DEEPGRAND: DEEP GRAPH NEURAL DIFFUSION

Anonymous authors
Paper under double-blind review

ABSTRACT

We propose the Deep Graph Neural Diffusion (DeepGRAND), a class of continuous-depth graph neural networks based on the diffusion process on graphs. DeepGRAND leverages a data-dependent scaling term and a perturbation to the graph diffusivity to make the real part of all eigenvalues of the diffusivity matrix become negative, which ensures two favorable theoretical properties: (i) the node representation does not exponentially converge to a constant vector as the model depth increases, thus alleviating the over-smoothing issue; (ii) the stability of the model is guaranteed by controlling the norm of the node representation. Compared to the baseline GRAND, DeepGRAND mitigates the accuracy drop-off with increasing depth and improves the overall accuracy of the model. We empirically corroborate the advantage of DeepGRAND over many existing graph neural networks on various graph deep learning benchmark tasks.

1 INTRODUCTION

Graph neural networks (GNNs) and machine learning on graphs (Bronstein et al., 2017; Scarselli et al., 2008) have been successfully applied in a wide range of applications including physical modeling (Duvenaud et al., 2015b; Gilmer et al., 2017; Battaglia et al., 2016), recommender systems (Monti et al., 2017; Ying et al., 2018), and social networks (Zhang & Chen, 2018; Qiu et al., 2018). Recently, more advanced GNNs have been developed to further improve the performance of the models and extend their application beyond machine learning, which include the graph convolutional networks (GCNs) (Kipf & Welling, 2017), ChebyNet (Defferrard et al., 2016), GraphSAGE (Hamilton et al., 2017), neural graph finger prints (Duvenaud et al., 2015a), message passing neural network (Gilmer et al., 2017), graph attention networks (GATs) (Veličković et al., 2018), and hyperbolic GNNs (Liu et al., 2019). A well-known problem of GNNs is that the performance of the model decreases significantly with increasing depth. This phenomena is a common plight of most GNN architectures and referred to as the oversmoothing issue of GNNs (Li et al., 2018; Oono & Suzuki, 2020; Chen et al., 2020).

1.1 MAIN CONTRIBUTIONS AND OUTLINE

In this paper, we propose Deep Graph Neural Diffusion (DeepGRAND), a class of continuous-depth graph neural networks based on the diffusion process on graphs that improves on various aspects of the baseline Graph Neural Diffusion (GRAND) (Chamberlain et al., 2021b). At its core, DeepGRAND introduces a data-dependent scaling term and a perturbation to the diffusion dynamic. With this design, DeepGRAND attains the following advantages:

1. DeepGRAND inherits the diffusive characteristic of GRAND while significantly mitigates the over-smoothing issue.
2. DeepGRAND achieves remarkably greater performance than existing GNNs and other GRAND variants when fewer nodes are labelled as training data, meriting its use in low-labelling rates situations.
3. Feature representation under the dynamic of DeepGRAND is guaranteed to remain bounded, ensuring numerical stability.

Organization. In Section 2, we give a concise description of the over-smoothing issue of general graph neural networks and the GRAND architecture. A rigorous treatment and further discussions

showing the inherent over-smoothing issue in GRAND is given in Section 3. The formulation for DeepGRAND is given in Section 4, where we also provide theoretical guarantees on the stability and ability to mitigate over-smoothing of DeepGRAND. Finally, we demonstrate the practical advantages of DeepGRAND in Section 5, showing reduced accuracy drop-off at higher depth and improved overall accuracy when compared to variants of GRAND and other popular GNNs.

1.2 Related work

Neural ODEs. Chen et al. (2018) introduced Neural ODEs, a class of continuous-depth neural networks with inherent residual connections. Follow-up works extended this framework through techniques such as augmentation (Dupont et al., 2019), regularization (Finlay et al., 2020), momentum (Xia et al., 2021). Continuous depth GNN was first proposed by Xhonneux et al. (2020).

Over-smoothing. The over-smoothing issue of GNNs was first recognized by Li et al. (2018). Many recent works have attempted to address this problem (Oono & Suzuki, 2020; Cai & Wang, 2020; Zhao & Akoglu, 2020). A connection between the over-smoothing issue and the homophily property of graphs was also recognized by Yan et al. (2021).

PDE-inspired graph neural networks. Partial differential equations (PDEs) are ubiquitous in modern mathematics, arising naturally in both pure and applied mathematical research. A number of recent works in graph machine learning has taken inspiration from classical PDEs, including: diffusion equation (Chamberlain et al., 2021b; Thorpe et al., 2022), wave equation (Eliasof et al., 2021), Beltrami flow (Chamberlain et al., 2021a), etc. Such approaches have led to interpretable architectures, and provide new solutions to traditional problems in GNNs such as over-smoothing (Rusch et al., 2022) and over-squashing (Topping et al., 2022).

2 Background

Notation. Let $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ denote a graph, where $\mathcal{V}$ is the vertex set with $|\mathcal{V}| = n$ and $\mathcal{E}$ is the edge set with $|\mathcal{E}| = e$. For a vertex u in $\mathcal{V}$, denote $\mathcal{N}_u$ the set of vertices neighbor to $u \in \mathcal{G}$. We denote d as the dimensionality of the features, i.e, number of features for each node. Following the convention of Goodfellow et al. (2016), we denote scalars by lower- or upper- case letters and vectors and matrices by lower- and upper-case boldface letters, respectively.

2.1 Graph neural networks and the over-smoothing issue

As noted by Bronstein et al. (2021), the vast majority of the literature on graph neural networks can be derived from just three basic flavours: convolutional, attentional and message-passing. The cornerstone architectures for these flavours are GCN (Kipf & Welling, 2017), GAT (Veličković et al., 2018) and MPNN (Gilmer et al., 2017), the last of which forms an overarching design over the other two. An update rule for all message passing neural networks can be written in the form

$$\boldsymbol{H}_u = \xi\left(\boldsymbol{X}_u, \bigoplus_{v\in\mathcal{N}_u} \mu(\boldsymbol{X}_u, \boldsymbol{X}_v)\right),$$

where μ is a learnable message function, $\bigoplus$ is a permutation-invariant aggregate function, and ξ is the update function. In the case of GCN or GAT, this can be further simplified to

$$\boldsymbol{H}_u = \sigma\left(\sum_{v\in\mathcal{N}_u\cup\{u\}} a_{uv}\mu(\boldsymbol{X}_v)\right), \tag{1}$$

where a is either given by the normalized augmented adjacency matrix (GCN) or the attention mechanism (GAT), μ is a linear transformation, and σ is an activation function.

The learning mechanism behind GCN was first analyzed by Li et al. (2018), who showed that graph convolution is essentially a type of Laplacian smoothing. It was also noted that as deeper layers are stacked, the architecture risks suffering from over-smoothing. This issue makes features indistinguishable and hurts the classification accuracy, which goes against the common understanding that

the deeper the model, the better its learning capacity will be. Furthermore, it prevents the stacking of many layers and impair the ability to model long range dependencies.

Over the past few years, over-smoothing has been observed in many traditional GNNs, prompting a flurry of research into understanding (Oono & Suzuki, 2020; Cai & Wang, 2020; Yan et al., 2021) and alleviating (Luan et al., 2019; Zhao & Akoglu, 2020; Rusch et al., 2022) the issue.

2.2 Graph Neural Diffusion

GRAND is a continuous-depth architecture for deep learning on graph proposed by Chamberlain et al. (2021b). It drew inspiration from the heat diffusion process in mathematical physics, and follows the same vein as other PDE-inspired neural networks (Rusch et al., 2022; Eliasof et al., 2021; Chamberlain et al., 2021a). Central to the formulation of GRAND is the diffusion equation on graph

$$\frac{\partial \boldsymbol{X}(t)}{\partial t} = \operatorname{div}[\boldsymbol{G}(\boldsymbol{X}(t), t)\nabla \boldsymbol{X}(t)], \tag{2}$$

where $\boldsymbol{G} = \operatorname{diag}\left(a(\boldsymbol{X}_i(t), \boldsymbol{X}_j(t), t)\right)$ is an $e \times e$ diagonal matrix giving the diffusivity between connected vertices, which describes the thermal conductance property of the graph. The architecture utilises the encoder-decoder design given by $\boldsymbol{Y} = \psi(\boldsymbol{X}(T))$, where $\boldsymbol{X}(T) \in \mathbb{R}^{n\times d}$ is computed as

$$\boldsymbol{X}(T) = \boldsymbol{X}(0) + \int_0^T \frac{\partial \boldsymbol{X}}{\partial t}(t)dt, \text{ with } \boldsymbol{X}(0) = \phi(\boldsymbol{X}). \tag{3}$$

In the simplest case when $\boldsymbol{G}$ is only dependent on the initial node features, the differential in equation 3 simplifies to

$$\frac{\partial \boldsymbol{X}}{\partial t}(t) = (\boldsymbol{A}(\boldsymbol{X}) - \boldsymbol{I})\boldsymbol{X}(t), \tag{4}$$

where $\boldsymbol{A}(\boldsymbol{X}) = [(a(\boldsymbol{X}_i(t), \boldsymbol{X}_j(t), t))]$ is an $n \times n$ matrix with the same structure as the adjacency matrix of the graph. From now on, we omit the term $\boldsymbol{X}$ when writing $\boldsymbol{A}(\boldsymbol{X})$. The entries of $\boldsymbol{A}$ are exactly those in $\boldsymbol{G}$, and thus determines the diffusivity. Furthermore, $\boldsymbol{A}$ can be informally thought of as the attention weight between vertices. Building upon this heuristic, GRAND models the attention matrix $\boldsymbol{A}$ in equation 4 by the multi-head self-attention mechanism, where

$$\boldsymbol{A} = \frac{1}{h}\sum_{l=1}^{h} \boldsymbol{A}^l(\boldsymbol{X}) \tag{5}$$

with h being the number of heads and the attention matrix $\boldsymbol{A}^l(\boldsymbol{X}) = (a^l(\boldsymbol{X}_i, \boldsymbol{X}_j))$, for $l = 1, \ldots, h$. With this specific implementation, which is called GRAND-l, we obtain $\boldsymbol{A}$ as a right-stochastic matrix with positive entries.

As has been pointed out by Chamberlain et al. (2021b); Thorpe et al. (2022), many GNN architectures, including GAT and GCN, can be formalised as a discretisation scheme of equation 2 if no non-linearity is used between the layers. The intuition behind the connection can readily be seen from equation 1, where each subsequent node data is computed similar or equal to a weighted average of the neighboring node features. This gives rise to the diffusive nature of these architectures, resonating with the work of Gasteiger et al. (2019), which asserts that diffusion improves graph learning.

The choice of using a time-independent attention function amounts to all the layers sharing the same parameters, making the model lightweight and less prone to over-fitting. The original authors claimed that this implementation solves the over-smoothing problem and performs well with many layers. However, our analysis shows that this is not the case.

3 Does GRAND suffer from over-smoothing?

Various works (Oono & Suzuki, 2020; Cai & Wang, 2020; Rusch et al., 2022) have attributed the occurrence of the over-smoothing issue to the exponential convergence of node representations. We formally define this phenomenon for continuous depth graph neural networks.

Definition 1. *Let $\boldsymbol{X}(t) \in \mathbb{R}^{n\times d}$ denote the feature representation at time $t \geq 0$. $\boldsymbol{X}$ is said to experience over-smoothing if there exists a vector $\boldsymbol{v} \in \mathbb{R}^d$ and constants $C_1, C_2 > 0$ such that for $\boldsymbol{V} = (\boldsymbol{v}, \boldsymbol{v}, \dots, \boldsymbol{v})^\top$*

$$\|\boldsymbol{X}(t) - \boldsymbol{V}\|_\infty \leq C_1 e^{-C_2 t}. \tag{6}$$

This definition is similar in spirit to the one given by Rusch et al. (2022). We will show that the GRAND architecture suffers from over-smoothing.

Proposition 1. *Given a positive right-stochastic matrix $\boldsymbol{A} \in \mathbb{R}^{n\times n}$, there exists an invertible matrix $\boldsymbol{P} \in \mathbb{C}^{n\times n}$ such that*

$$\boldsymbol{P}^{-1}(\boldsymbol{A} - \boldsymbol{I})\boldsymbol{P} = \begin{pmatrix} 0 & 0 & \dots & 0 \\ 0 & \boldsymbol{J}_1 & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & \boldsymbol{J}_m \end{pmatrix} \tag{7}$$

where each $\boldsymbol{J}_i$ is a Jordan block associated with some eigenvalue β_i of $\boldsymbol{A} - \boldsymbol{I}$ with $\operatorname{Re}\beta_i < 0$, and $\boldsymbol{P}$ can be chosen such that its first column is the vector $(1, 1, \dots, 1)^\top$.

The Jordan decomposition of $\boldsymbol{A}$ allows us to study the long term behaviour of GRAND through its matrix spectrum. The fact that the real part of all eigenvalues of $\boldsymbol{A} - \boldsymbol{I}$ is equal or lesser than 0 will be essential in our analysis.

Proposition 2. *With the dynamic given by equation 3, equation 4, and equation 5, $\boldsymbol{X}$ experiences over-smoothing.*

Remark 1. *A random walk viewpoint of GRAND was given by Thorpe et al. (2022). Using this, the authors were able to show node representations in the forward Euler discretization of the GRAND dynamic given by*

$$\boldsymbol{X}(k\delta_t) = \boldsymbol{X}((k-1)\delta_t) + \delta_t(\boldsymbol{A} - \boldsymbol{I})\boldsymbol{X}((k-1)\delta_t), \text{ for } k = 0, 1, 2, \dots, K \text{ and } \delta_t < 1,$$

converges in distribution to a stationary state. Our spectral method can be adapted to this discretized setting (see Appendix D). We note that our analysis is strictly stronger through both the use of the ∞-norm and the exact exponential rate of convergence.

Proposition 2 conclusively shows that GRAND-l still suffers from over-smoothing. Of course, if we consider more general variants of GRAND, the above arguments no longer hold. Although we can not rigorously assert that over-smoothing affect all implementations of GRAND, it can be argued that the occurrence of this phenomenon in GRAND intuitively makes sense since diffusion have the tendency to 'even out' distribution over time (further discussion can be found in Appendix B). Hence, a purely diffusion based model like GRAND is inherently ill-suited for deep networks.

4 DEEPGRAND: DEEPER GRAPH NEURAL DIFFUSION

4.1 MODEL FORMULATION

We propose DeepGRAND, a new class of continuous-depth graph neural networks based on GRAND that retains the advantage of diffusion and is capable of learning at much higher depth. It leverages a perturbation to the graph diffusivity $\boldsymbol{A} - (1+\epsilon)\boldsymbol{I}$ and a scaling factor $\langle \boldsymbol{X}(t)\rangle^\alpha$ to make the model both more stable and more resilient to the over-smoothing issue.

Denote the i-th column of $\boldsymbol{X}$ by $\boldsymbol{X}_i$, and the usual 2-norm by $\|\cdot\|$. For constant $\alpha > 0$, we define the column-wise norm matrix $\langle \boldsymbol{X}(t)\rangle^\alpha \in \mathbb{R}^{n\times d}$

$$\langle \boldsymbol{X}(t)\rangle^\alpha = \begin{pmatrix} \|\boldsymbol{X}_1\|^\alpha & \|\boldsymbol{X}_2\|^\alpha & \dots & \|\boldsymbol{X}_d\|^\alpha \\ \|\boldsymbol{X}_1\|^\alpha & \|\boldsymbol{X}_2\|^\alpha & \dots & \|\boldsymbol{X}_d\|^\alpha \\ \vdots & \vdots & \ddots & \vdots \\ \|\boldsymbol{X}_1\|^\alpha & \|\boldsymbol{X}_2\|^\alpha & \dots & \|\boldsymbol{X}_d\|^\alpha \end{pmatrix}, \tag{8}$$

The dynamic of DeepGRAND is given by equation 3, with $\frac{\partial \boldsymbol{X}}{\partial t}$ given by

$$\frac{\partial \boldsymbol{X}}{\partial t}(t) = \big(\boldsymbol{A} - (1+\epsilon)\boldsymbol{I}\big)\boldsymbol{X}(t) \odot \langle \boldsymbol{X}(t)\rangle^\alpha, \tag{9}$$

where $\odot$ is the Hadamard product. Our proposed model thus represents an alteration to the central GRAND dynamics, one which we will shown to be fruitful in the coming sections.

4.2 THEORETICAL ADVANTAGES OF DEEPGRAND

To explain the theoretical motivation behind DeepGRAND, we first note that the convergence property of all nodes to a pre-determined feature vector is not necessarily an undesirable trait. It guarantees the model will remain bounded and not 'explode'. Our perturbation by ϵ serves to slightly strengthen this behaviour. Its purpose is to reduce the real part of all eigenvalues to negative values, making all node representations converge to 0 as the integration limit T tends to infinity.

We observe that it is in fact the rate of convergence that is the chief factor in determining the range of effective depth. If the model converges very slowly, it is clear we can train it at high depth without ever having to worry about over-smoothing. As such, if we can control the convergence rate, we would be able to alleviate the over-smoothing issue. The addition of the term $\langle \boldsymbol{X}(t) \rangle^{\alpha} \in \mathbb{R}^{n \times d}$ serves this purpose. As the node representations come close to their limits, this term acts as a scaling factor to slow down the convergent process.

An exact bound for the convergent rate of our model is presented in Proposition 3.

Proposition 3. *Assuming $\boldsymbol{A}$ is a right-stochastic and radial matrix. With the dynamic given in equation 9, we have the bound for each column $\boldsymbol{X}_i$*

$$\left((2+\epsilon)\alpha T + \|\boldsymbol{X}_i(0)\|^{-\alpha}\right)^{\frac{-1}{\alpha}} \leq \|\boldsymbol{X}_i(T)\| \leq \left(\epsilon \alpha T + \|\boldsymbol{X}_i(0)\|^{-\alpha}\right)^{\frac{-1}{\alpha}}. \tag{10}$$

The equation 10 shows that each column $\boldsymbol{X}_i$ is bounded on both sides by polynomial-like terms. Hence, it can no longer converge at exponential speed. The over-smoothing problem is thus alleviated. Note also that the right hand side of equation 10 converges to 0 as T goes to infinity. This ensures node representation will not explode in the long run and helps with numerical stability.

5 EMPIRICAL ANALYSIS

In this section, we conduct experiments to compare the performance of our proposed method DeepGRAND and the baseline GRAND. We aim to point out that DeepGRAND is able to achieve better accuracy when trained with higher depth and limitted number of labelled nodes per class. We also compare our results with those of GRAND++ (Thorpe et al., 2022) and popular GNNs architectures such as GCN, GAT and GraphSage.

5.1 EXPERIMENTS

5.1.1 DATA

Cora (McCallum et al., 2000) : A citation network consisting of more than 2700 publications with more than 5000 connections and 7 classes. Each node is a vector of 0s and 1s indicating the presence - absence of the corresponding words in a corpus of 1433 words.

Citeseer (Giles et al., 1998) : Similar to Cora, the Citeseer dataset is a network of 3312 scientific publications with 4732 connections and each publication is classified into 6 classes. The node vectors consist of 0s and 1s indicating the presence - absence of corresponding words in a library of 3703 words.

Pubmed (Sen et al., 2008) : Contains 19717 scientific publications related to diabetes from the Pubmed database separated into 3 classes. The network consists of 44338 connections and each node vector is the TF/IDF representation from a corpus of 500 unique words.

CoauthorCS (Monti et al., 2016) : Co-authorship graph of authors with publications in computer science fields. The nodes of the network represent the authors and two authors are connected if they share co-publish. Each node represents each author's papers keywords and the classes represents the authors' most notable research fields or studies. This graph consists of 18333 nodes and 163788 edges with 15 different classes.

Computers (McAuley et al., 2015) : A co-purchase graph extracted from Amazon where each node represents a product and the edges represent the co-purchased relations. The node features are bag-of-words vectors extracted from the product reviews. The dataset consists of 13752 nodes with 491722 edges. All the nodes are classified into 10 different product categories.

Photo (McAuley et al., 2015) : Similar to Computers, the node features in the Photo dataset represents the bag-of-words vectors derived from the product review and nodes (products) are connected if frequently co-purchased. The dataset consists of 7487 nodes with 119043 edges. The nodes are classified into 8 classes representing 8 product categories.

5.1.2 SETUP

Experiment setup For most of the experiments, we used the same set of hyperparameters as GRAND and only changed some settings for the ablation studies specified in sections 5.2, 5.3 and 5.4. Similar to GRAND-l, we denote the specific implimentation of DeepGRAND when $\boldsymbol{A}$ is only dependent on the initial node features as DeepGRAND-l. The experiments are conducted on the common benchmarks for node classification tasks : Cora, Citeseer, Pubmed, Computers, Photo and CoauthorCS using our NVIDIA RTX 3090 graphics card.

One thing to note is that we have made slight adjustments to the original implementation of GRAND. Firstly, in the original work, the authors of GRAND included a learnable scaling term α in front of equation 4. Secondly, they also added the value of $\boldsymbol{X}(0)$ scaled by a learnable scalar β to the equation. The final GRAND dynamics used in the original implementation is indicated in equation 11.

$$\frac{\partial \boldsymbol{X}}{\partial t}(t) = \alpha(\boldsymbol{A}(\boldsymbol{X}) - \boldsymbol{I})\boldsymbol{X}(t) + \beta\boldsymbol{X}(0) \tag{11}$$

The initial value $\boldsymbol{X}(0)$ is treated as a source term added to the overall dynamics before each evaluation step of the ODE solver. This modification is task specific. For fair comparison of GRAND and DeepGRAND, we have removed this from the original implementation and all of our experiments.

5.1.3 EVALUATION

We aim to demonstrate the advantages of DeepGRAND by evaluating our proposed method using the following experiments:

Ablation study with different depths (5.2) : For this experiment, we trained both GRAND and DeepGRAND using different integration limits T and compare the respective test accuracies.

- **For Cora, Citeseer and Pubmed** : We used the default planetoid splitting method available for Cora, Citeseer and Pubmed. For each T value, we trained GRAND-l, GRAND-nl and DeepGRAND-l with 10 random seeds (weights initializations) per method then calculated mean and standard deviation of the test accuracies across the random seeds.
- **For Computers, Photo and CoauthorCS** : We used random splitting for the 3 remaining benchmarks. For each T value, we trained all methods with 10 random seeds and calculated mean and standard deviation of the test accuracies across the random seeds.

Ablation study with different label rates (5.3) : For all of the benchmarks, we used random splitting and trained GRAND-l, GRAND-nl and DeepGRAND-l with different number of labelled nodes per class. For each experiment, we ran with 10 random seeds per method and calculated mean and standard deviation of the test accuracies across the seeds.

Ablation study - effect of α (5.4) : For this section we aimed to study the effect of the exponential α on the ODE solver's ability to integrate. For each α value, we trained DeepGRAND-l with 10 random seeds and calculated the mean and standard deviation of the test accuracies across the seeds.

Note : For hyper-parameters tuning for DeepGRAND-l, we reused most of the hyper-parameters tuned for GRAND (number of neurons for hidden layers, attention dimensionality, ODE block types, number of attention heads, etc.), and only tuned the hyper-parameters that are specific to DeepGRAND - the α and ϵ values specified in equation 9.

Summary : For both experiment sets presented in Table 1 and Table 2, we observed that our proposed dynamics achieved a substantially higher test accuracies than the baseline method and is less prone to performance decay effect when trained with deep architectures (large depth T). Furthermore, the results in Table 3 reveals that DeepGRAND-l is also more robust when the number of labelled nodes per class is low.

5.2 DEEPGRAND IS MORE ADAPTABLE TO DEEP ARCHITECTURES

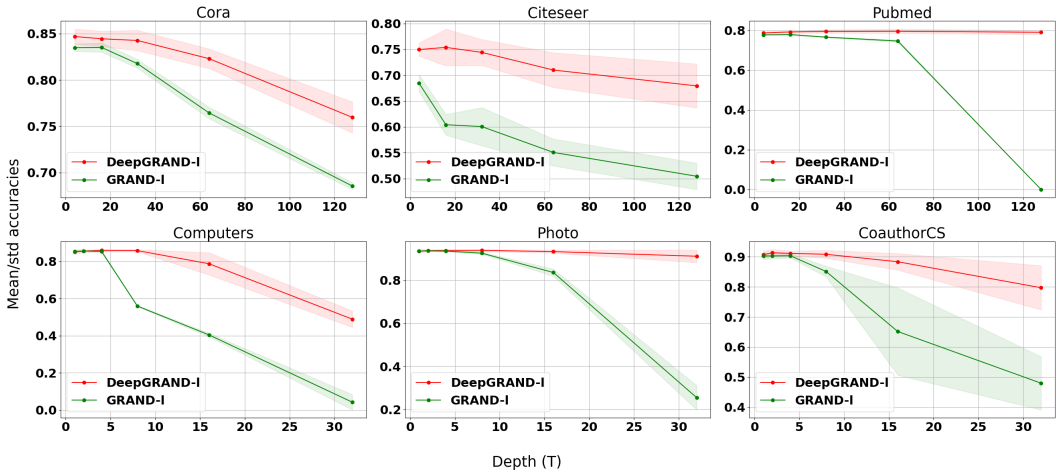

Figure 1: Effect of depth T on the performance of DeepGRAND-l and GRAND-l on Cora, Citeseer, Pubmed, Computers, Photo and CoauthorCS benchmarks. For the first three benchmarks, we used the default planetoid split to conduct our experiments. For the later three, we used random split. For each depth value T, we used 10 random weights initializations then calculated the mean and standard deviation of the test accuracies.

Table 1: The means and standard deviations of test accuracies of DeepGRAND-l and variants of GRAND experimented with different depths on Cora, Citeseer and Pubmed under the default Planetoid split. Experiments were done using 10 random weights initializations for each dataset and depth value. NA: ODE solver failed. (Note: The results for GRAND++-l was imported from (Thorpe et al., 2022)).

Dataset	Depth (T)	GRAND-l	GRAND-nl	GRAND++-l	DeepGRAND-l (ours)
Cora	4	83.49 ± 0.40	83.31 ± 0.27	81.98 ± 1.42	**84.69 ± 0.23**
	16	83.51 ± 0.51	82.53 ± 0.35	82.49 ± 1.37	**84.45 ± 0.29**
	32	81.76 ± 0.47	81.66 ± 0.25	82.48 ± 0.71	**84.26 ± 0.44**
	64	76.45 ± 0.63	78.70 ± 1.08	80.99 ± 1.76	**82.13 ± 0.67**
	128	68.55 ± 0.38	74.60 ± 0.51	**80.29 ± 1.98**	76.35 ± 0.89
Citeseer	4	68.45 ± 1.66	69.01 ± 1.52	72.58 ± 3.79	**75.39 ± 0.59**
	16	60.42 ± 2.00	60.65 ± 1.78	73.84 ± 2.66	**74.97 ± 0.57**
	32	60.06 ± 3.69	58.77 ± 1.55	73.29 ± 1.29	**74.42 ± 1.06**
	64	55.06 ± 2.61	55.71 ± 2.28	**72.81 ± 2.18**	71.00 ± 1.08
	128	50.45 ± 2.56	52.90 ± 1.80	NA	**67.93 ± 0.78**
Pubmed	4	77.78 ± 0.10	77.50 ± 0.34	79.20 ± 0.74	**79.61 ± 0.98**
	16	77.93 ± 0.36	77.76 ± 0.15	79.49 ± 0.84	**79.52 ± 0.56**
	32	76.65 ± 0.26	78.41 ± 0.37	**79.81 ± 1.61**	79.26 ± 0.35
	64	74.68 ± 0.52	73.31 ± 1.23	NA	**79.08 ± 0.59**
	128	NA	NA	NA	**78.74 ± 0.46**

We provide empirical evidence for our argument that the dynamics of DeepGRAND is more resilient to deeper architectures. Specifically, we compared the change in performance of both GRAND and DeepGRAND as the integration limit T increases. In Table 1, for each T value in $\{4, 16, 32, 64, 128\}$, we trained both methods for Cora, Citeseer and Pubmed datasets using the default planetoid splitting method with 10 random initializations. For the other benchmarks: Computers, Photo and CoauthorCS, we splitted the datasets randomly and trained with 10 random initializations for each integration limit T in $\{1, 2, 4, 8, 16, 32\}$ and recorded the results in Table 2. For each initialization, we used the default label rate of 20 labelled nodes per class as indicated in

Table 2: The means and standard deviations of test accuracies of DeepGRAND-l and variants of GRAND experimented with different depths on the Computers, Photo and CoauthorCS datasets under random split. Experiments were done using 10 random weights initializations for each dataset and depth value. NA: ODE solver failed. (Note: The results for GRAND++-l was imported from (Thorpe et al., 2022)).

Dataset	Depth (T)	GRAND-l	GRAND-nl	GRAND++-l	DeepGRAND-l (ours)
Computers	1	85.32 ± 0.62	85.67 ± 0.14	84.11 ± 0.51	**85.87 ± 0.30**
	2	85.57 ± 2.89	85.02 ± 0.67	84.34 ± 0.71	**85.78 ± 0.28**
	4	85.36 ± 0.49	84.23 ± 0.52	84.19 ± 0.93	**85.54 ± 0.49**
	8	55.94 ± 0.32	52.14 ± 0.44	80.45 ± 1.24	**85.12 ± 0.32**
	16	40.40 ± 1.02	42.11 ± 3.03	**78.97 ± 2.33**	78.71 ± 2.23
	32	4.18 ± 4.10	8.19 ± 5.13	**76.01 ± 1.33**	48.89 ± 0.28
Photo	1	93.55 ± 0.20	93.35 ± 0.31	92.93 ± 0.84	**93.86 ± 0.30**
	2	93.65 ± 0.24	93.28 ± 0.29	**93.89 ± 0.22**	93.80 ± 0.28
	4	93.52 ± 0.32	92.29 ± 0.41	93.54 ± 0.38	**93.76 ± 0.22**
	8	92.57 ± 0.47	92.50 ± 0.57	93.01 ± 0.52	**93.46 ± 0.42**
	16	83.58 ± 1.67	93.29 ± 0.36	92.69 ± 0.61	**93.29 ± 0.36**
	32	25.45 ± 5.63	45.23 ± 7.32	**92.94 ± 0.90**	91.13 ± 0.95
CoauthorCS	1	90.31 ± 1.11	91.12 ± 0.44	90.42 ± 0.76	**91.36 ± 0.16**
	2	90.31 ± 0.86	90.79 ± 0.14	90.53 ± 0.54	**91.17 ± 0.65**
	4	90.37 ± 0.74	89.90 ± 0.75	**90.89 ± 0.36**	90.85 ± 0.62
	8	85.13 ± 1.77	87.18 ± 1.04	90.18 ± 0.47	**90.74 ± 0.65**
	16	65.16 ± 14.53	77.68 ± 3.74	**90.24 ± 0.30**	88.37 ± 1.11
	32	47.96 ± 8.86	49.71 ± 12.08	NA	**79.73 ± 2.35**

the original GRAND implementation. On all benchmarks, we observe that our proposed dynamics performed substantially better for larger values of T, indicating less performance degradation under the effect of over-smoothing.

5.3 DEEPGRAND IS MORE RESILIENT UNDER LIMITED LABELLED TRAINING DATA

In previous experiments, we showed that DeepGRAND outperforms the baseline GRAND when trained with high integration limit with 20 labelled nodes per class. In the following experiments, we aim to demonstrate that DeepGRAND also achieves superior results with limited number of labelled nodes per class. For all of the benchmarks : Cora, Citeseer, Pubmed, Computers, Photo and CoauthorCS, we reused the best depth T values obtained from the hyper-parameter tuning results from the authors of GRAND for experiments of both GRAND and DeepGRAND to compare the performance under different numbers of labelled nodes per class. For each dataset, we experimented with 1, 2, 5, 10, 20 labelled nodes per class and compared the test accuracies of GRAND-l, GRAND-nl and DeepGRAND-l. On top of GRAND, our results recorded in Table 3 also shows that DeepGRAND outperforms some of the common GNN architectures like GCN, GAT and GraphSage.

5.4 EFFECT OF α ON THE DYNAMICS OF DEEPGRAND

The exponential α defined in our DeepGRAND dynamics directly influences the ability to integrate of the ODE solver. Specifically, a high α value (from 0.5 to 1.5) will likely cause the ODE solver to receive the MaxNFE error (Max Number of Function Evaluations exceeded). Hence, harming the overall performance.

In the following experiment, for each exponential value α in $\{0.0001, 0.1, 0.5\}$, we trained with 10 random weights initializations and increased the integration depth T from 4 to 128 to observe the behavior the test accuracies. The results in Figure 2 suggest the performance of DeepGRAND under smaller α values are significantly better. This gives us a rough idea of what range of α values to use when we perform hyper-parameter tuning for DeepGRAND.

Table 3: The means and standard deviations of test accuracies of DeepGRAND-l, variants of GRAND, and other common GNNs experimented with different number of labelled nodes per class. (Note: The results for GRAND++-l, GCN, GAT and GraphSage were imported from Thorpe et al. (2022)).

Dataset	# labelled	GRAND-l	GRAND-nl	GRAND++-l	GCN	GAT	GraphSage	DeepGRAND-l (ours)
Cora	20	**82.86 ± 1.12**	82.72±2.45	**82.95 ± 1.37**	82.07 ± 2.03	79.92 ± 2.28	80.04 ± 2.54	**83.87 ± 0.49**
	10	**80.67 ± 2.19**	80.51 ± 1.11	**80.86 ± 2.99**	78.82 ± 5.38	76.31 ± 4.87	75.04 ± 5.03	**82.47 ± 1.11**
	5	77.09 ± 3.05	**77.68 ± 2.85**	**77.80 ± 4.46**	73.86 ± 7.97	71.04 ± 5.74	68.14 ± 6.95	**79.00 ± 1.87**
	2	**74.23 ± 5.58**	**69.44 ± 4.27**	66.92 ± 10.04	60.85 ± 14.01	58.30± 13.55	53.96 ± 12.18	**67.57 ± 8.02**
	1	**57.93± 8.09**	**55.86 ± 10.04**	54.94 ± 16.09	47.72 ± 15.33	47.86± 15.38	43.04 ± 14.01	**63.79 ± 7.14**
Citeseer	20	71.74 ± 2.94	73.16 ± 3.09	**73.53 ± 3.31**	**74.21 ± 2.90**	73.22 ± 2.90	72.02 ± 2.82	**73.64 ± 1.73**
	10	66.26 ± 4.19	67.84 ± 3.96	**72.34 ± 2.42**	**72.18 ± 3.47**	71.35 ± 4.92	68.90 ± 5.08	**72.87 ± 2.29**
	5	**69.00 ± 3.74**	66.90 ± 4.62	**70.03 ± 3.63**	67.24 ± 4.19	67.37 ± 5.08	64.79 ± 5.16	**70.84 ± 3.04**
	2	**58.35 ± 8.98**	56.35 ± 5.54	**64.98 ± 8.31**	58.06 ± 9.76	55.55 ± 9.19	54.39 ± 11.37	**63.26 ± 3.89**
	1	49.65 ± 8.67	47.32 ± 6.66	**58.95 ± 9.59**	48.94 ± 10.24	**50.31 ± 14.27**	48.81 ± 11.45	**53.06 ± 8.49**
Pubmed	20	**78.42 ± 0.46**	75.19 ± 1.77	**79.16 ± 1.37**	76.89 ± 3.27	75.55 ± 4.11	74.55 ± 3.09	**79.23 ± 1.23**
	10	74.10 ± 1.88	**74.18 ± 1.99**	**75.13 ± 3.88**	72.59 ± 3.19	72.44 ± 3.50	70.74 ± 3.11	**76.95 ± 3.24**
	5	71.05 ± 1.87	**72.07 ± 2.15**	**71.99 ± 1.91**	68.69 ± 7.93	68.54 ± 5.75	66.07 ± 6.16	**73.37 ± 2.26**
	2	**71.44 ± 3.85**	65.50 ± 9.49	**69.31 ± 4.87**	60.45 ± 16.20	60.24 ± 14.44	58.97 ± 12.65	**70.75 ± 3.86**
	1	62.41 ± 7.59	**63.47 ± 5.47**	**65.94 ± 4.87**	58.61 ± 12.83	58.84 ± 12.81	55.53 ± 12.71	**64.96 ± 7.08**
Computers	20	**84.04 ± 0.98**	83.28 ± 1.24	**85.73 ± 0.50**	82.94 ± 1.54	80.05 ± 1.81	79.98 ± 0.96	**87.27 ± 1.45**
	10	82.34 ± 2.18	81.27 ± 3.02	**82.99 ± 0.81**	**82.53 ± 0.74**	76.04 ± 0.35	74.66 ± 1.29	**83.79 ± 1.45**
	5	78.69 ± 0.79	79.65 ± 2.02	**82.64 ± 0.56**	**82.47 ± 0.97**	71.43 ± 7.34	64.83 ± 1.62	**82.06 ± 2.02**
	2	66.21 ± 11.26	65.11 ± 8.31	**76.47 ± 1.48**	**76.90 ± 1.49**	65.07 ± 8.86	42.63 ± 42.9	**75.26 ± 2.20**
	1	**49.80 ± 14.97**	47.26 ± 11.23	**67.65 ± 0.37**	49.46 ± 1.65	37.14 ± 7.81	27.65 ± 2.39	**65.35 ± 6.65**
Photo	20	**93.22 ± 0.44**	91.76 ± 1.51	**93.55 ± 0.38**	91.95 ± 0.11	89.38 ± 2.48	91.29 ± 0.67	**93.52 ± 0.40**
	10	**90.80 ± 1.37**	89.02 ± 2.54	**90.65 ± 1.19**	90.41 ± 0.35	87.42 ± 2.38	84.38 ± 1.75	**90.59 ± 2.27**
	5	88.06 ± 2.59	**88.31 ± 1.63**	**88.33 ± 1.21**	**88.86 ± 1.56**	83.01 ± 3.64	78.26 ± 1.93	87.75 ± 1.59
	2	82.60 ± 2.96	80.61 ± 4.43	**83.71 ± 0.90**	**83.61 ± 0.71**	76.89 ± 4.89	51.93 ± 4.21	**84.59 ± 1.80**
	1	75.05 ± 5.44	**76.33 ± 4.79**	**83.12 ± 0.78**	**82.94 ± 2.17**	73.58 ± 8.15	45.36 ± 7.13	73.50 ± 5.06
CoauthorCS	20	90.99 ± 0.56	90.59 ± 0.97	90.80 ± 0.34	91.09 ± 0.35	79.95 ± 2.88	**91.33 ± 0.36**	**91.53 ± 0.33**
	10	89.00 ± 2.08	**89.95 ± 0.68**	86.94 ± 0.46	88.60 ± 0.50	74.71 ± 3.35	**89.68 ± 0.39**	**89.39 ± 0.89**
	5	84.19 ± 3.59	**87.01 ± 1.97**	84.83 ± 0.84	**86.66 ± 0.43**	71.65 ± 4.53	**89.06 ± 0.69**	86.05 ± 4.64
	2	75.19 ± 3.84	**76.66 ± 6.85**	76.53 ± 1.85	**83.61 ± 1.49**	63.12 ± 6.09	76.51 ± 1.31	**79.25 ± 3.89**
	1	56.58 ± 8.44	**66.44 ± 8.17**	60.30 ± 1.50	**65.22 ± 2.25**	51.13 ± 5.24	61.35 ± 1.35	**65.15 ± 6.34**

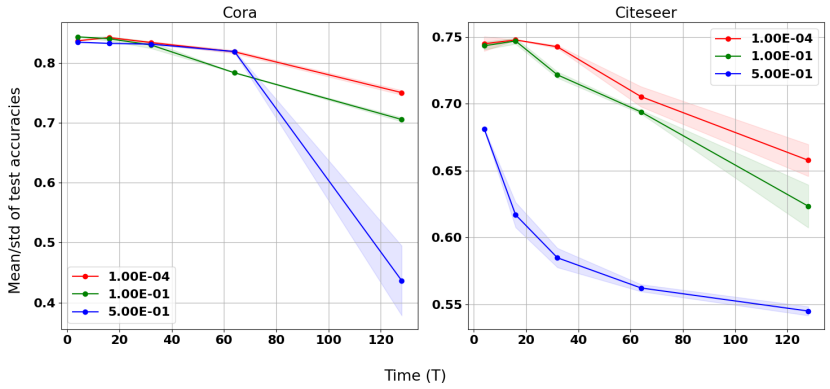

Figure 2: The change in test accuracies of DeepGRAND-l as depth value T increases under different α values. **Left** - effect of α on the accuracies of Cora, **Right** - effect of α on the accuracies of Citeseer. Each α value is trained on 10 random seeds for both benchmarks.

6 Conclusion

We propose DeepGRAND, a class of continuous-depth graph neural networks that leverage a novel data-dependent scaling term and perturbation to the graph diffusivity to decrease the saturation rate of the underlying diffusion process, thus alleviating the over-smoothing issue. We also prove that the proposed method stabilize the learning of the model by controlling the norm of the node representation. We theoretically and empirically showed its advantage over GRAND and other popular GNNs in term of resiliency to over-smoothing and overall performance. DeepGRAND is a first-order system of ODEs. It is natural to explore an extension of DeepGRAND to a second-order system of ODEs and study the dynamics of this system. It is also interesting to leverage advanced methods in improving the Neural ODEs (Chen et al., 2018) to further develop DeepGRAND.

REFERENCES

Peter Battaglia, Razvan Pascanu, Matthew Lai, Danilo Jimenez Rezende, et al. Interaction networks for learning about objects, relations and physics. *Advances in neural information processing systems*, 29, 2016.

Michael M Bronstein, Joan Bruna, Yann LeCun, Arthur Szlam, and Pierre Vandergheynst. Geometric deep learning: going beyond euclidean data. *IEEE Signal Processing Magazine*, 34(4):18–42, 2017.

Michael M. Bronstein, Joan Bruna, Taco Cohen, and Petar Veličković. Geometric deep learning: Grids, groups, graphs, geodesics, and gauges, 2021. URL https://arxiv.org/abs/2104.13478.

Chen Cai and Yusu Wang. A note on over-smoothing for graph neural networks, 2020. URL https://arxiv.org/abs/2006.13318.

Benjamin Paul Chamberlain, James Rowbottom, Davide Eynard, Francesco Di Giovanni, Xiaowen Dong, and Michael M. Bronstein. Beltrami flow and neural diffusion on graphs. In A. Beygelzimer, Y. Dauphin, P. Liang, and J. Wortman Vaughan (eds.), *Advances in Neural Information Processing Systems*, 2021a. URL https://openreview.net/forum?id=4YlE2huxEsl.

Benjamin Paul Chamberlain, James Rowbottom, Maria I. Gorinova, Stefan D Webb, Emanuele Rossi, and Michael M. Bronstein. GRAND: Graph neural diffusion. In *The Symbiosis of Deep Learning and Differential Equations*, 2021b. URL https://openreview.net/forum?id=_1fu_cjsaRE.

Deli Chen, Yankai Lin, Wei Li, Peng Li, Jie Zhou, and Xu Sun. Measuring and relieving the over-smoothing problem for graph neural networks from the topological view. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pp. 3438–3445, 2020.

Ricky T. Q. Chen, Yulia Rubanova, Jesse Bettencourt, and David K Duvenaud. Neural ordinary differential equations. In S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett (eds.), *Advances in Neural Information Processing Systems*, volume 31. Curran Associates, Inc., 2018. URL https://proceedings.neurips.cc/paper/2018/file/69386f6bb1dfed68692a24c8686939b9-Paper.pdf.

Yongqiang Chen, Han Yang, Yonggang Zhang, MA KAILI, Tongliang Liu, Bo Han, and James Cheng. Understanding and improving graph injection attack by promoting unnoticeability. In *International Conference on Learning Representations*, 2022. URL https://openreview.net/forum?id=wkMG8cdvh7-.

Michaël Defferrard, Xavier Bresson, and Pierre Vandergheynst. Convolutional neural networks on graphs with fast localized spectral filtering. In *Proceedings of the 30th International Conference on Neural Information Processing Systems*, NIPS'16, pp. 3844–3852, Red Hook, NY, USA, 2016. Curran Associates Inc. ISBN 9781510838819.

Emilien Dupont, Arnaud Doucet, and Yee Whye Teh. Augmented neural odes. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett (eds.), *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc., 2019. URL https://proceedings.neurips.cc/paper/2019/file/21be9a4bd4f81549a9d1d241981cec3c-Paper.pdf.

David K Duvenaud, Dougal Maclaurin, Jorge Iparraguirre, Rafael Bombarell, Timothy Hirzel, Alan Aspuru-Guzik, and Ryan P Adams. Convolutional networks on graphs for learning molecular fingerprints. In C. Cortes, N. Lawrence, D. Lee, M. Sugiyama, and R. Garnett (eds.), *Advances in Neural Information Processing Systems*, volume 28. Curran Associates, Inc., 2015a. URL https://proceedings.neurips.cc/paper/2015/file/f9be311e65d81a9ad8150a60844bb94c-Paper.pdf.

David K Duvenaud, Dougal Maclaurin, Jorge Iparraguirre, Rafael Bombarell, Timothy Hirzel, Alán Aspuru-Guzik, and Ryan P Adams. Convolutional networks on graphs for learning molecular fingerprints. *Advances in neural information processing systems*, 28, 2015b.

Moshe Eliasof, Eldad Haber, and Eran Treister. PDE-GCN: Novel architectures for graph neural networks motivated by partial differential equations. In A. Beygelzimer, Y. Dauphin, P. Liang, and J. Wortman Vaughan (eds.), *Advances in Neural Information Processing Systems*, 2021. URL https://openreview.net/forum?id=wWtk6GxJB2x.

Chris Finlay, Jörn-Henrik Jacobsen, Levon Nurbekyan, and Adam M. Oberman. How to train your neural ode: the world of jacobian and kinetic regularization. In *ICML*, pp. 3154–3164, 2020. URL http://proceedings.mlr.press/v119/finlay20a.html.

Johannes Gasteiger, Stefan Weißenberger, and Stephan Günnemann. Diffusion improves graph learning. In *Conference on Neural Information Processing Systems (NeurIPS)*, 2019.

C. Lee Giles, Kurt D. Bollacker, and Steve Lawrence. CiteSeer. In *Proceedings of the third ACM conference on Digital libraries - DL '98*. ACM Press, 1998. doi: 10.1145/276675.276685. URL https://doi.org/10.1145/276675.276685.

Justin Gilmer, Samuel S Schoenholz, Patrick F Riley, Oriol Vinyals, and George E Dahl. Neural message passing for quantum chemistry. In *International conference on machine learning*, pp. 1263–1272. PMLR, 2017.

Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT Press, 2016. http://www.deeplearningbook.org.

William L. Hamilton, Rex Ying, and Jure Leskovec. Inductive representation learning on large graphs. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, NIPS'17, pp. 1025–1035, Red Hook, NY, USA, 2017. Curran Associates Inc. ISBN 9781510860964.

Thomas N. Kipf and Max Welling. Semi-Supervised Classification with Graph Convolutional Networks. In *Proceedings of the 5th International Conference on Learning Representations*, ICLR '17, 2017. URL https://openreview.net/forum?id=SJU4ayYgl.

Qimai Li, Zhichao Han, and Xiao-Ming Wu. Deeper insights into graph convolutional networks for semi-supervised learning, 2018. URL https://arxiv.org/abs/1801.07606.

Qi Liu, Maximilian Nickel, and Douwe Kiela. Hyperbolic graph neural networks. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett (eds.), *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc., 2019. URL https://proceedings.neurips.cc/paper/2019/file/103303dd56a731e377d01f6a37badae3-Paper.pdf.

Sitao Luan, Mingde Zhao, Xiao-Wen Chang, and Doina Precup. Break the ceiling: Stronger multi-scale deep graph convolutional networks. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett (eds.), *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc., 2019. URL https://proceedings.neurips.cc/paper/2019/file/ccdf3864e2fa9089f9eca4fc7a48ea0a-Paper.pdf.

Sitao Luan, Chenqing Hua, Qincheng Lu, Jiaqi Zhu, Mingde Zhao, Shuyuan Zhang, Xiao-Wen Chang, and Doina Precup. Is heterophily a real nightmare for graph neural networks on performing node classification?, 2022. URL https://openreview.net/forum?id=LBv-JtAmm4P.

Yao Ma, Xiaorui Liu, Neil Shah, and Jiliang Tang. Is homophily a necessity for graph neural networks? In *International Conference on Learning Representations*, 2022. URL https://openreview.net/forum?id=ucASPPD9GKN.

Julian McAuley, Christopher Targett, Qinfeng Shi, and Anton van den Hengel. Image-based recommendations on styles and substitutes, 2015. URL https://arxiv.org/abs/1506.04757.

Andrew Kachites McCallum, Kamal Nigam, Jason Rennie, and Kristie Seymore. *Information Retrieval*, 3(2):127–163, 2000. doi: 10.1023/a:1009953814988. URL https://doi.org/10.1023/a:1009953814988.

Federico Monti, Davide Boscaini, Jonathan Masci, Emanuele Rodolà, Jan Svoboda, and Michael M. Bronstein. Geometric deep learning on graphs and manifolds using mixture model cnns, 2016. URL https://arxiv.org/abs/1611.08402.

Federico Monti, Michael Bronstein, and Xavier Bresson. Geometric matrix completion with recurrent multi-graph neural networks. *Advances in neural information processing systems*, 30, 2017.

Kenta Oono and Taiji Suzuki. Graph neural networks exponentially lose expressive power for node classification. In *International Conference on Learning Representations*, 2020. URL https://openreview.net/forum?id=S1ldO2EFPr.

Jiezhong Qiu, Jian Tang, Hao Ma, Yuxiao Dong, Kuansan Wang, and Jie Tang. Deepinf: Social influence prediction with deep learning. In *Proceedings of the 24th ACM SIGKDD international conference on knowledge discovery & data mining*, pp. 2110–2119, 2018.

T. Konstantin Rusch, Benjamin P. Chamberlain, James Rowbottom, Siddhartha Mishra, and Michael M. Bronstein. Graph-coupled oscillator networks, 2022. URL https://arxiv.org/abs/2202.02296.

Franco Scarselli, Marco Gori, Ah Chung Tsoi, Markus Hagenbuchner, and Gabriele Monfardini. The graph neural network model. *IEEE transactions on neural networks*, 20(1):61–80, 2008.

Prithviraj Sen, Galileo Namata, Mustafa Bilgic, Lise Getoor, Brian Galligher, and Tina Eliassi-Rad. Collective classification in network data. *AI Magazine*, 29(3):93, September 2008. doi: 10.1609/aimag.v29i3.2157. URL https://doi.org/10.1609/aimag.v29i3.2157.

Matthew Thorpe, Tan Minh Nguyen, Hedi Xia, Thomas Strohmer, Andrea Bertozzi, Stanley Osher, and Bao Wang. GRAND++: Graph neural diffusion with a source term. In *International Conference on Learning Representations*, 2022. URL https://openreview.net/forum?id=EMxu-dzvJk.

Jake Topping, Francesco Di Giovanni, Benjamin Paul Chamberlain, Xiaowen Dong, and Michael M. Bronstein. Understanding over-squashing and bottlenecks on graphs via curvature. In *International Conference on Learning Representations*, 2022. URL https://openreview.net/forum?id=7UmjRGzp-A.

Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, and Yoshua Bengio. Graph Attention Networks. *International Conference on Learning Representations*, 2018. URL https://openreview.net/forum?id=rJXMpikCZ.

Louis-Pascal Xhonneux, Meng Qu, and Jian Tang. Continuous graph neural networks. In Hal Daumé III and Aarti Singh (eds.), *Proceedings of the 37th International Conference on Machine Learning*, volume 119 of *Proceedings of Machine Learning Research*, pp. 10432–10441. PMLR, 13–18 Jul 2020. URL https://proceedings.mlr.press/v119/xhonneux20a.html.

Hedi Xia, Vai Suliafu, Hangjie Ji, Tan Minh Nguyen, Andrea Bertozzi, Stanley Osher, and Bao Wang. Heavy ball neural ordinary differential equations. In A. Beygelzimer, Y. Dauphin, P. Liang, and J. Wortman Vaughan (eds.), *Advances in Neural Information Processing Systems*, 2021. URL https://openreview.net/forum?id=fYLfs9yrtMQ.

Yujun Yan, Milad Hashemi, Kevin Swersky, Yaoqing Yang, and Danai Koutra. Two sides of the same coin: Heterophily and oversmoothing in graph convolutional neural networks. *CoRR*, abs/2102.06462, 2021. URL https://arxiv.org/abs/2102.06462.

Rex Ying, Ruining He, Kaifeng Chen, Pong Eksombatchai, William L Hamilton, and Jure Leskovec. Graph convolutional neural networks for web-scale recommender systems. In *Proceedings of the 24th ACM SIGKDD international conference on knowledge discovery & data mining*, pp. 974–983, 2018.

Muhan Zhang and Yixin Chen. Link prediction based on graph neural networks. *Advances in neural information processing systems*, 31, 2018.

Lingxiao Zhao and Leman Akoglu. Pairnorm: Tackling oversmoothing in gnns. In *International Conference on Learning Representations*, 2020. URL https://openreview.net/forum?id=rkecl1rtwB.

Jiong Zhu, Yujun Yan, Lingxiao Zhao, Mark Heimann, Leman Akoglu, and Danai Koutra. Beyond homophily in graph neural networks: Current limitations and effective designs. In H. Larochelle, M. Ranzato, R. Hadsell, M.F. Balcan, and H. Lin (eds.), *Advances in Neural Information Processing Systems*, volume 33, pp. 7793–7804. Curran Associates, Inc., 2020. URL https://proceedings.neurips.cc/paper/2020/file/58ae23d878a47004366189884c2f8440-Paper.pdf.

Appendix for "DeepGRAND: Deeper Graph Neural Diffusion"

A TECHNICAL PROOFS

Consider the dynamic given by equation 3 and equation 4. We say that a matrix is postitive if and only if all of its entries are positive. Clearly, $\boldsymbol{A}$ is a postive right-stochastic matrix. We recall the well-known Perron-Frobenius theorem

Theorem 1 (Perron-Frobenius)**.** *Let $\boldsymbol{M}$ be a positive matrix. There is a positive real number r, called the Perron–Frobenius eigenvalue, such that r is an eigenvalue of M and any other eigenvalue λ (possibly complex) in absolute value is strictly smaller than r , $|\lambda| < r$. This eigenvalue is simple and its eigenspace is one-dimensional.*

Proof of Proposition 1. Since $\boldsymbol{A}$ is right-stochastic, its Perron-Frobenius eigenvalue is $\alpha_1 = 1$. Its eigenspace has the basis $\boldsymbol{u}_1 = \{1, 1, \dots, 1\}$. Suppose $\{\alpha_1, \alpha_2, \dots, \alpha_k\}$ is the complex spectrum of $\boldsymbol{A}$. That is, they are all complex eigenvalues of $\boldsymbol{A}$. The matrix $\boldsymbol{A} - \boldsymbol{I}$ has eigenvalues $\beta_i = \alpha_i - 1$ for all $i = \overline{1, k}$, so $\beta_1 = 0$ and $\operatorname{Re} \beta_i < 0$ for all $i = \overline{2, k}$. There exists a basis containing $\boldsymbol{u}_1$ of $\mathbb{C}^n$ comprising of generalized eigenvectors of $\boldsymbol{A}$. By rearranging the vectors if needed, the transition matrix $\boldsymbol{P}$ from the standard basis to this basis satisfies

$$\boldsymbol{P}^{-1}(\boldsymbol{A} - \boldsymbol{I})\boldsymbol{P} = \begin{pmatrix} 0 & 0 & \dots & 0 \\ 0 & \boldsymbol{J}_1 & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & \boldsymbol{J}_m \end{pmatrix},$$

where each $\boldsymbol{J}_i$ is a Jordan block associated with some eigenvalue β_i. □

Proof of Proposition 2. Let $\boldsymbol{J} = \boldsymbol{P}^{-1}\boldsymbol{A}\boldsymbol{P}$ and $\boldsymbol{Z}(t) = \boldsymbol{P}^{-1}\boldsymbol{X}(t)$ as in Proposition 1, we can rewrite equation 4 as

$$\frac{\partial \boldsymbol{Z}}{\partial t}(t) = \boldsymbol{J}\boldsymbol{Z}(t),$$

The solution to this matrix differential equation is

$$\boldsymbol{Z}(t) = \exp(t\boldsymbol{J})\boldsymbol{Z}(0),$$

where

$$\exp(t\boldsymbol{J}) = \begin{pmatrix} 1 & 0 & 0 & \dots & 0 \\ & \exp(t\boldsymbol{J}_1) & 0 & \ddots & \vdots \\ & & \exp(t\boldsymbol{J}_2) & \ddots & \vdots \\ & & & \ddots & \vdots \\ & & & & 0 \\ & & & & \exp(t\boldsymbol{J}_m) \end{pmatrix}.$$

Let $\overline{\boldsymbol{Z}}$ be the matrix with the same size as $\boldsymbol{Z}(0)$ obtained by setting every entry in all but the first row of $\boldsymbol{Z}(0)$ be equal to 0. We have

$$\boldsymbol{Z}(t) - \overline{\boldsymbol{Z}} = \boldsymbol{Z}(t) - \begin{pmatrix} 1 & 0 & \dots & 0 \\ 0 & 0 & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & 0 \end{pmatrix} \boldsymbol{Z}(0) = \begin{pmatrix} 0 & 0 & 0 & \dots & 0 \\ & \exp(t\boldsymbol{J}_1) & 0 & \ddots & \vdots \\ & & \exp(t\boldsymbol{J}_2) & \ddots & \vdots \\ & & & \ddots & \vdots \\ & & & & 0 \\ & & & & \exp(t\boldsymbol{J}_m) \end{pmatrix} \boldsymbol{Z}(0).$$

Denote the size of each Jordan block $\boldsymbol{J}_i$ as n_i, we have well-known equality

$$\exp(t\boldsymbol{J}_i) = e^{\beta_i t}\begin{pmatrix} 1 & t & \frac{t^2}{2} & \dots & \frac{t^{n_i-1}}{(n_i-1)!} \\ & 1 & t & \ddots & \vdots \\ & & 1 & \ddots & \vdots \\ & & & \ddots & \vdots \\ & & & & t \\ & & & & 1 \end{pmatrix}.$$

Hence, every non-zero entry of $\boldsymbol{Z}(t) - \overline{\boldsymbol{Z}}$ has the form $e^{\beta_i t}P(t)$ for some $\beta_i < 0$ and polynomial P. Each entry can thus be bounded above in norm by an exponential term of the form $c_1 e^{-c_2 t}$ for some $c_1, c_2 > 0$. A simple calculation shows that $\|\boldsymbol{X}(t) - \boldsymbol{P}\overline{\boldsymbol{Z}}\|_\infty = \|\boldsymbol{P}(\boldsymbol{Z}(t) - \overline{\boldsymbol{Z}})\|_\infty$ would also be bounded by an exponential term. That is, there exists some $C_1, C_2 > 0$ such that

$$\|\boldsymbol{X}(t) - \boldsymbol{P}\overline{\boldsymbol{Z}}\|_\infty \leq C_1 e^{-C_2 t}. \tag{12}$$

Recall that the first column of $\boldsymbol{P}$ is $\boldsymbol{u}^\top = (1, 1, \dots, 1)^\top$, so that

$$\boldsymbol{P}\overline{\boldsymbol{Z}} = \begin{pmatrix} \boldsymbol{Z}(0)_{1,1} & \boldsymbol{Z}(0)_{1,2} & \dots & \boldsymbol{Z}(0)_{1,d} \\ \boldsymbol{Z}(0)_{1,1} & \boldsymbol{Z}(0)_{1,2} & \dots & \boldsymbol{Z}(0)_{1,d} \\ \vdots & \vdots & \ddots & \vdots \\ \boldsymbol{Z}(0)_{1,1} & \boldsymbol{Z}(0)_{1,2} & \dots & \boldsymbol{Z}(0)_{1,d} \end{pmatrix}.$$

Let $\boldsymbol{v} = (\boldsymbol{Z}(0)_{1,1}, \boldsymbol{Z}(0)_{1,2}, \dots, \boldsymbol{Z}(0)_{1,d})^\top$ and $\boldsymbol{V} = (\boldsymbol{v}, \boldsymbol{v}, \dots, \boldsymbol{v})^\top$. We can rewrite equation 12 as

$$\|\boldsymbol{X}(t) - \boldsymbol{V}\|_\infty \leq C_1 e^{-C_2 t}. \tag{13}$$

The claim that $\boldsymbol{v} \in \mathbb{R}^d$ follows from the fact that it is the limit of real-valued vectors with regards to the $\|\cdot\|_\infty$ norm. □

Proof of Proposition 3. Let $\boldsymbol{Y}_i(t) = \|\boldsymbol{X}_i(t)\|^4$, we have

$$\begin{aligned} \boldsymbol{Y}_i'(t) &= 4\|\boldsymbol{X}_i(t)\|^2 \langle \boldsymbol{X}_i'(t), \boldsymbol{X}_i(t)\rangle \\ &= 4\|\boldsymbol{X}_i(t)\|^2 \langle (\boldsymbol{A} - (1+\epsilon)\boldsymbol{I})\boldsymbol{X}_i(t)\|\boldsymbol{X}_i\|^\alpha, \boldsymbol{X}_i(t)\rangle \\ &= 4\|\boldsymbol{X}_i(t)\|^{2+\alpha} \langle (\boldsymbol{A} - (1+\epsilon)\boldsymbol{I})\boldsymbol{X}_i(t), \boldsymbol{X}_i(t)\rangle \\ &= 4\|\boldsymbol{X}_i(t)\|^{2+\alpha} \langle \boldsymbol{A}\boldsymbol{X}_i(t), \boldsymbol{X}_i(t)\rangle - 4(1+\epsilon)\|\boldsymbol{X}_i(t)\|^{4+\alpha} \end{aligned}$$

Since $\boldsymbol{A}$ is a right-stochastic and radial matrix, we have

$$\langle \boldsymbol{A}\boldsymbol{X}_i(t), \boldsymbol{X}_i(t)\rangle \leq \|\boldsymbol{A}\|\|\boldsymbol{X}_i(t)\|^2 = \|\boldsymbol{X}_i(t)\|^2,$$

and

$$-\langle \boldsymbol{A}\boldsymbol{X}_i(t), \boldsymbol{X}_i(t)\rangle \leq \|\boldsymbol{A}\|\|\boldsymbol{X}_i(t)\|^2 = \|\boldsymbol{X}_i(t)\|^2,$$

so that

$$-\|\boldsymbol{X}_i(t)\|^2 \leq \langle \boldsymbol{A}\boldsymbol{X}_i(t), \boldsymbol{X}_i(t)\rangle \leq \|\boldsymbol{X}_i(t)\|^2.$$

Hence, we deduce that

$$4\|\boldsymbol{X}_i(t)\|^{4+\alpha}(-2-\epsilon) \leq \boldsymbol{Y}_i'(t) \leq 4\|\boldsymbol{X}_i(t)\|^{4+\alpha}(-\epsilon). \tag{14}$$

Multiply both sides of equation 14 with $-\frac{\alpha}{4}\|\boldsymbol{X}_i(t)\|^{-4-\alpha} = -\frac{\alpha}{4}\boldsymbol{Y}_i(t)^{-1-\alpha/4}$, and by noting that $-\frac{\alpha}{4}\boldsymbol{Y}_i'\boldsymbol{Y}_i^{-1-\alpha/4} = (\boldsymbol{Y}_i^{-\alpha/4})'$, we have

$$\alpha(2+\epsilon) \geq (\boldsymbol{Y}_i^{-\alpha/4})' \geq \alpha\epsilon.$$

Integrate from 0 to T and rearranging the appropriate terms, we get

$$(2+\epsilon)\alpha T + \boldsymbol{Y}_i(0)^{-\alpha/4} \geq \boldsymbol{Y}_i(T)^{-\alpha/4} \geq \epsilon\alpha T + \boldsymbol{Y}_i(0)^{-\alpha/4}.$$

Finally, by noting that $\boldsymbol{Y}_i^{\frac{-\alpha}{4}} = \|\boldsymbol{X}_i\|^{-\alpha}$, we easily get the bound equation 10

$$\left((2+\epsilon)\alpha T + \|\boldsymbol{X}_i(0)\|^{-\alpha}\right)^{\frac{-1}{\alpha}} \leq \|\boldsymbol{X}_i(T)\| \leq \left(\epsilon\alpha T + \|\boldsymbol{X}_i(0)\|^{-\alpha}\right)^{\frac{-1}{\alpha}}.$$

□

B PHYSICAL INTERPRETATION OF GRAND

Informally, diffusion describes the process by which a diffusing material is transported from a region of higher to lower density through random microscopic motions. A natural example is the process of heat diffusion, which occurs when a hot object touches a cold object. Heat will diffuse between them until both objects are of the same temperature.

GRAND approaches deep learning on graphs as a continuous diffusion process and treat GNNs as discretisations of an underlying PDE. In doing so, the dynamic in GRAND is the same as that in (heat) diffusion problems. We use this viewpoint to consider the over-smoothing issue.

Consider each node in a given graph as a point in space containing some amount of thermal energy. Each pair of points is connected through a heat pipe. The thermal conductivity (also known as the diffusivity) of each pipe is specific to each pair of points. A thermal conductivity of 0 represents two points that are not connected in the graph. The exact formulation of graph diffusion up to equation 4 represents a closed system. That is, the total amount of energy in all nodes is invariant of time.

In node classification problems, nodes within the same class are often hypothesized as sharing strong connection with each other. This is known as the homophily assumption, i.e. 'like attracts like'. For example, friends are more likely to share an interest, and papers from the same research area tends to cite each other. Homophily is a key principle of many real-world networks, and its effect on GNNs has gained traction as a research direction (Zhu et al., 2020; Yan et al., 2021; Chen et al., 2022; Luan et al., 2022; Ma et al., 2022).

In the context of GRAND, we can assume nodes within the same class as sharing a highly conductive heat pipe. As such, thermal energy is transferred effectively between them, quickly bridging any gap in temperature. This allows for rapid clustering of nodes into classes of different thermal energy levels, which can then be passed into a fully connected layer to perform classification.

This interpretation gives a surprising intuitive explanation to the occurrence of the over-smoothing issue in GRAND. If the dynamic carries on for too long, the energy level of every node will exponentially converge to the average thermal energy, given that the graph is sufficiently connected. Hence, we argue that GRAND (and more generally, any GNN based purely on diffusion) is inherently prone to suffer from over-smoothing.

C ON NEURAL ODES

Traditional neural networks such as residual neural networks, normalizing flows, recurrent-neural-networks (RNNs) learn complicated mappings via the composition of multiple transformations to the hidden states. For example, the residual network updates the future hidden state using the following equation:

$$\boldsymbol{h}_{t+1} = \boldsymbol{h}_t + f(\boldsymbol{h}_t, t, \theta) \tag{15}$$

Where $t \in \{1, \ldots, T\}$ is the layer index of the neural network and $\boldsymbol{h}_t$ is the hidden state at layer t. This iterative update rule can be seen as the Euler discretisation of a continuous transformation.

Neural ODEs (Chen et al., 2018) are a class of continuous-depth (or continuous-time) neural networks where the transformation step t is infinitesimally small. Specifically, the hidden state $\boldsymbol{h}_t$ is parameterized using a continuous dynamics with respect to time t:

$$\frac{d\boldsymbol{h}_t}{dt} = f(\boldsymbol{h}_t, t, \theta) \tag{16}$$

Where $f(\boldsymbol{h}_t, t, \theta)$ is specified by a neural network parameterized by θ. Starting from the initial state $\boldsymbol{h}(0)$, Neural ODEs learn the final representation $\boldsymbol{h}(T)$ by solving 16 using a numerical integrator (often with an adaptive step-size solver or an adaptive solver for short) given an error tolerance. Integrating 16 from 0 to T in a single forward pass requires the adaptive solver to evaluate $f(\boldsymbol{h}_t, t, \theta)$ at multiple time-steps. The computational complexity of the forward pass is determined by the number of function evaluations.

The only problem with updating the hidden state by solving 16 numerically is that the black-box ODE solver is not a differentiable operation. Therefore, the usual back-propagation method for optimizing traditional neural networks does not work for Neural ODEs.

The adjoint sensitivity method (or adjoint method) is a memory-efficient alternative of the traditional back-propagation method for training Neural ODEs. We denote $\boldsymbol{h}(T)$ as the prediction of Neural ODEs and the loss between $\boldsymbol{h}(T)$ and the ground truth is $\mathcal{L}$. Then, we define the adjoint state as $\boldsymbol{a}(t) = \partial\mathcal{L}/\partial\boldsymbol{h}(t)$, we have:

$$\frac{d\mathcal{L}}{d\theta} = \int_0^T \boldsymbol{a}(t)^T \frac{\partial f(\boldsymbol{h}(t), t, \theta)}{\partial\theta} dt \tag{17}$$

Where the adjoint state $\boldsymbol{a}(t)$ satisfies the following dynamics:

$$\frac{d\boldsymbol{a}(t)}{dt} = -\boldsymbol{a}(t)^T \frac{\partial f(\boldsymbol{h}(t), t, \theta)}{\partial\boldsymbol{h}(t)} \tag{18}$$

Since the adjoint state in 18 can be solved numerically using an ODE solver, the gradient of the loss function $\mathcal{L}$ with respect to θ in 17 can be evaluated. The computational complexity of the backward pass is determined by the number of function evaluations used by the ODE solver to solve for the adjoint state.

D EULER DISCRETIZATION OF GRAND ALSO SUFFERS FROM OVER-SMOOTHING

Recall that the forward Euler discretization of the GRAND dynamic was given by Thorpe et al. (2022) as

$$\boldsymbol{X}(k\delta_t) = \boldsymbol{X}((k-1)\delta_t) + \delta_t(\boldsymbol{A} - \boldsymbol{I})\boldsymbol{X}((k-1)\delta_t), \tag{19}$$

where $1 > \delta_t > 0$ is the fixed step size, $k = 1, 2, \ldots, K$ denotes the layers from 1 to K and $\boldsymbol{X}_k := \boldsymbol{X}(k\delta_t)$ is the node feature at the k-th layer.

We mirror our analysis as in Section 3 almost completely. Some calculations will be omitted for clarity. First, we give an analogous definition to 1 for discrete GNNs.

Definition 2. *Let $\boldsymbol{X}_k \in \mathbb{R}^{n\times d}$ denote the feature representation at the k-th layer of some discrete GNN dynamic. $(\boldsymbol{X}_k)$ is said to experience over-smoothing if there exists a vector $\boldsymbol{v} \in \mathbb{R}^d$ and constants $C_1, C_2 > 0$ such that for $\boldsymbol{V} = (\boldsymbol{v}, \boldsymbol{v}, \ldots, \boldsymbol{v})^\top$*

$$\|\boldsymbol{X}_k - \boldsymbol{V}\|_\infty \leq C_1 e^{-C_2 k}. \tag{20}$$

Utilising Proposition 1, we can show that

Proposition 4. *With the dynamic given by equation 5 and equation 19, $(\boldsymbol{X}_k)$ experiences over-smoothing.*

Proof. Let $\boldsymbol{J} = \boldsymbol{P}^{-1}\boldsymbol{A}\boldsymbol{P}$ and $\boldsymbol{Z}_k = \boldsymbol{P}^{-1}\boldsymbol{X}_k$ as in Proposition 1, we can rewrite equation 19 as

$$\boldsymbol{Z}_k = \boldsymbol{Z}_{k-1} + \delta_t(\boldsymbol{J} - \boldsymbol{I})\boldsymbol{Z}_{k-1} = ((1-\delta_t)\boldsymbol{I} + \delta_t\boldsymbol{J})\,\boldsymbol{Z}_{k-1} = ((1-\delta_t)\boldsymbol{I} + \delta_t\boldsymbol{J})^k\,\boldsymbol{Z}_0, \tag{21}$$

where

$$((1-\delta_t)\boldsymbol{I} + \delta_t\boldsymbol{J})^k = \begin{pmatrix} 1 & 0 & \cdots & 0 \\ & ((1-\delta_t)\boldsymbol{I} + \delta_t\boldsymbol{J}_1)^k & \ddots & \vdots \\ & & \ddots & \vdots \\ & & & 0 \\ & & & ((1-\delta_t)\boldsymbol{I} + \delta_m\boldsymbol{J})^k \end{pmatrix}.$$

Let $\overline{\boldsymbol{Z}}$ be the matrix with the same size as $\boldsymbol{Z}_0$ obtained by setting every entry in all but the first row of $\boldsymbol{Z}_0$ be equal to 0. We have

$$\boldsymbol{Z}_k - \overline{\boldsymbol{Z}} = \begin{pmatrix} 0 & 0 & \cdots & 0 \\ & ((1-\delta_t)\boldsymbol{I} + \delta_t\boldsymbol{J}_1)^k & \ddots & \vdots \\ & & \ddots & \vdots \\ & & & 0 \\ & & & ((1-\delta_t)\boldsymbol{I} + \delta_m\boldsymbol{J})^k \end{pmatrix} \boldsymbol{Z}_0.$$

Each block $(1-\delta_t)\boldsymbol{I}+\delta_t\boldsymbol{J}_1$ has spectral radius lesser than 1. Hence, every non-zero entry of $\boldsymbol{Z}_k-\overline{\boldsymbol{Z}}$ can be bounded above in norm by an exponential term of the form $c_1e^{-c_2t}$ for some $c_1, c_2 > 0$. We can deduce that there exists some $C_1, C_2 > 0$ such that

$$\|\boldsymbol{X}_k - \boldsymbol{P}\overline{\boldsymbol{Z}}\|_\infty \leq C_1e^{-C_2k}. \tag{22}$$

Recall that the first column of $\boldsymbol{P}$ is $\boldsymbol{u}^\top = (1, 1, \ldots, 1)^\top$. Set $\boldsymbol{v}$ to be the first row of $\boldsymbol{Z}_0$. We can rewrite equation 22 as

$$\|\boldsymbol{X}_k - \boldsymbol{V}\|_\infty \leq C_1e^{-C_2k}.$$

The claim that $\boldsymbol{v} \in \mathbb{R}^d$ follows from the fact that it is the limit of real-valued vectors with regards to the $\|\cdot\|_\infty$ norm. □