A Training Configurations

Data statistics. We summarize the data statistics in our experiments in Table 1.

Table 1: Dataset statistics of the three learning tasks in our experiments.

Learning Task	Dataset	Nodes	Edges	Train/Dev/Test Nodes	Split Ratio (%)
Semi-supervised	Cora	2,708	5,429	140/500/1,000	5.2/18.5/36.9
	Citeseer	3,327	4,732	120/500/1,000	3.6/15.0/30.1
	Pubmed	19,717	44,338	60/500/1,000	0.3/2.5/5.1
Fully-supervised	Cora	2,708	5,429	1624/541/543	60.0/20.0/20.0
	Citeseer	3,327	4,732	1996/665/666	60.0/20.0/20.0
	Pubmed	19,717	44,338	11830/3943/3944	60.0/20.0/20.0
Inductive (large-scale)	Reddit	233K	11.6M	152K/24K/55K	65.2/10.3/23.6

Training hyper-parameters. For both fully and semi-supervised node classification tasks on the citation networks, Cora, Citeseer and Pubmed, we train our DGC following the hyper-parameters in SGC [5]. Specifically, we train DGC for 100 epochs using Adam [2] with learning rate 0.2. For weight decay, as in SGC, we tune this hyperparameter on each dataset using hyperopt [1] for 10,000 trails. For the large-scale inductive learning task on the Reddit network, we also follow the protocols of SGC [5], where we use L-BFGS [3] optimizer for 2 epochs with no weight decay.

B Omitted Proofs

B.1 Proof of Theorem 1

Theorem 1. *The heat kernel* $\mathbf{H}_t = e^{-t\mathbf{L}}$ *admits the following eigen-decomposition,*

$$\mathbf{H}_t = \mathbf{U} \begin{pmatrix} e^{-\lambda_1 t} & 0 & \cdots & 0 \\ 0 & e^{-\lambda_2 t} & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & e^{-\lambda_n t} \end{pmatrix} \mathbf{U}^\top.$$

As a result, with $\lambda_i \geq 0$*, we have*

$$\lim_{t\to\infty} e^{-\lambda_i t} = \begin{cases} 0, & \text{if } \lambda_i > 0 \\ 1, & \text{if } \lambda_i = 0 \end{cases}, \; i = 1, \ldots, n. \tag{1}$$

Proof. With the eigen-decomposition of the Laplacian $\mathbf{L} = \mathbf{U}\mathbf{\Lambda}\mathbf{U}^\top$, the heat kernel can be written equivalently as

$$\mathbf{H}_t = e^{-t\mathbf{L}} = \sum_{k=0}^{\infty} \frac{t^k}{k!}(-\mathbf{L})^k = \sum_{k=0}^{\infty} \frac{t^k}{k!}\left[\mathbf{U}(-\mathbf{\Lambda})\mathbf{U}^\top\right]^k = \mathbf{U}\left[\sum_{k=0}^{\infty} \frac{t^k}{k!}(-\mathbf{\Lambda})^k\right]\mathbf{U}^T = \mathbf{U}e^{-t\mathbf{\Lambda}}\mathbf{U}^T, \tag{2}$$

which corresponds to the eigen-decomposition of the heat kernel with eigen-vectors in the orthogonal matrix $\mathbf{U}$ and eigven-values in the diagonal matrix $e^{-t\mathbf{\Lambda}}$. Now it is easy to see the limit behavior of the heat kernel as $t \to \infty$ from the spectral domain. □

B.2 Proof of Theorem 2

Theorem 2. *For the general initial value problem*

$$\begin{cases} \frac{d\mathbf{X}_t}{dt} & = -\mathbf{L}\mathbf{X}_t, \\ \mathbf{X}_0 & = \mathbf{X}, \end{cases} \tag{3}$$

with any finite terminal time T, the numerical error of the forward Euler method

$$\hat{\mathbf{X}}_T^{(K)} = \left(\mathbf{I} - \frac{T}{K}\mathbf{L}\right)^K \mathbf{X}_0. \tag{4}$$

with K propagation steps can be upper bounded by

$$\|\mathbf{e}_T^{(K)}\| \leq \frac{T\|\mathbf{L}\|\|\mathbf{X}_0\|}{2K}\left(e^{T\|\mathbf{L}\|} - 1\right). \tag{5}$$

Proof. Consider a general Euler forward scheme for our initial problem

$$\hat{\mathbf{X}}^{(k+1)} = \hat{\mathbf{X}}^{(k)} - h\mathbf{L}\hat{\mathbf{X}}_t, \quad k = 0, 1, \ldots, K-1, \quad \mathbf{X}^{(0)} = \mathbf{X}, \tag{6}$$

where $\hat{\mathbf{X}}^{(k)}$ denotes the approximated $\mathbf{X}$ at step k, h denotes the step size and the terminal time $T = Kh$. We denote the global error at step k as

$$\mathbf{e}_k = \mathbf{X}^{(k)} - \hat{\mathbf{X}}^{(k)}, \tag{7}$$

and the truncation error of the Euler forward finite difference (Eqn. (6)) at step k as

$$\mathbf{T}^{(k)} = \frac{\mathbf{X}^{(k+1)} - \mathbf{X}^{(k)}}{h} + \mathbf{L}\mathbf{X}^{(k)}. \tag{8}$$

We continue by noting that Eqn. (8) can be written equivalently as

$$\mathbf{X}^{(k+1)} = \mathbf{X}^{(k)} + h\left(\mathbf{T}^{(k)} - \mathbf{L}\mathbf{X}^{(k)}\right). \tag{9}$$

Taking the difference of Eqn. (9) and (6), we have

$$\mathbf{e}^{(k+1)} = (1 - h\mathbf{L})\mathbf{e}^{(k)} + h\mathbf{T}^{(k)}, \tag{10}$$

whose norm can be upper bounded as

$$\left\|\mathbf{e}^{(k+1)}\right\| \leq (1 + h\|\mathbf{L}\|)\left\|\mathbf{e}^{(k)}\right\| + h\left\|\mathbf{T}^{(k)}\right\|. \tag{11}$$

Let $M = \max_{0\leq k\leq K-1}\|\mathbf{T}^{(k)}\|$ be the upper bound on global truncation error, we have

$$\left\|\mathbf{e}^{(k+1)}\right\| \leq (1 + h\|\mathbf{L}\|)\left\|\mathbf{e}^{(k)}\right\| + hM. \tag{12}$$

By induction, and noting that $1 + h\|\mathbf{L}\| \leq e^{h\|\mathbf{L}\|}$ and $\mathbf{e}^{(0)} = \mathbf{X}^{(0)} - \hat{\mathbf{X}}^{(0)} = \mathbf{0}$, we have

$$\left\|\mathbf{e}^{(K)}\right\| \leq \frac{M}{\|\mathbf{L}\|}\left[(1 + h\|\mathbf{L}\|)^n - 1\right] \leq \frac{M}{\|\mathbf{L}\|}\left(e^{Kh\|\mathbf{L}\|} - 1\right). \tag{13}$$

Now we note that $\frac{d\mathbf{X}^{(k)}}{dt} = -\mathbf{L}\mathbf{X}^{(k)}$ and applying Taylor's theorem, there exists $\delta \in [nh, (k+1)h]$ such that the truncation error $\mathbf{T}^{(k)}$ in Eqn. (8) follows

$$\mathbf{T}^{(k)} = \frac{1}{2h}\mathbf{L}^2\mathbf{X}_\delta. \tag{14}$$

Thus the truncation error can be bounded by

$$\left\|\mathbf{T}^{(k)}\right\| = \frac{1}{2h}\|\mathbf{L}\|^2\|\mathbf{X}_\delta\| \leq \frac{1}{2h}\|\mathbf{L}\|^2\|\mathbf{X}_0\|, \tag{15}$$

because

$$\|\mathbf{X}_\delta\| = \left\|e^{-\delta\mathbf{L}}\mathbf{X}_0\right\| \leq \|\mathbf{X}_0\|, \ \forall\delta \geq 0. \tag{16}$$

Together with the fact $T = Kh$, we have

$$\left\|\mathbf{e}^{(K)}\right\| \leq \frac{\|\mathbf{L}\|^2\|\mathbf{X}_0\|}{2h\|\mathbf{L}\|}\left(e^{Kh\|\mathbf{L}\|} - 1\right) = \frac{T\|\mathbf{L}\|\|\mathbf{X}_0\|}{2K}\left(e^{T\|\mathbf{L}\|} - 1\right), \tag{17}$$

which completes the proof. □

B.3 Proof of Theorem 3

For the ground-truth data generation process

$$\mathbf{Y} = \mathbf{X}_c \mathbf{W}_c + \sigma_y \boldsymbol{\varepsilon}_y,\ \boldsymbol{\varepsilon}_y \sim \mathcal{N}(\mathbf{0}, \mathbf{I}); \tag{18}$$

together with the data corruption process,

$$\frac{d\widetilde{\mathbf{X}}_t}{dt} = \mathbf{L}\widetilde{\mathbf{X}}_t, \text{ where } \widetilde{\mathbf{X}}_0 = \mathbf{X}_c \text{ and } \widetilde{\mathbf{X}}_{T^*} = \mathbf{X}. \tag{19}$$

and the final state $\mathbf{X}$ denote the observed data. Then, we have the following bound its population risks.

Theorem 3. *Denote the population risk of the ground truth regression problem with weight* $\mathbf{W}$ *as*

$$R(\mathbf{W}) = \mathbb{E}_{p(\mathbf{X}_c, \mathbf{Y})} \left\| \mathbf{Y} - \mathbf{X}_c \mathbf{W} \right\|^2. \tag{20}$$

and that of the corrupted regression problem as

$$\hat{R}(\mathbf{W}) = \mathbb{E}_{p(\hat{\mathbf{X}}, \mathbf{Y})} \left\| \mathbf{Y} - [\mathbf{S}^{(\hat{T}/K)}]^K \mathbf{X}\mathbf{W} \right\|^2. \tag{21}$$

Supposing that $\mathbb{E}\|\mathbf{X}_c\|^2 = M < \infty$*, we have the following upper bound on the latter risk:*

$$\hat{R}(\mathbf{W}) \leq R(\mathbf{W}) + 2\left\|\mathbf{W}\right\|^2 \left(\mathbb{E} \left\| \mathbf{e}_{\hat{T}}^{(K)} \right\|^2 + M \left\| e^{T^\star \mathbf{L}} \right\|^2 \cdot \left\| e^{-T^\star \mathbf{L}} - e^{-\hat{T}\mathbf{L}} \right\|^2 \right).$$

Proof. Given the fact that $\mathbf{X}_c = \tilde{\mathbf{X}}_0 = e^{-T^*\mathbf{L}}\mathbf{X}$, we can decompose the corrupted population risk as follows

$$\begin{aligned}
&\hat{R}(\mathbf{W}) = \mathbb{E}_{p(\hat{\mathbf{X}}, \mathbf{Y})} \left\| \mathbf{Y} - \left[\mathbf{S}^{(\hat{T}/K)}\right]^K \mathbf{X}\mathbf{W} \right\|^2 \\
=&\mathbb{E}_{p(\mathbf{X}, \mathbf{Y})} \left\| \mathbf{Y} - \mathbf{X}_c\mathbf{W} + \left(e^{-T^\star \mathbf{L}} - \left[\mathbf{S}^{(\hat{T}/K)}\right]^K \right) \mathbf{X}\mathbf{W} \right\|^2 \\
\leq&\mathbb{E}_{p(\mathbf{X}, \mathbf{Y})} \left\| \mathbf{Y} - \mathbf{X}_c\mathbf{W} \right\|^2 + + \left\|\mathbf{W}\right\|^2 \mathbb{E}_{p(\mathbf{X}, \mathbf{Y})} \left\| \left(\left[e^{-\hat{T}\mathbf{L}} - \mathbf{S}^{(\hat{T}/K)} \right]^K \right) \mathbf{X} + \left(e^{-T^\star \mathbf{L}} - e^{-\hat{T}\mathbf{L}} \right) \mathbf{X} \right\|^2 \\
\leq&\mathbb{E}_{p(\mathbf{X}, \mathbf{Y})} \left\| \mathbf{Y} - \mathbf{X}_c\mathbf{W} \right\|^2 + \left\|\mathbf{W}\right\|^2 \mathbb{E}_{p(\mathbf{X}, \mathbf{Y})} \left\| \mathbf{e}_{\hat{T}}^{(K)} + \left(e^{-T^\star \mathbf{L}} - e^{-\hat{T}\mathbf{L}} \right) e^{T^\star \mathbf{L}} \mathbf{X}_c \right\|^2 \\
\leq&R(\mathbf{W}) + 2\left\|\mathbf{W}\right\|^2 \left(\mathbb{E} \left\| \mathbf{e}_{\hat{T}}^{(K)} \right\|^2 + M \left\| e^{T^\star \mathbf{L}} \right\|^2 \left\| e^{-T^\star \mathbf{L}} - e^{-\hat{T}\mathbf{L}} \right\|^2 \right),
\end{aligned} \tag{22}$$

which completes the proof. □

C Further Comparison of SIGN and DGC

Here, we provide a more detailed comparison of DGC and SIGN [4]. In particular, the SIGN model is

$$Y = \xi(Z\Omega), \quad Z = \sigma([X\Theta_0, A_1 X\Theta_1, \ldots, A_r X\Theta_r]),$$

where σ, ξ are nonlinearities, $A_k = A^k$ is the k-hop propagation matrix, and Θ_k, Ω are weight matrices. Our DGC-Euler model takes the form

$$Y = \xi(X^{(K)}\Omega), \quad X^{(k)} = (1 - T/K)X^{(k-1)} + (T/K) \cdot AX^{(k-1)}, \quad k = 2, \ldots, K.$$

The two models 1) both apply all feature propagation before the classification model and so that can 2) both pre-process the propagation matrix and save it for later training. Nevertheless, there are several critical differences between SIGN and DGC:

- **Linear v.s. Nonlinear.** DGC is a linear model, while SIGN is nonlinear.

- **Multi-scale SGC (SIGN) v.s. single-scale continuous diffusion (DGC).** SIGN is a multi-scale method that extracts every possible scale ($A^r X, r = 0, 1, \dots$) for feature propagation. Thus, SIGN resembles a multi-scale SGC, but still inherits some of the limitations of SGC, e.g. a fixed step size $\Delta t = 1$. On the contrary, the goal of DGC is to find the optimal tradeoff between under-smoothing and over-smoothing with a flexible choice of T (real-numbered) and fine-grained integration (K), so it only uses a single-scale propagation kernel (Eq. 12).
- **Model Size.** As a result, the model size of (linear) SIGN is proportional to the number of scales r, while the model size of DGC is independent of T and K.

Overall, we can see that the two are closely related. Below, we further compare DGC with SIGN in terms of both their performance as well as their computational efficiency.

C.1 Fine-grained Performance Comparison

To take a closer look at the difference between the two methods, we compare the two methods with the same terminal time T.

Setup. We conduct the experiments on the Cora dataset (semi-supervised). We re-produce SIGN as it has not reported results on these datasets. For comparison, we follow the same protocol of SGC, using the same optimizer, learning rate, training epochs; and (automatically) tune the weight decay and propagation steps (K or r) at each terminal time T.

Table 2: Comparison of test accuracy (%) with different time T ranging from 1 to 6.

Methods	1	2	3	4	5	5.3	6
SGC	72.4	80.5	79.2	81.0	78.8	N/A	80.5
SIGN	72.4	77.3	78.9	80.6	81.3	N/A	81.7
DGC	78.0	81.9	82.5	83.0	83.1	83.3	81.5

Results. As shown in Table 2, we have the following findings:

- DGC still outperforms SIGN for $T \leq 5$, while being slightly worse at $T = 6$ due to over-smoothing;
- DGC can flexibly choose a real-numbered terminal time, e.g., $T = 5.3$, to find the best tradeoff between under-smoothing and over-smoothing (83.3 acc), while the terminal time of SIGN and SGC has to be an integer;
- Single-stage methods (SGC & DGC) have bigger advantages at earlier time, while SIGN can surpass SGC at later stages by aggregating multi-scale information.

The empirical results show that although useful, multi-scale techniques cannot fully solve the limitations of SGC, while DGC can overcome these limitations by decoupling T and K.

C.2 Computation Time

Here, we further compare the explicit training time. We also include SIGN with a different number of scales (r) as a baseline. The experiments are conducted on the same platform.

Table 3: Comparison of training time on the Pubmed dataset.

Method	Preprocessing Time	Training Time	Total Time
SGC / DGC ($K = 2$)	3.8 ms	61.5 ms	65.3 ms
SIGN ($r = 2$)	5.9 ms	78.7 ms	84.6 ms
DGC ($K = 100$)	169.2 ms	55.8 ms	225.0 ms
SIGN ($r = 100$)	2.4 s	106.9 ms	2.6 s
GCN	0	17.0 s	17.0 s

We note that the comparison of DGC ($K = 100$) v.s. SGC ($K = 2$) is merely designed to show how the extra propagation steps do not contribute much to the total time. Remarkably, it does not mean that DGC takes 100 steps at all settings. Sometimes, a few steps ($K < 10$) are enough to attain the best performance. From Table 3, we have the following findings: 1) both SIGN and DGC can be much faster than GCN by pre-processing the propagation kernels; 2) DGC is still faster than SIGN with the same propagation steps by being a single-scale method.

Table 4: Comparison of training time on the PubMed dataset.

K or r	2	10	20	50	100
DGC	3.8 ms	17.5 ms	34.9 ms	86.5 ms	169.2 ms
SIGN	5.9 ms	20.5 ms	51.5 ms	162.3 ms	2.4 s

To better understand the difference in computation time between DGC and SIGN, we note that SIGN is a multi-scale method, and it stores every intermediate scale of A, i.e., $[A^1, A^2, \ldots, A^r]$ in the preprocessing stage with a memory complexity $O(r)$. On the contrary, DGC is a single-scale method and only needs to store one single propagation matrix $S^{(K)}$, so its memory complexity is $O(1)$. In modern deep learning frameworks (PyTorch in our case), it takes time to keep expanding the working GPU memory due to the copy operation. As a result, when r or K is very large (like 100), SIGN can be much more memory-intensive than SGC ($100\times$). This results in a large difference in the preprocessing time between SIGN and DGC, as shown above in Table 4. There might be some tricky ways to optimize the pipeline, but here we stick to the vanilla (also official) implementation for a fair comparison.

References

[1] James Bergstra, Brent Komer, Chris Eliasmith, Dan Yamins, and David D Cox. Hyperopt: a python library for model selection and hyperparameter optimization. *Computational Science & Discovery*, 8(1):014008, 2015. 1

[2] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *ICLR*, 2015. 1

[3] Dong C Liu and Jorge Nocedal. On the limited memory BFGS method for large scale optimization. *Mathematical Programming*, 45(1):503–528, 1989. 1

[4] Emanuele Rossi, Fabrizio Frasca, Ben Chamberlain, Davide Eynard, Michael Bronstein, and Federico Monti. SIGN: Scalable inception graph neural networks. *arXiv preprint arXiv:2004.11198*, 2020. 3

[5] Felix Wu, Tianyi Zhang, Amauri Holanda de Souza Jr, Christopher Fifty, Tao Yu, and Kilian Q Weinberger. Simplifying graph convolutional networks. *ICML*, 2019. 1