Summary

This material presents the supplementary paper from the main paper due to the space limitation. In Section 1, the detailed calculation process of gradient for coefficients are presented. In Section 2, the proofs of each Lemma from the main paper are provideds. Additional details for LSAP architecture that were omitted from the main manuscript are presented in Section 3. In Section 4, hyperparameters of LSAP and baselines for both node classification and graph classification and more results on brain network classification on ADNI data are shown. The implementation details are written in Section 5, and detailed model complexity is in Section 6. Adjustment of the trained scales at training phase is visualized in Section 7. Finally, we additionally discuss our framework in Section 8.

1. Gradients of Polynomial Coefficients with Scale

In this section, we are going to show the detailed derivation of gradient $\frac{\partial c_{s,n}}{\partial s}$ for the three polynomials, i.e., Chebyshev, Hermite and Laguerre.

Chebyshev Polynomial. The expansion coefficient for Chebyshev polynomial is given as

$$c^T_{s,n} = (2-\delta_{n0})(-1)^n e^{-\frac{sb}{2}} I_n\left(\frac{sb}{2}\right), \tag{1}$$

and the derivative of $c^T_{s,n}$ with respect to s is obtained as

$$\begin{aligned}
\frac{\partial c^T_{s,n}}{\partial s} &= 2(-1)^n e^{\frac{-sb}{2}} \left(\frac{-b}{2}\right) I_n\left(\frac{sb}{2}\right) \\
&\quad + 2(-1)^n e^{\frac{-sb}{2}} I_n\left(\frac{sb}{2}\right)\frac{b}{2} \\
&= 2(-1)^n e^{\frac{-sb}{2}} \left(-\frac{b}{2} I_n\left(\frac{sb}{2}\right) + \frac{b}{2} I_n^{'}\left(\frac{sb}{2}\right)\right) \\
&= (-1)^n b e^{\frac{-sb}{2}} \left(I_n^{'}\left(\frac{sb}{2}\right) - I_n\left(\frac{sb}{2}\right)\right) \\
&= (-1)^n b e^{\frac{-sb}{2}} \left(I_{n-1}\left(\frac{sb}{2}\right) - \frac{n}{\frac{sb}{2}} I_n\left(\frac{sb}{2}\right) - I_n\left(\frac{sb}{2}\right)\right) \\
&= (-1)^n b e^{\frac{-sb}{2}} \left(I_{n-1}\left(\frac{sb}{2}\right) - \left(\frac{2n}{sb}+1\right) I_n\left(\frac{sb}{2}\right)\right) \\
&\left(\because I_n^{'}(x) = I_{n-1}(x) - \frac{n}{x} I_n(x)\right).
\end{aligned} \tag{2}$$

Hermite Polynomial. The expansion coefficient for Hermite polynomial is written as

$$c^H_{s,n} = \frac{1}{n!}\left(\frac{-s}{2}\right)^n e^{\frac{s^2}{4}}, \tag{3}$$

and the derivative of $c^H_{s,n}$ with respect to s is computed as

$$\begin{aligned}
\frac{\partial c^H_{s,n}}{\partial s} &= \frac{n}{n!}\left(\frac{-s}{2}\right)^{n-1}\left(\frac{-1}{2}\right) e^{\frac{s^2}{4}} + \frac{1}{n!}\left(\frac{-s}{2}\right)^n e^{\frac{s^2}{4}} \frac{s}{2} \\
&= \frac{1}{n!} e^{\frac{s^2}{4}} \left(\frac{-s}{2}\right)^n \left(\frac{-n}{2}\left(\frac{-s}{2}\right)^{-1} + \frac{s}{2}\right) \\
&= \frac{1}{n!} e^{\frac{s^2}{4}} \left(\frac{-s}{2}\right)^n \left(\frac{n}{s} + \frac{s}{2}\right) \\
&= \frac{1}{n!} e^{\frac{s^2}{4}} \left(\frac{-s}{2}\right)^n \frac{n}{s}\frac{s}{2}\frac{2}{s} + \frac{1}{n!} e^{\frac{s^2}{4}} \left(\frac{-s}{2}\right)^n \frac{s}{2} \\
&= \frac{s e^{\frac{s^2}{4}}}{2n!} \left(\frac{-s}{2}\right)^n \left(\frac{2n}{s^2} + 1\right).
\end{aligned} \tag{4}$$

Laguerre Polynomial. The expansion coefficient for Laguerre polynomial is written as

$$c^L_{s,n} = \frac{s^n}{(s+1)^{n+1}}, \tag{5}$$

and the derivative of $c^L_{s,n}$ with respect to s is calculated as

$$\begin{aligned}
\frac{\partial c^L_{s,n}}{\partial s} &= \frac{ns^{n-1}(s+1)^{n+1} - s^n (n+1)(s+1)^n}{\left\{(s+1)^{n+1}\right\}^2} \\
&= \frac{ns^{n-1}(s+1)^{n+1} - s^n (n+1)(s+1)^n}{(s+1)^{2n}(s+1)^2} \\
&= \frac{ns^{n-1}(s+1) - s^n (n+1)}{(s+1)^{n}(s+1)^2} \\
&= \frac{ns^n + ns^{n-1} - ns^n - s^n}{(s+1)^{n+2}} \\
&= \frac{ns^{n-1} - s^n}{(s+1)^{n+2}} \\
&= \frac{\left(\frac{n}{s} - 1\right) s^n}{(s+1)^{n+2}} \\
&= \frac{s^{n-1}(n-s)}{(s+1)^{n+2}}.
\end{aligned} \tag{6}$$

2. Proofs of Lemma

Lemma 1. *Consider an orthogonal polynomial P_n over interval $[a,b]$ with inner product $\int_a^b P_n(\lambda)P_k(\lambda)w(\lambda)d\lambda = \delta_{nk}$, where $w(\lambda)$ is the weight function. If P_n expands the heat kernel, the expansion coefficients $c_{s,n}$ with respect to* **s** *are differentiable and $\frac{\partial c_{s,n}}{\partial s} = -\int_a^b \lambda e^{-\mathbf{s}\lambda} P_n(\lambda)w(\lambda)d\lambda$.*

Proof. From (Huang et al. 2020) and Eq. (4) in the main paper, the exponential weight of the heat kernel can be expanded by polynomials P_n and coefficients $c_{\mathbf{s},n}$ as

$$e^{-\mathbf{s}\lambda} = \sum_{n=0}^{\infty} c_{\mathbf{s},n} P_n(\lambda). \tag{7}$$

Multiplying both sides of Eq. (7) by $P_k(\lambda)w(\lambda)$, and taking the integral over λ from a to b,

$$\int_a^b e^{-\mathbf{s}\lambda} P_k(\lambda)w(\lambda)d\lambda = \int_a^b \sum_{n=0}^{\infty} c_{\mathbf{s},n} P_n(\lambda)P_k(\lambda)w(\lambda)d\lambda. \tag{8}$$

Since P_n is an orthogonal polynomial and satisfies the inner product equation, $\int_a^b P_n(\lambda)P_k(\lambda)w(\lambda)d\lambda = \delta_{nk}$, so the right-hand side of Eq. (8) follows as

$$\begin{aligned}
&\int_a^b \sum_{n=0}^{\infty} c_{\mathbf{s},n} P_n(\lambda)P_k(\lambda)w(\lambda)d\lambda \\
&= \sum_{n=0}^{\infty} c_{\mathbf{s},n} \int_a^b P_n(\lambda)P_k(\lambda)w(\lambda)d\lambda \\
&= \sum_{n=0}^{\infty} c_{\mathbf{s},n}\delta_{nk}.
\end{aligned} \tag{9}$$

δ_{nk} in Eq. (9) is equal to one when $n = k$, otherwise zero. So, $c_{\mathbf{s},n}$ can be expressed as

$$c_{\mathbf{s},n} = \int_a^b e^{-\mathbf{s}\lambda} P_n(\lambda)w(\lambda)d\lambda. \tag{10}$$

From the Eq. (10), the $\mathbf{s}$ only depends on the exponential term, $e^{-\mathbf{s}\lambda}$. Therefore, the expansion coefficients of the heat kernel are differentiable, and the derivative is

$$\frac{\partial c_{\mathbf{s},n}}{\partial \mathbf{s}} = -\int_a^b \lambda e^{-\mathbf{s}\lambda} P_n(\lambda)w(\lambda)d\lambda. \tag{11}$$

□

Lemma 2. *Let a graph convolution be operated by Eq. (8) (in the main paper), which approximates the convolution with P_n and $c_{s,n}$. If a loss $\mathcal{L}_{\mathit{err}}$ for node-wise classification is defined as cross-entropy between a prediction $\hat{Y} = \sigma(H_K)$, where $\sigma(\cdot)$ is a softmax function, and the true Y, then*

$$\begin{aligned}
\frac{\partial \mathcal{L}_{\mathit{err}}}{\partial s} =&(\hat{Y} - Y) \times \sigma_k^{'}([\sum_{n=0}^{m} c_{s,n}P_n(\hat{L})]H_{k-1}W_k)W_k^{\mathsf{T}} \\
&\times (\sum_{n=0}^{m} P_n(\hat{L})H_{k-1}^{\mathsf{T}} + [\sum_{n=0}^{m} c_{s,n}P_n(\hat{L})]\frac{\partial H_{k-1}}{\partial c_{s,n}})\frac{\partial c_{s,n}}{\partial s}
\end{aligned} \tag{12}$$

where $\sigma_k^{'}$ is the derivative of σ_k.

Proof. Let N be the number of nodes, and J be the number of labels. The loss $\mathcal{L}_{\textbf{err}}$ across all labeled nodes V^L is defined as

$$\mathcal{L}_{\textbf{err}} = -\frac{1}{N}\sum_{i \in V^L}\sum_{j=1}^{J} Y_{ij}\ln\hat{Y}_{ij}. \tag{13}$$

The derivative of $\mathcal{L}$ can be calculated with respect to scale $\mathbf{s}$ of $c_{\mathbf{s},n}$ via chain rule as,

$$\frac{\partial \mathcal{L}_{\textbf{err}}}{\partial \mathbf{s}} = \frac{\partial \mathcal{L}_{\textbf{err}}}{\partial H_K}\frac{\partial H_K}{\partial c_{\mathbf{s},n}}\frac{\partial c_{\mathbf{s},n}}{\partial \mathbf{s}}. \tag{14}$$

Since we use softmax to produce pseudo-probability, the $\frac{\partial \mathcal{L}_{\textbf{err}}}{\partial H_K}$ is derived as

$$\frac{\partial \mathcal{L}_{\textbf{err}}}{\partial H_K} = \hat{Y} - Y. \tag{15}$$

From Eq. (8) in the main paper, the $\frac{\partial H_K}{\partial c_{\mathbf{s},n}}$ becomes

$$\begin{aligned}
\frac{\partial H_K}{\partial c_{\mathbf{s},n}} =& \sigma_k^{'}([\sum_{n=0}^{m} c_{\mathbf{s},n}P_n(\hat{L})]H_{k-1}W_k) \times W_k^{\mathsf{T}} \\
&\times (\sum_{n=0}^{m} P_n(\hat{L})H_{k-1}^{\mathsf{T}} + [\sum_{n=0}^{m} c_{\mathbf{s},n}P_n(\hat{L})]\frac{\partial H_{k-1}}{\partial c_{\mathbf{s},n}})
\end{aligned} \tag{16}$$

where the derivative of H_k is recursively defined with H_{k-1} along the hidden layer. Using chain rule with Eq. (15), (16) and $\frac{\partial c_{\mathbf{s},n}}{\partial \mathbf{s}}$ that depends on the type of polynomial, the gradient on $\mathcal{L}_{\textbf{err}}$ for node classification is written as

$$\begin{aligned}
\frac{\partial \mathcal{L}_{\textbf{err}}}{\partial \mathbf{s}} =&(\hat{Y} - Y) \times \sigma_k^{'}([\sum_{n=0}^{m} c_{\mathbf{s},n}P_n(\hat{L})]H_{k-1}W_k)W_k^{\mathsf{T}} \\
&\times (\sum_{n=0}^{m} P_n(\hat{L})H_{k-1}^{\mathsf{T}} + [\sum_{n=0}^{m} c_{\mathbf{s},n}P_n(\hat{L})]\frac{\partial H_{k-1}}{\partial c_{\mathbf{s},n}})\frac{\partial c_{\mathbf{s},n}}{\partial \mathbf{s}}.
\end{aligned} \tag{17}$$

□

Lemma 3. *Let H_k from Eq. (8) (in the main paper) be a graph convolution operation, which is operated by the heat kernel with polynomial P_n and coefficients $c_{s,n}$. If a loss $\mathcal{L}_{\mathit{err}}$ for classifying graph-wise label is defined as cross-entropy between a prediction $\hat{Y} = \sigma(H_R)$, where $\sigma(\cdot)$ is a softmax function, and the true Y, then*

$$\begin{aligned}
\frac{\partial \mathcal{L}_{\mathit{err}}}{\partial s} =&(\hat{Y} - Y) \times \frac{\partial H_R}{\partial H_K} \times \sigma_k^{'}([\sum_{n=0}^{m} c_{s,n}P_n(\hat{L})]H_{k-1}W_k)W_k^{\mathsf{T}} \\
&\times (\sum_{n=0}^{m} P_n(\hat{L})H_{k-1}^{\mathsf{T}} + [\sum_{n=0}^{m} c_{s,n}P_n(\hat{L})]\frac{\partial H_{k-1}}{\partial c_{s,n}})\frac{\partial c_{s,n}}{\partial s}
\end{aligned} \tag{18}$$

where $\sigma_k^{'}$ is the derivative of σ_k.

Proof. Let T be the sample size, and J be the set of class labels. As in the node classification, the loss $\mathcal{L}_{\textbf{err}}$ is defined with cross-entropy as

$$\mathcal{L}_{\textbf{err}} = -\frac{1}{T}\sum_{t=1}^{T}\sum_{j \in J} Y_{tj}\ln\hat{Y}_{tj} \tag{19}$$

Also, the derivative of $\mathcal{L}$ can be calculated in terms of scale s of $c_{s,n}$ via chain rule. As we use an additional MLP for graph classification, i.e., Eq. (15) in the main paper, the gradient of $\mathcal{L}_{\textbf{err}}$ along H_R must be computed and plugged into Eq. (12) as

$$\frac{\partial \mathcal{L}_{\textbf{err}}}{\partial \mathbf{s}} = \frac{\partial \mathcal{L}_{\textbf{err}}}{\partial H_R}\frac{\partial H_R}{\partial H_K}\frac{\partial H_K}{\partial c_{\mathbf{s},n}}\frac{\partial c_{\mathbf{s},n}}{\partial \mathbf{s}}. \tag{20}$$

Since the activation function of output layer is a softmax, the $\frac{\partial \mathcal{L}_{\textbf{err}}}{\partial H_R}$ are the same as Eq. (15),

$$\frac{\partial \mathcal{L}_{\textbf{err}}}{\partial H_R} = \hat{Y} - Y. \tag{21}$$

The $\frac{\partial H_R}{\partial H_k}$ varies depending on the choice of H_R, and the $\frac{\partial H_k}{\partial c_{\mathbf{s},n}}$ is the same as Eq. (16),

$$\begin{aligned}
\frac{\partial H_K}{\partial c_{\mathbf{s},n}} =& \sigma_k^{'}([\sum_{n=0}^{m} c_{\mathbf{s},n}P_n(\hat{L})]H_{k-1}W_k) \times W_k^{\mathsf{T}} \\
&\times (\sum_{n=0}^{m} P_n(\hat{L})H_{k-1}^{\mathsf{T}} + [\sum_{n=0}^{m} c_{\mathbf{s},n}P_n(\hat{L})]\frac{\partial H_{k-1}}{\partial c_{\mathbf{s},n}}).
\end{aligned} \tag{22}$$

Table 1: Node classification accuracy (%) on Amazon Computers, Amazon Photo, and Coauthor CS. The number is the best accuracy from 10 replicated experiments like (Luo et al. 2022), and the best results are in **bold** except *Exact* method.

Model	**Amazon Computer**	**Amazon Photo**	**Coauthor CS**
MLP (3-layers)	84.63	91.96	95.63
GCN	90.49	93.91	93.32
3ference	90.74	95.05	95.99
GAT	92.51	95.16	93.70
GDC	92.00	94.57	93.45
GraphHeat	92.62	94.77	93.29
LSAP-C	95.53	97.25	**96.37**
LSAP-H	**96.01**	**97.38**	**96.37**
LSAP-L	93.67	96.66	95.91
Exact	95.34	98.17	97.14

With these components and $\frac{\partial c_{\mathbf{s},n}}{\partial \mathbf{s}}$ that depends on the type of polynomial, the gradient on $\mathcal{L}_{\mathbf{err}}$ is computed as

$$\frac{\partial \mathcal{L}_{\mathbf{err}}}{\partial \mathbf{s}} = (\hat{Y} - Y) \times \frac{\partial H_R}{\partial H_K} \times \sigma_k^{'}([\sum_{n=0}^{m} c_{\mathbf{s},n} P_n(\hat{L})] H_{k-1} W_k) W_k^{\mathsf{T}} \times (\sum_{n=0}^{m} P_n(\hat{L}) H_{k-1}^{\mathsf{T}} + [\sum_{n=0}^{m} c_{\mathbf{s},n} P_n(\hat{L})] \frac{\partial H_{k-1}}{\partial c_{\mathbf{s},n}}) \frac{\partial c_{\mathbf{s},n}}{\partial \mathbf{s}}. \tag{23}$$

□

The $\frac{\partial H_R}{\partial H_K}$ for the MLP used within our framework is given in the following Section.

3. LSAP Architecture

Readout for Graph Classification. The $f_R(\cdot)$ with weights W_R takes H_K from the convolution layers as an input and returns H_R. In our model architecture, H_R is chosen as 2-layer Multi-layer perceptron (MLP) as

$$H_R = \sigma_{R_2}(\sigma_{R_1}(H_K W_{R_1}) W_{R_2}) \tag{24}$$

where R_1 and R_2 correspond to first and second layer for MLP structure, respectively. W_{R_1} and W_{R_2} denote weights and Rectified Linear Unit (ReLU) was used for the non-linear activation functions σ_{R_1} and σ_{R_2} for each layer. To make our model end-to-end trainable, the derivative of H_R with respect to H_K is computed as

$$\frac{\partial H_R}{\partial H_K} = \sigma_{R_2}^{'}(\sigma_{R_1}(H_K W_{R_1}) W_{R_2}) W_{R_2} \sigma_{R_1}^{'}(H_K W_{R_1}) W_{R_1}. \tag{25}$$

4. Experiments

To understand our experiments, we describe datasets of our experiments in detail. We conducted experiments on standard node classification datasets that provide connected and undirected graphs. Cora, Citeseer and Pubmed are constructed as citation networks. The nodes are papers, the edges are citations of one paper to another, the features are bag-of-words representation of papers, and the labels are specified as an academic topic. Amazon Computer and Amazon Photo define co-purchase networks. The nodes are goods, the edges indicate whether two goods are frequently purchased together, the features are bag-of-words representation of product reviews, and the labels are categories of goods. Coauthor CS (Shchur et al. 2018) is a co-authorship network. The nodes are authors, the edges indicate whether two authors co-authored a paper, the features are keywords from papers, and the labels are each author's field of study.

In the main paper, specifically in Table 3, we reported the mean and standard deviation of the results of 10 replicates of the experiment and compared it with the results of *3ference* (Luo et al. 2022). However, since the result of *3ference* is the maximum value among the 10 experimental results, we also wrote the maximum value in the supplementary and compared it, which is in Table 1. Our model showed the best performance compared to the baselines.

In the main experiments, we compared the performances between LSAP and baselines on node classification and graph classification. For node classification, standard benchmarks (Cora, Citeseer and Pubmed) and additional datasets (Amazon Computers, Amazon Photo, and Coauthor CS) were used to validate the performance of LSAP. For graph classification, the ADNI dataset containing a number of subjects with brain network was used to classify AD-specific labels. Hyperparameters should be set so that the model can properly learn data in each experiment. Table 2 shows the detailed parameters of LSAP and baselines for the main experiments.

The trained scales on the brain network classification with Exact and LSAP are visualized in Fig. 1 and Fig. 2 containing additional interpretable results visualized from various views. In Table 3, based on Exact, smallest scales that appear in common across Exact and LSAPs are listed. 7 ROIs were detected in Exact and LSAPs, 6 ROIs were detected in 3 models, and only 2 ROIs were in 2 models for cortical thickness feature. Using FDG, 10 ROIs were detected in every model, 6 ROIS were detected in 3 models, and 2 ROIs were detected in 2 models. In addition, similar ROIs overall showed similar values of scale in all models.

5. Implementation Details

We stacked K=2 heat kernel convolution layers with rectified linear unit (ReLU) as the activation function and softmax function at the output to predict the node or graph classes. The initial scale for every node was set to 2. The n=20 for LSAP-C and LSAP-L, and n=30 for LSAP-H as they empirically demonstrated the best convergence. The same number of hidden nodes in W_k was used within each the dataset, drop-out rate was 0.5, and other hyperparameters such as learning rate for W_k and s were properly tuned to get the best results (given in the supplementary). For the node classification, we used early stopping to stop training when the validation accuracy stopped increasing. For graph classification, the readout function $f_R(\cdot)$ was defined as MLP with 2 layers with 16 hidden nodes (with ReLU), and 5-fold cross validation (CV) was used.

6. Model complexity

Exact method requires full eigendecomposition of Graph Laplacian, which takes $O(N^3)$ where N is the number of

Table 2: Hyperparameters for classification tasks.

Task	Dataset	Model	Hidden Units	Learning Rate	Dropout Rate	Regularization (α)	Scale's Learning Rate (β_s)	LSAP-C (b)
Node Classification	**Cora**	**LSAP** & Exact	64	0.01	0.5	0.1	1	1.48
	Citeseer	**LSAP** & Exact	32	0.01	0.5	1	10	1.50
	Pubmed	**LSAP** & Exact	64	0.1	0.5	1	10	1.65
	Amazon Computer	GAT	32	0.01	0.5	-	-	-
		GDC	32	0.0001	0.5	-	-	-
		GraphHeat	32	0.01	0.5	-	-	-
		LSAP & Exact	32	0.001	0.5	1	1	1.60
	Amazon photo	GAT	32	0.01	0.5	-	-	-
		GDC	16	0.0001	0.5	-	-	-
		GraphHeat	32	0.01	0.5	-	-	-
		LSAP & Exact	32	0.01	0.5	1	10	1.59
	Coauthor CS	GAT	32	0.01	0.5	-	-	-
		GDC	16	0.0001	0.5	-	-	-
		GraphHeat	32	0.01	0.5	-	-	-
		LSAP & Exact	32	0.01	0.5	1	1	1.41
Graph Classification	**Cortical Thickness**	SVM (Linear)	-	-	-	-	-	-
		MLP (2-layers)	16	0.01	0.5	-	-	-
		GCN	16	0.01	0.5	-	-	-
		GAT	16	0.01	0.1	-	-	-
		GDC	16	0.01	0.5	-	-	-
		GraphHeat	32	0.01	0.5	-	-	-
		ADC	16	0.01	0.5	-	-	-
		LSAP & Exact	16	0.01	0.5	1	1	1.20
	FDG	SVM (Linear)	-	-	-	-	-	-
		MLP (2-layers)	16	0.01	0.5	-	-	-
		GCN	16	0.01	0.5	-	-	-
		GAT	16	0.01	0.1	-	-	-
		GDC	16	0.01	0.5	-	-	-
		GraphHeat	16	0.01	0.5	-	-	-
		ADC	16	0.01	0.5	-	-	-
		LSAP & Exact	16	0.01	0.5	1	1	1.20

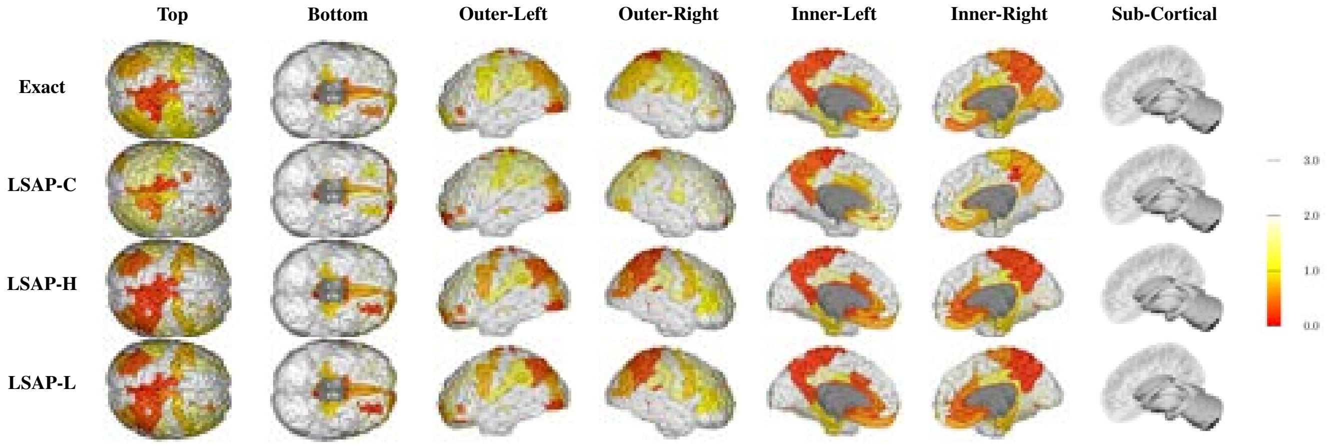

Figure 1: Visualization of learned scales on the cortical and sub-cortical regions of a brain. This visualization shows the scale of each RoI through the classification result using Cortical Thickness feature.

nodes. However, our method does not have this computation and only use graph Laplacian directly to compute polynomial basis, which will be used to construct approximated heat kernel with $|\mathbf{s}| = N$ parameters. Then, as we basically used GCN framework and redefined the convolution with approximated heat kernel, time complexity of construction of heat kernel is simply added to time complexity of GCN.

The time complexity of L-layer GCN is $O(LND^2)$ where D is dimension of features. When we use only first order approximation of heat kernel, time complexity of our process becomes $O(N)$ which is highly marginal compared to the GCN pipeline. The memory complexity of GCN is $O(LND + LD^2)$. Memory complexity of our frameworks takes $O(N^2)$ for construction of polynomial basis, which will be added to memory complexity of GCN, which can be further reduced with polynomial approximation. At the cost of some time and memory in addition to the GCN, we gain significant improvements in both node and graph tasks.

7. Adjustment of the learned scales

Fig. 3 displays the outcomes pertaining to the scale of each ROI in the brain during every quarter interval throughout the complete iteration within node-wise method. In contrast to prior approaches such as GraphHeat (Xu et al. 2020) and ADC (Zhao et al. 2021), which employ a uniform global scale for all regions of interest (ROIs), our proposed method

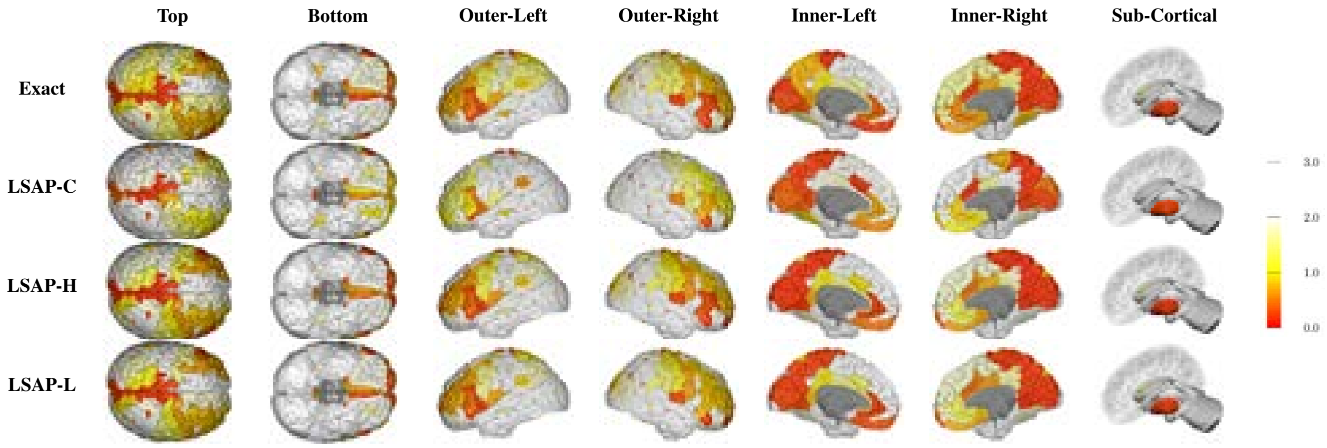

Figure 2: Visualization of learned scales on the cortical and sub-cortical regions of a brain. This visualization shows the scale of each RoI through the classification result using FDG feature.

Table 3: ROIs with the smallest trained scales for AD classification. (L) and (R) denote left and right hemisphere.

Cortical Thickness						FDG					
# of common models	**ROI**	**Exact**	**LSAP-C**	**LSAP-H**	**LSAP-L**	**# of common models**	**ROI**	**Exact**	**LSAP-C**	**LSAP-H**	**LSAP-L**
4	(R) S.pericallosal	0.035	0.097	0.033	0.050	**4**	(L) G&S.paracentral	0.034	0.052	0.049	0.066
	(R) Lat.Fis.ant.Horizont	0.048	0.074	0.039	0.037		(L) G.front.inf.Orbital	0.036	0.071	0.060	0.043
	(L) Lat.Fis.ant.Horizont	0.049	0.083	0.038	0.051		(R) G.precuneus	0.041	0.044	0.034	0.051
	(L) G&S.paracentral	0.059	0.034	0.046	0.044		(R) S.ortibal.med.olfact	0.047	0.078	0.054	0.059
	(L) G&S.occipital.inf	0.062	0.071	0.043	0.055		(R) G.cingul.Post.ventral	0.055	0.056	0.055	0.051
	(R) Lat.Fis.ant.Vertical	0.064	0.108	0.046	0.041		(R) S.oc.temp.lat	0.055	0.065	0.045	0.063
	(L) G.cingul.Post.ventral	0.068	0.127	0.035	0.066		(R) G.oc.temp.med.Lingual	0.055	0.076	0.043	0.040
3	(L) S.suborbital	0.036	-	0.054	0.060		(L) Sub.put	0.058	0.077	0.047	0.060
	(L) S.temporal.inf	0.045	0.129	0.039	0.057		(L) S.postcentral	0.061	0.069	0.060	0.018
	(L) S.occipital.ant	0.050	0.119	-	0.064		(R) G.front.inf.Orbital	0.063	0.093	0.069	0.050
	(R) S.precentral.inf.part	0.054	0.052	-	0.057	**3**	(R) S.intrapariet&P.trans	0.046	0.095	-	0.054
	(R) S.oc.temp.med&Lingual	0.055	-	0.052	0.062		(L) G.cuneus	0.049	-	0.044	0.062
	(L) G.temp.sup.G.T.transv	0.068	0.104	0.049	-		(R) S.temporal.sup	0.049	-	0.028	0.056
2	(R) G.parietal.sup	0.035	-	0.056	-		(R) S.calcarine	0.053	0.059	-	0.047
	(R) Lat.Fis.post	0.049	0.067	-	-		(R) G&S.paracentral	0.055	-	0.051	0.048
	(L) Lat.Fis.ant.Vertical	0.059	-	-	0.040		(R) G.cuneus	0.057	-	0.057	0.066
	(R) S.suborbital	0.064	-	-	0.048	**2**	(R) G&S.cingul.Mid.Ant	0.062	0.041	-	-
							(R) Sub.put	0.065	0.037	-	-
	RMSE for all ROIs	-	**0.5358**	**0.4193**	**0.4072**		**RMSE** for all ROIs	-	**0.5827**	**0.2904**	**0.2899**

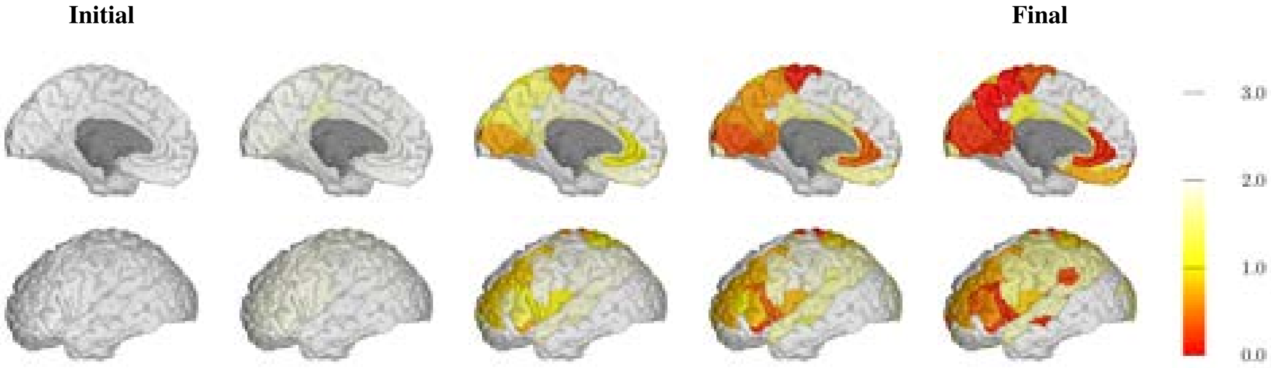

Figure 3: Visualization of adjustment of the trained scales on the cortical regions of a brain at training phase using FDG biomarker. Top: Inner part of left hemisphere, Bottom: Outer part of left hemisphere.

looks for multivariate scales for each individual ROI during the training stage. The initial scales for all ROIs are uniformly set to the same value, and subsequently tuned to align with the corresponding data. As the learning phase progresses from its initial state, several ROIs associated with AD become spotlighted, which are colored by close to red in Fig. 3.

8. Discussions

Reasons for evaluating LSAP on the ADNI data. ADNI dataset was a suitable bench mark for the following reasons: 1) As the performance of node-wise scale training is sufficiently validated on node-classification with well-curated benchmarks, we wanted to run experiments on a real dataset with a practical application for scientific discovery such as Neuroimaging. 2) Brain has a common structure that can be translated to registered ROIs (i.e., nodes), where the results from LSAP can be clinically interpreted and convey interesting results to the readers. 3) ADNI study provide relatively large sample-size, and we have performed experiments with 5-fold cross validation to ensure unbiased results even though we are using only one dataset.

Limitations of LSAP. As LSAP needs to localize and update the scale for each node, training takes longer compared to conventional GNN that perform uniform convolution across all nodes, but this issue is well-addressed with our training on the scales within approximation. Another limitation is that LSAP operates with fixed set of nodes for graph classification. However, such a setting has an advantage with LSAP as we learn node-wise scale which yield interpretable results that may lead to scientific discoveries.

References

Huang, S.-G.; Lyu, I.; Qiu, A.; and Chung, M. K. 2020. Fast polynomial approximation of heat kernel convolution on manifolds and its application to brain sulcal and gyral graph pattern analysis. *IEEE transactions on medical imaging*, 39(6): 2201–2212.

Luo, Y.; Luo, G.; Yan, K.; and Chen, A. 2022. Inferring from References with Differences for Semi-Supervised Node Classification on Graphs. *Mathematics*, 10(8): 1262.

Shchur, O.; Mumme, M.; Bojchevski, A.; and Günnemann, S. 2018. Pitfalls of graph neural network evaluation. *arXiv preprint arXiv:1811.05868*.

Xu, B.; Shen, H.; Cao, Q.; Cen, K.; and Cheng, X. 2020. Graph convolutional networks using heat kernel for semi-supervised learning. *arXiv preprint arXiv:2007.16002*.

Zhao; et al. 2021. Adaptive diffusion in graph neural networks. *Advances in NeurIPS*, 34: 23321–23333.