet approximators. W_{s_i} is the wavelet coefficient (vector representation) at scale s_i.

.

$s > 0$, we could learn different parameterizations of the filter function at every scale. Note that approximating the functions at different scales in this manner would incur a storage cost of $\mathcal{O}(JN)$ for the filter coefficients. Along with this the storage and computation complexity would increase J times for dynamically computing the filter coefficients from the GNNs. Thus we propose a "rescaling trick" wherein to obtain the operator at any scale $s > 0$, we perform the appropriate change of variables in equation 2 to get $g(sL)$, keeping the coefficients $c_{j,k}$ fixed. That is for a scale s, instead of mapping $\lambda \rightarrow g(\lambda)$ it would be mapped to $\lambda \rightarrow g(s\lambda)$. It helps maintain parameter efficiency(by a factor of J) as the GNN weights(and also filter coefficients) are tied across all the scales. Note here the exact filter learned would vary since the maximum frequency is the same. Hence, a bandpass at scale 'one' may get converted to a highpass at scale 'two'. Once we approximate $g(\lambda)$, we use it in learning the wavelet coefficients (output of spectral module) $W_f(s, n)$ as described in equation 1.

.

Lemma 1. *Consider $G^t(\lambda)$ to be the filter function at time t. Assume the Markov property to hold in the dynamic setting where the desired filter function($G^{t+1}(\Lambda)$) at time $t+1$ depends on the past state at time t($G^t(\Lambda)$). Consider this mapping between the past state and the current states to be captured by an arbitrary functional f such that $G^{t+1}(\lambda) = f(G^t(\lambda_1), G^t(\lambda_2), \dots G^t(\lambda_N), \lambda)$ and we assume f to be L lipschitz continuous. Further, let $C_t = U_t G^t(\lambda) U_t^T \in R^{N \times N}$ represents the convolution support of the desired spectral filter and C_t^a be the learnt convolution support at time t. Then, we have:*

(i) $\left\|C_{t+1}^a - C_{t+1}\right\|_F \leq LN^2 \sqrt{\|C_t^a - C_t\|_F^2 + \epsilon_{ca}^2} + \epsilon_{fa}$

(ii) $\left\|C_{t+1}^a - C_t^a\right\|_F \leq \|C_{t+1} - C_t\|_F + 2\sqrt{N}\epsilon_{ca}$

where ϵ_{ca} and ϵ_{fa} are the filter polynomial(Chebyshev) and function approximation errors and depends on the order of the polynomial, number of training samples, model size etc.

Above result gives us a relation between the error at times t and $t+1$ and has a factor of $N^2 L$. Thus it requires the filter function to be smooth($L < \frac{1}{N^2}$) for convergence because under the given Markov assumptions, the past errors could accumulate in future timesteps. Hence, GRU/LSTM in the Dynamic Parameter Generation module is used to decouple the approximation of the filter function at a given timestep from the error in previous timesteps.

Spatial Module

Yang et al. (2022) concluded that high-frequency components of the signal on the graph get strengthened with a high correlation with each other, and the smooth signals become weak (unsmooth spectrum) i.e., the cosine similarity between the transformed signal and the low eigenvector reduces with the layers. In our setting, we illustrate that the factor by which the signal corresponding to the low-frequency component gets weakened is directly proportional to the magnitude of the frequency response at that frequency.

Lemma 2. *Let $G(\lambda)$ be the frequency response at frequency λ. Let $\lambda_1 \geq \lambda_2 \geq \cdots \geq \lambda_n$ be the eigenvalues in descending order and $p_1, p_2, \dots p_n$ be the corresponding eigenvectors of the laplacian of the graph. Define λ_{max} to be the eigenvalue at which $G(\lambda)$ is maximum. Let C^l represent the convolution support($UG(\Lambda)U^T$) of the spectral filter at layer l. Then the factor by which the cosine similarity between consecutive layers dampens is* $\lim_{l \to \infty} \frac{|cos(\langle C^{l+1} h, p_n \rangle)|}{|cos(\langle C^l h, p_n \rangle|} = \frac{G(\lambda_n)}{G(\lambda_{max})}$.

If λ_{max} belongs to higher regions of the spectrum and $G(\lambda_n)$ is lower, then as the layers increase, the signals will lose the low-frequency information. Since the Spectral Module aims to capture the high-frequency components that may lead to an unsmooth spectrum, we explicitly strengthen the low-frequency components by using local neighborhood aggregation. For this, we inherit the message passing GNNs, the parameters of which are generated using the RNN module inspired from (Pareja et al. 2020). The process to evolve the spatial module($ESpatial$) is:

1: **function** $[H_t^{(l+1)}, W_t^{(l)}] = ESpatial(A_t, H_t^{(l)}, W_{t-1}^{(l)})$

2: $\quad W_t^{(l)} = RNN(W_{t-1}^{(l)})$

3: $H_t^{(l+1)} = GNN(A_t, H_t^{(l)}, W_t^{(l)})$

4: **end function**

Integration Module

Homogeneous representation Module(HRM): aims to achieve homogeneous representations from the spatial and spectral properties along with time features, which is essential for its usage in downstream tasks. A straightforward way is by concatenating two representations. However, due to the dynamic nature of the graph that evolves with time, we propose a learnable module that provides a notion of distance between the representations and helps induce position/structure information. Due to graph dynamism, it is important for the features to satisfy the shift invariance property:

Definition 1 (Shift Invariance (Li et al. 2021)). *Any two vectors $v_i = f_1(v_i^{'})$ and $v_j = f_1(v_j^{'})$ satisfy the shift invariance property if the inner product $\langle v_i, v_j \rangle$ is a function of $v_i^{'} - v_j^{'}$ i.e. $\langle v_i, v_j \rangle = f_2(v_i^{'} - v_j^{'})$.*

f_1 is an arbitrary function and f_2 is a linear transformation of f_1. Above property ensures that the relative distance between two nodes is maintained in the inner product space even if their absolute positions change(for example with addition of new nodes in the graph with time). Inspired from Fourier features (Rahimi and Recht 2007, 2008), for a node i, the spectral($v_{gi} \in R^{d_g}$) and spatial($v_{li} \in R^{d_l}$) embeddings are concatenated($\|$) along with the timestamp($t \in R^{d_t}$) information if available. Then, it is passed to an MLP.

$$v_{gli} = MLP(v_{gi} \| v_{li} \| t)$$

In order to obtain Fourier features(v_{ffi}) from above intermediate representation, we take the element-wise sine,cosine and concatenate the two as:

$$v_{ffi} = sin(v_{gli}) \| cos(v_{gli})$$

Property 1. *The vector v obtained by concatenation of the element wise sine and cosine of another vector $v^{'}$ i.e. $v = (sin(v^{'}) \| cos(v^{'}))$, satisfies the shift invariance property.*

The property 1 can be readily noted by observing that $cos(a-b) = cos(a)cos(b) + sin(a)sin(b)$. Taking the inner product($\langle . \rangle$) of the above features for two nodes(i, j) gives

$$\langle v_{ffi}, v_{ffj} \rangle = \sum cos(W_r(v_{gli} - v_{glj})) = ff_{W_r}(v_{gli} - v_{glj})$$

Property 1 is beneficial if these representations are used in an attention-based model such as (Vaswani et al. 2017; Veličković et al. 2018) as we get a notion of closeness(similarity) in the embedding space. The final *homogeneous representation* for node i (v_{hri}) is obtained as:

$$v_{hri} = W_{hr2} \ \ \sigma(W_{hr1}(sin(v_{gli}) \| cos(v_{gli})))$$

where $W_{hr2} \in R^{d \times d}, W_{hr1} \in R^{d \times (d_g + d_l + d_t)}$ are learnable weights and σ is the activation function.

Aggregation(AM): Once we achieve homogeneous representations for the features, we can use these in the downstream task by applying a layer of MLP. However, we also perform another level of aggregation to learn effective representations. While, in principle, we could use any of the existing message passing frameworks for this aggregation, we adopt a sparse variant of the attention mechanism inspired by (Vaswani et al. 2017) for computational benefits. Specifically, consider $X \in R^{N \times d_f}$ to be the node feature learned from the spatial and spectral modules. Now we define for the l-th layer, $W_Q^l, W_K^l \in R^{d_{out} \times d_f}$ to be the learnable weight matrices for the query and key of the self attention respectively. We apply self attention on the transformed features followed by softmax to get the aggregation weights $w_{ij}^l = softmax(\sum_{d_k} \hat{w_{ij}^l})$ between nodes i, j, where $\hat{w_{ij}^l} = \frac{W_Q^l X[i]^T \odot W_K^l X[j]^T}{d_{out}}$ if nodes i and j are connected in the graph and 0 otherwise.

Overall Complexity

The complexity of spectral module is $\mathcal{O}(|E| + N \sum_{j=0}^{J} M_j)$(cf., appendix) where M_j is the order of the polynomial of the j-th filter head. The spatial modules can compute the features in a $\mathcal{O}(|E| + N)$ complexity. The integration module further has two components: HRM and AM. The HRM module would have a computational complexity of $\mathcal{O}(N)$ whereas for the AM, it depends on the underlying aggregator. In our choices, it would be an $\mathcal{O}(|E| + N)$ complexity. Thus the overall computational complexity comes to $\mathcal{O}(3|E| + N(3 + \sum_{j=0}^{J} M_j)) = \mathcal{O}(|E| + N)$, for bounded degree graphs further reduces to $\mathcal{O}(N)$ for a given snapshot at time t.

Experiments

Now, we present comprehensive experiments to evaluate our proposed framework. We borrow datasets and its preprocessing/splitting settings used in previous best baselines (Xiang, Huang, and Wang 2022; Pareja et al. 2020).

Datasets: Table **??** summarizes eight datasets for link prediction, edge classification, and node classification. Each dataset contains a sequence of time-ordered graphs. SBM is a synthetic dataset to simulate evolving community structures. BC-OTC is a who-trusts-whom transaction network where the node represents users, and the edges represent the ratings that a member gives others in a range of -10(maximum distrust) to +10(maximum trust). BC-Alpha is similar to BC-OTC, albeit the transactions are on a different bitcoin network. UCI dataset is a student community network where nodes represent the students, and the edges represent the messages exchanged between them. AS dataset summarizes a temporal communication network indicating traffic flow between routers. In the Reddit dataset, nodes are the source and target subreddits, and the edges represent the sentiment's polarity between users. The Elliptic dataset represents licit vs illicit transactions on the elliptic network of bitcoin transactions. The nodes represent transactions and edges represent payment flows. Finally, for Brain dataset, nodes represent tiny regions/cubes in the brain, and the edges are their connectivity.

Baselines: In order to show the efficacy of our framework, we compare with the following competitive baselines: (1) *Static graph methods*: GCN (Kipf and Welling 2017), GAT (Veličković et al. 2018). (2) *Dynamic graph*

	# Nodes	# Edges	# Time Steps (Train / Val / Test)	Task
BC-OTC	5,881	35,588	95 / 14 / 28	EC
BC-Alpha	3,777	24,173	95 / 13 / 28	EC
Reddit	55,863	858,490	122 / 18 / 34	EC
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

Table 1: Data set statistics and details. In the task column, LP refers to Link Prediction, EC refers to Edge Classification and NC is Node Classification.

methods such as GCN-GRU (Pareja et al. 2020), DynGEM (Goyal et al. 2017), GAEN (Shi et al. 2021), the variants of dyngraph2vec (Goyal, Chhetri, and Canedo 2020): DG2vec v1 and DG2vec v2, the best performing variants of EvolveGCN (Pareja et al. 2020): EGCN-O and EGCN-H, and the two variants of LEDG (Xiang, Huang, and Wang 2022): LEDGCN and LEDGAT.

Task and Experiment Settings: For the link prediction task, we report mean reciprocal rank(MRR) and mean average precision(MAP) as in the baselines. For each of the given nodes u, v, our model gives embeddings in the final layer h_u^t, h_v^t respectively. We predict the link between these nodes by concatenating the vectors for the corresponding nodes. Following the standard practices, we perform negative sampling and optimize the cross entropy loss function. For edge classification, our model classifies the edges between two nodes(u, v) by concatenating the vectors for the corresponding nodes(h_u^t, h_v^t) at time t and we report the micro F1 score. For node classification similar to the baselines, our model reports micro F1 (Brain dataset) or minority class F1 (Elliptic dataset) by classifying given node nodes(u) as belonging to a certain class at time t.

The best hyperparameters search has the range as: Number of layers $\in \{1, 2\}$, Hidden dimension $\in \{32, 64, 128\}$, Number of heads $\in \{4, 8, 16\}$, Filter order $\in \{4, 8, 16\}$, Wavelet scales $\in [0.1, 10]$. The rest of the parameters and settings are borrowed from previous works (Pareja et al. 2020; Xiang, Huang, and Wang 2022).

Variants of our Framework: In the main results we show three variants of our method with different aggregators. We use the MLP(DEFT-MLP), GAT(DEFT-GAT) and sparse Transformer(DEFT-T) as aggregators. The rationale behind selecting GAT and MLP in addition to the proposed transformer aggregator is to contrast it with a less expressive static attention mechanism in GAT and a control in MLP.

Results

Results on Link Prediction

Table 2 summarizes link prediction results. The key results are: 1) across datasets, our model (DEFT-T) significantly outperform all baselines which focus on learning local dependencies (in static and dynamic graph settings). It illustrates our framework's effectiveness in learning homogeneous representations of local and global dependencies in evolving graphs. 2) Interestingly, a simple MLP aggregator (DEFT-MLP) on the learned spectral and spatial properties already achieves better results than most of the baselines on UCI and AS datasets (including ones for dynamic graphs). 3) The transformer variant (DEFT-T) gives consistently better results than the GAT and MLP variants indicating the aggregation using the sparse self-attention of transformers is essential for the framework on these datasets.

Datasets	SBM		UCI		AS	
Metrics	MAP	MRR	MAP	MRR	MAP	MRR
GCN	0.189	0.014	0.0001	0.047	0.002	0.181
GAT	0.175	0.013	0.0001	0.047	0.020	0.139
DynGEM	0.168	0.014	0.021	0.106	0.053	0.103
GCN-GRU	0.190	0.012	0.011	0.099	0.071	0.339
DG2vec V1	0.098	0.008	0.004	0.054	0.033	0.070
DG2vec V2	0.159	0.012	0.021	0.071	0.071	0.050
GAEN	0.183	0.008	0.0001	0.049	0.130	0.051
EGCN-H	0.195	0.014	0.013	0.090	0.153	0.363
EGCN-O	0.199	0.014	0.027	0.138	0.114	0.275
LEDG-GCN	0.196	0.015	0.032	0.163	0.193	0.469
LEDG-GAT	0.182	0.012	0.026	0.149	0.233	0.384
DEFT-MLP	0.166	0.012	**0.054**	0.172	0.174	0.357
DEFT-GAT	0.097	0.008	0.050	0.170	0.031	0.093
DEFT-T	**0.242**	**0.022**	0.050	**0.201**	**0.588**	**0.647**

Table 2: Link prediction results where mean average precision (MAP) and mean reciprocal rank (MRR) are displayed. Best values are bold, second bests are underlined.

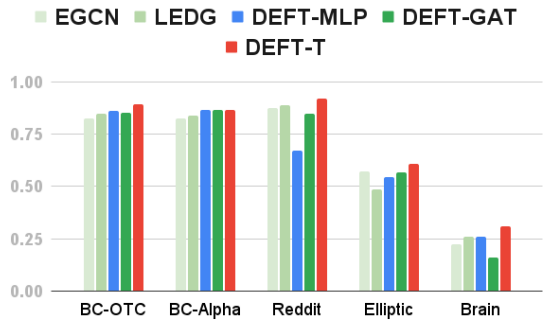


Figure 3: Performance of edge classification and node classification, with F1 scores on the y-axis.

Results on Edge Classification

The results of edge classification on the three datasets (BC-OTC, BC-Alpha, Reddit) are given in Figure 3 and show the micro averaged F1 score. We observe from the results that at least one variant of our method outperforms the best baselines variants: EvolveGCN and LEDG. Our MLP-based configuration achieves better results than baselines on BC-OTC and BC-Alpha, whereas DEFT-T consistently outperforms the baselines and other DEFT variants.

Results on Node Classification

Figure 3 shows the results of node classification on Brain(homophily ratio (Zhu et al. 2020): 26%) and Elliptic(homophily ratio: 96%) datasets. DEFT would consider

the brain structure on the Brain dataset while capturing global interactions. DEFT has a substantial performance gain on the Brain dataset, which has a high heterophily ratio requiring capturing global interactions in graphs compared to existing models that only perform local aggregation. On the homophilic Elliptic dataset, DEFT also performs good by capturing low-frequency signals. It confirms that when the spatial and spectral properties are combined using an optimal aggregator to learn the local and global dependencies, the model results in robust performance across tasks.

Ablation Studies

Datasets	SBM		UCI	
Metrics	MAP	MRR	MAP	MRR
GWNN	0.1789	0.0121	0.0076	0.0820
Transformer	0.2052	0.0174	0.0308	0.1441
GT	0.2166	0.01805	0.0310	0.1414
SAN	0.2143	0.0180	0.0388	0.1822
DEFT-woSpectral	0.2123	0.0171	0.0460	0.1764
DEFT-woSpatial	0.2083	0.0171	0.0421	0.1806
DEFT-woHRM	0.2410	0.0217	0.0429	0.1588
DEFT-staticSpectral	0.2297	0.0198	0.0419	0.1506
DEFT-T	0.2421	0.0220	0.0501	0.2007

Table 3: Table shows the link prediction results for the static wavelet and transformer architectures along with ablations of individual components of our model.

Model Ablation Study: To study the effect of each module of DEFT, we systematically remove modules and create several sub-configuration of our best performing model (DEFT-T): (i) **DEFT-woSpectral** does not have the spectral module (ii) **DEFT-woSpatial** does not have the spatial module (iii) **DEFT-woHRM** with the Homogeneous Representation Module removed(the spatial and spectral features are simply added), (iv) **DEFT-staticSpectral** has the spectral wavelets that are learnable but do not evolve with time and in a graph-specific manner. We also compare our best model with recent transformer-based models used for static graphs such as transformer (Vaswani et al. 2017), SAN (Kreuzer et al. 2021), GT (Dwivedi and Bresson 2020), and static wavelet baseline GWNN (Xu et al. 2019a).

The ablation study (Table 3) provides several key insights to understand proposed DEFT framework. Firstly, the lesser values reported by static (non-learnable) wavelet GNN (i.e., GWNN) and the static (but learnable) variant of our model (i.e., DEFT-staticSpectral) conclude that learning dynamic wavelets benefits in the case of evolving graphs. Secondly, the significant drop in performance when the spatial or spectral module is removed provides an essential finding that for dynamic graphs, these modules learn useful and orthogonal representations. When these modules are combined using an effectively learned homogeneous representation, the model (DEFT-T) provides significantly superior performance than its variants. Lastly, the static transformer models consistently report lesser values than our model.

Filter parameter selection: In spectral module, there are two parameters to be tuned: filter(polynomial) order and the wavelet scale. In this study, we try to understand how

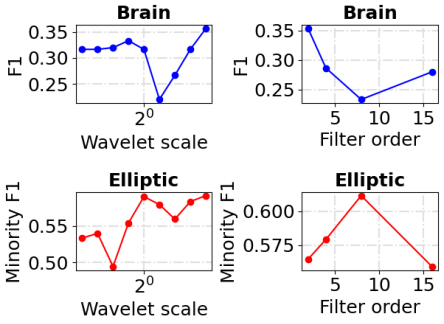


Figure 4: Effect of the parameters wavelet scale (plotted in log scale) and filter order on the performance of the model.

selection of parameters affects model performance. From Figure 4 illustrating results on Brain and Elliptic datasets, we make the following observations: (1) *Wavelet scales*: In Brain dataset, the result dips with higher scale and then increases. For Elliptic, a general increasing trend is observed. In the case of the learnt filters for the given graphs, it seems that as the scales decrease the aggregation range of the spectral GNN increases(as in figure 1). Thus, brain dataset being heterophilic seems to benefit from scales in the lower region. Larger scales may work well for the homophilic Elliptic dataset as the aggregation focuses on a relatively smaller neighborhood, while also capturing the heterophily signals in the minority illicit transactions. (2) *Filter order*: On Brain dataset, there is a decreasing trend with increasing filter order, whereas, for Elliptic, an optimal result is attained in the middle region. This could be because the Brain dataset being small needs simpler filters(such as single mode band reject etc.); with higher filter order, it may have overfit. The larger Elliptic dataset, on the other hand, benefits from a larger filter order. Despite being homophilic for the majority nodes, it may need complex filters to handle the minority nodes due to the peculiar and global nature of illicit transactions. Thus, it benefits from relatively higher filter orders. These observations conclude that the parameters of the order of polynomial and the wavelet scales are dataset dependent.

Conclusion

Our work proposes a concept of *learnable spectral graph wavelets* to capture global dependencies by omitting the need for full eigen decomposition of dynamic graphs. Furthermore, we implement it in the DEFT framework that integrates the graph's spectral and spatial properties for dynamic graphs. From the results on a wide range of tasks for dynamic graphs, we infer that the proposed method is able to capture the local(short range) and global(extended range) properties effectively. Also, it can capture a broad range of interactions respecting the graph structure, thus avoiding noise while being sparse(computationally efficient). Future works could build upon frameworks for joint time and graph Fourier transform for dynamic graphs to generate wavelets and study if these could be used for graph sampling to provide computational efficiency on extremely large graphs.

Acknowledgments

This work was partly supported by JSPS KAKENHI Grant Number JP21K21280. We thank anonymous reviewers for constructive feedback.

References

Balcilar, M.; Renton, G.; Héroux, P.; Gaüzère, B.; Adam, S.; and Honeine, P. 2020. Analyzing the expressive power of graph neural networks in a spectral perspective. In *International Conference on Learning Representations*.

Bastos, A.; Nadgeri, A.; Singh, K.; Kanezashi, H.; Suzumura, T.; and Mulang', I. O. 2022. How Expressive are Transformers in Spectral Domain for Graphs? *Transactions on Machine Learning Research*.

Dwivedi, V. P.; and Bresson, X. 2020. A Generalization of Transformer Networks to Graphs. arXiv:2012.09699.

Goyal, P.; Chhetri, S. R.; and Canedo, A. 2020. dyngraph2vec: Capturing network dynamics using dynamic graph representation learning. *Knowl. Based Syst.*, 187.

Goyal, P.; Kamra, N.; He, X.; and Liu, Y. 2017. DynGEM: Deep Embedding Method for Dynamic Graphs. *IJCAI Workshop on Representation Learning for Graphs*.

Hamilton, W.; Ying, Z.; and Leskovec, J. 2017. Inductive representation learning on large graphs. In *Advances in neural information processing systems*, 1024–1034.

Hammond, D. K.; Vandergheynst, P.; and Gribonval, R. 2011. Wavelets on graphs via spectral graph theory. *Applied and Computational Harmonic Analysis*, 30(2): 129–150.

Hammond, D. K.; Vandergheynst, P.; and Gribonval, R. 2019. The spectral graph wavelet transform: Fundamental theory and fast computation. In *Vertex-Frequency Analysis of Graph Signals*, 141–175. Springer.

Han, K.; Wang, Y.; Guo, J.; Tang, Y.; and Wu, E. 2022. Vision GNN: An Image is Worth Graph of Nodes. *arXiv preprint arXiv:2206.00272*.

Hornik, K. 1991. Approximation capabilities of multilayer feedforward networks. *Neural networks*, 4(2): 251–257.

Kazemi, S. M.; Goel, R.; Jain, K.; Kobyzev, I.; Sethi, A.; Forsyth, P.; and Poupart, P. 2020. Representation learning for dynamic graphs: A survey. *J. Mach. Learn. Res.*, 21(70): 1–73.

Khoshraftar, S.; and An, A. 2022. A Survey on Graph Representation Learning Methods. *arXiv preprint arXiv:2204.01855*.

Kipf, T. N.; and Welling, M. 2017. Semi-Supervised Classification with Graph Convolutional Networks. In *5th International Conference on Learning Representations, ICLR 2017*.

Kreuzer, D.; Beaini, D.; Hamilton, W. L.; Létourneau, V.; and Tossou, P. 2021. Rethinking Graph Transformers with Spectral Attention. *NeurIPS 2021*, abs/2106.03893.

Levie, R.; Monti, F.; Bresson, X.; and Bronstein, M. M. 2018. Cayleynets: Graph convolutional neural networks with complex rational spectral filters. *IEEE Transactions on Signal Processing*, 67(1): 97–109.

Li, Y.; Si, S.; Li, G.; Hsieh, C.-J.; and Bengio, S. 2021. Learnable Fourier Features for Multi-dimensional Spatial Positional Encoding. In Beygelzimer, A.; Dauphin, Y.; Liang, P.; and Vaughan, J. W., eds., *Advances in Neural Information Processing Systems*.

Manessi, F.; Rozza, A.; and Manzo, M. 2020. Dynamic graph convolutional networks. *Pattern Recognit.*, 97.

Narayan, A.; and Roe, P. H. 2018. Learning graph dynamics using deep neural networks. *IFAC-PapersOnLine*, 51(2): 433–438.

Oppenheim, A. V.; and Schafer, R. W. 1975. *Digital signal processing*. Prentice-Hall international editions. Prentice-Hall. ISBN 0132141078.

Pareja, A.; Domeniconi, G.; Chen, J.; Ma, T.; Suzumura, T.; Kanezashi, H.; Kaler, T.; Schardl, T. B.; and Leiserson, C. E. 2020. EvolveGCN: Evolving Graph Convolutional Networks for Dynamic Graphs. In *The Thirty-Fourth AAAI Conference on Artificial Intelligence, AAAI 2020*, 5363–5370. AAAI Press.

Prasad, A.; and Kan, M.-Y. 2019. Glocal: Incorporating global information in local convolution for keyphrase extraction. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, 1837–1846.

Rahimi, A.; and Recht, B. 2007. Random Features for Large-Scale Kernel Machines. In *Advances in Neural Information Processing Systems 20, Proceedings of the Twenty-First Annual Conference on Neural Information Processing Systems*, 1177–1184. Curran Associates, Inc.

Rahimi, A.; and Recht, B. 2008. Weighted Sums of Random Kitchen Sinks: Replacing minimization with randomization in learning. In *Advances in Neural Information Processing Systems 21, Proceedings of the Twenty-Second Annual Conference on Neural Information Processing Systems*, 1313–1320. Curran Associates, Inc.

Seo, Y.; Defferrard, M.; Vandergheynst, P.; and Bresson, X. 2016. Structured Sequence Modeling with Graph Convolutional Recurrent Networks. *CoRR*, abs/1612.07659.

Shi, M.; Huang, Y.; Zhu, X.; Tang, Y.; Zhuang, Y.; and Liu, J. 2021. GAEN: Graph Attention Evolving Networks. In *IJCAI*, 1541–1547.

Srinivas, A.; Lin, T.-Y.; Parmar, N.; Shlens, J.; Abbeel, P.; and Vaswani, A. 2021. Bottleneck transformers for visual recognition. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 16519–16529.

Stärk, H.; Ganea, O.; Pattanaik, L.; Barzilay, R.; and Jaakkola, T. S. 2022. EquiBind: Geometric Deep Learning for Drug Binding Structure Prediction. *CoRR*, abs/2202.05146.

Tremblay, N.; and Borgnat, P. 2014. Graph wavelets for multiscale community mining. *IEEE Transactions on Signal Processing*, 62(20): 5227–5239.

Vaswani, A.; Shazeer, N.; Parmar, N.; Uszkoreit, J.; Jones, L.; Gomez, A. N.; Kaiser, Ł.; and Polosukhin, I. 2017. Attention is all you need. In *Advances in neural information processing systems*, 5998–6008.

Veličković, P.; Cucurull, G.; Casanova, A.; Romero, A.; Liò, P.; and Bengio, Y. 2018. Graph Attention Networks. In *International Conference on Learning Representations*.

Wu, L.; Chen, Y.; Ji, H.; and Liu, B. 2021a. Deep Learning on Graphs for Natural Language Processing. In *Proceedings of the 44th International ACM SIGIR Conference on Research and Development in Information Retrieval*, 2651–2653. Association for Computing Machinery.

Wu, Z.; Jain, P.; Wright, M.; Mirhoseini, A.; Gonzalez, J. E.; and Stoica, I. 2021b. Representing long-range context for graph neural networks with global attention. *Advances in Neural Information Processing Systems*, 34: 13266–13279.

Xiang, X.; Huang, T.; and Wang, D. 2022. Learning to Evolve on Dynamic Graphs (SA). In *Thirty-Sixth AAAI Conference on Artificial Intelligence, AAAI 2022*. AAAI Press.

Xu, B.; Shen, H.; Cao, Q.; Qiu, Y.; and Cheng, X. 2019a. Graph Wavelet Neural Network. In *International Conference on Learning Representations*.

Xu, D.; Cheng, W.; Luo, D.; Gu, Y.; Liu, X.; Ni, J.; Zong, B.; Chen, H.; and Zhang, X. 2019b. Adaptive Neural Network for Node Classification in Dynamic Networks. In Wang, J.; Shim, K.; and Wu, X., eds., *2019 IEEE International Conference on Data Mining, ICDM 2019, Beijing, China, November 8-11, 2019*, 1402–1407. IEEE.

Xu, K.; Hu, W.; Leskovec, J.; and Jegelka, S. 2019c. How Powerful are Graph Neural Networks? arXiv:1810.00826.

Xu, M.; Singh, A. V.; and Karniadakis, G. E. 2022. DynG2G: An Efficient Stochastic Graph Embedding Method for Temporal Graphs. *IEEE Transactions on Neural Networks and Learning Systems*.

Yang, M.; Shen, Y.; Li, R.; Qi, H.; Zhang, Q.; and Yin, B. 2022. A New Perspective on the Effects of Spectrum in Graph Neural Networks. In *International Conference on Machine Learning*, 25261–25279. PMLR.

Ying, C.; Cai, T.; Luo, S.; Zheng, S.; Ke, G.; He, D.; Shen, Y.; and Liu, T. 2021. Do Transformers Really Perform Badly for Graph Representation? In *Advances in Neural Information Processing Systems 34: Annual Conference on Neural Information Processing Systems 2021, NeurIPS 2021*, 28877–28888.

Zhu, J.; Yan, Y.; Zhao, L.; Heimann, M.; Akoglu, L.; and Koutra, D. 2020. Beyond Homophily in Graph Neural Networks: Current Limitations and Effective Designs. In *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020*.