ETH
Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

Seminar for Applied Mathematics SAM

Gradient Gating for Deep Multi-Rate Learning on Graphs

T. K. Rusch and B. P. Chamberlain and M. W. Mahoney and M.

M. Bronstein and S. Mishra

Research Report No. 2022-41

October 2022
Latest revision: March 2023

Seminar für Angewandte Mathematik
Eidgenössische Technische Hochschule
CH-8092 Zürich
Switzerland

Funding ERC: 770880 COMANFLO

GRADIENT GATING FOR DEEP MULTI-RATE LEARNING ON GRAPHS

T. Konstantin Rusch
ETH Zürich, ICSI and UC Berkeley
konstantin.rusch@sam.math.ethz.ch

Benjamin P. Chamberlain
Charm Therapeutics

Michael W. Mahoney
ICSI, LBNL, and UC Berkeley

Michael M. Bronstein
University of Oxford

Siddhartha Mishra
ETH Zürich

ABSTRACT

We present Gradient Gating ($\mathbf{G}^2$), a novel framework for improving the performance of Graph Neural Networks (GNNs). Our framework is based on gating the output of GNN layers with a mechanism for multi-rate flow of message passing information across nodes of the underlying graph. Local gradients are harnessed to further modulate message passing updates. Our framework flexibly allows one to use any basic GNN layer as a wrapper around which the multi-rate gradient gating mechanism is built. We rigorously prove that $\mathbf{G}^2$ alleviates the oversmoothing problem and allows the design of deep GNNs. Empirical results are presented to demonstrate that the proposed framework achieves state-of-the-art performance on a variety of graph learning tasks, including on large-scale heterophilic graphs.

1 INTRODUCTION

Learning tasks involving graph structured data arise in a wide variety of problems in science and engineering. Graph Neural Networks (GNNs) (Sperduti, 1994; Goller & Kuchler, 1996; Sperduti & Starita, 1997; Frasconi et al., 1998; Gori et al., 2005; Scarselli et al., 2008; Bruna et al., 2014; Defferrard et al., 2016; Kipf & Welling, 2017; Monti et al., 2017; Gilmer et al., 2017) are a popular deep learning architecture for graph-structured and relational data. GNNs have been successfully applied in domains including computer vision and graphics (Monti et al., 2017), recommender systems (Ying et al., 2018), transportation (Derrow-Pinion et al., 2021), computational chemistry (Gilmer et al., 2017), drug discovery (Gaudelet et al., 2021), particle physics (Shlomi et al., 2020) and social networks. See Zhou et al. (2019); Bronstein et al. (2021) for extensive reviews.

Despite the widespread success of GNNs and a plethora of different architectures, several fundamental problems still impede their efficiency on realistic learning tasks. These include the bottleneck (Alon & Yahav, 2021), oversquashing (Topping et al., 2021), and oversmoothing (Nt & Maehara, 2019; Oono & Suzuki, 2020) phenomena. Oversmoothing refers to the observation that all node features in a deep (multi-layer) GNN converge to the same constant value as the number of layers is increased. Thus, and in contrast to standard machine learning frameworks, oversmoothing inhibits the use of very deep GNNs for learning tasks. These phenomena are likely responsible for the unsatisfactory empirical performance of traditional GNN architectures in *heterophilic* datasets, where the features or labels of a node tend to be different from those of its neighbors (Zhu et al., 2020).

Given this context, our main goal is to present a novel framework that alleviates the oversmoothing problem and allows one to implement very deep multi-layer GNNs that can significantly improve performance in the setting of heterophilic graphs. Our starting point is the observation that in standard Message-Passing GNN architectures (MPNNs), such as GCN (Kipf & Welling, 2017) or GAT (Velickovic et al., 2018), each node gets updated at exactly the *same rate* within every hidden layer. Yet, realistic learning tasks might benefit from having different rates of propagation (flow) of information on the underlying graph. This insight leads to a novel *multi-rate message passing* scheme capable of learning these underlying rates. Moreover, we also propose a novel procedure that harnesses graph gradients to ameliorate the oversmoothing problem. Combining these elements leads to a new architecture described in this paper, which we term **Gradient Gating** ($\mathbf{G}^2$).

Main Contributions. We will demonstrate the following advantages of the proposed approach:

- $\mathbf{G}^2$ is a flexible framework wherein any standard message-passing layer (such as GAT, GCN, GIN, or GraphSAGE) can be used as the coupling function. Thus, it should be thought of as a framework into which one can plug existing GNN components. The use of multiple rates and gradient gating facilitates the implementation of deep GNNs and generally improves performance.
- $\mathbf{G}^2$ can be interpreted as a discretization of a dynamical system governed by nonlinear differential equations. By investigating the stability of zero-Dirichlet energy steady states of this system, we rigorously prove that our gradient gating mechanism prevents oversmoothing. To complement this, we also prove a partial converse, that the lack of gradient gating can lead to oversmoothing.
- We provide extensive empirical evidence demonstrating that $\mathbf{G}^2$ achieves state-of-the-art performance on a variety of graph learning tasks, including on large heterophilic graph datasets.

2 GRADIENT GATING

Let $\mathcal{G} = (\mathcal{V}, \mathcal{E} \subseteq \mathcal{V} \times \mathcal{V})$ be an undirected graph with $|\mathcal{V}| = v$ nodes and $|\mathcal{E}| = e$ edges (unordered pairs of nodes $\{i, j\}$ denoted $i \sim j$). The 1-*neighborhood* of a node i is denoted $\mathcal{N}_i = \{j \in \mathcal{V} : i \sim j\}$. Furthermore, each node i is endowed with an m-dimensional feature vector $\mathbf{X}_i$; the node features are arranged into a $v \times m$ matrix $\mathbf{X} = (\mathbf{X}_{ik})$ with $i = 1, \dots, v$ and $k = 1, \dots, m$.

A typical residual Message-Passing GNN (MPNN) updates the node features by performing several iterations of the form,

$$\mathbf{X}^n = \mathbf{X}^{n-1} + \sigma(\mathbf{F}_\theta(\mathbf{X}^{n-1}, \mathcal{G})), \tag{1}$$

where $\mathbf{F}_\theta$ is a *learnable* function with parameters θ, and σ is an element-wise non-linear activation function. Here $n \geq 1$ denotes the n-th hidden layer with $n = 0$ being the input.

One can interpret (1) as a discrete dynamical system in which $\mathbf{F}$ plays the role of a *coupling function* determining the interaction between different nodes of the graph. In particular, we consider local (1-neighborhood) coupling of the form $\mathbf{Y}_i = (\mathbf{F}(\mathbf{X}, \mathcal{G}))_i = \mathbf{F}(\mathbf{X}_i, \{\!\{\mathbf{X}_{j \in \mathcal{N}_i}\}\!\})$ operating on the multiset of 1-neighbors of each node. Examples of such functions used in the graph machine learning literature (Bronstein et al., 2021) are *graph convolutions* $\mathbf{Y}_i = \sum_{j \in \mathcal{N}_i} c_{ij} \mathbf{X}_j$ (GCN, (Kipf & Welling, 2017)) and *graph attention* $\mathbf{Y}_i = \sum_{j \in \mathcal{N}_i} a(\mathbf{X}_i, \mathbf{X}_j) \mathbf{X}_j$ (GAT, (Velickovic et al., 2018)).

We observe that in (1), at each hidden layer, every node and every feature channel gets updated with exactly the same rate. However, it is reasonable to expect that in realistic graph learning tasks one can encounter multiple rates for the flow of information (node updates) on the graph. Based on this observation, we propose a **multi-rate (MR)** generalization of (1), allowing updates to each node of the graph and feature channel with different rates,

$$\mathbf{X}^n = (1 - \boldsymbol{\tau}^n) \odot \mathbf{X}^{n-1} + \boldsymbol{\tau}^n \odot \sigma(\mathbf{F}_\theta(\mathbf{X}^{n-1}, \mathcal{G})), \tag{2}$$

where $\boldsymbol{\tau}$ denotes a $v \times m$ matrix of rates with elements $\boldsymbol{\tau}_{ik} \in [0, 1]$. Rather than fixing $\boldsymbol{\tau}$ prior to training, we aim to learn the different update rates based on the node data $\mathbf{X}$ and the local structure of the underlying graph $\mathcal{G}$, as follows

$$\boldsymbol{\tau}^n(\mathbf{X}^{n-1}, \mathcal{G}) = \bar{\sigma}(\hat{\mathbf{F}}_{\hat{\theta}}(\mathbf{X}^{n-1}, \mathcal{G})), \tag{3}$$

where $\hat{\mathbf{F}}_{\hat{\theta}}$ is another learnable 1-neighborhood coupling function, and $\bar{\sigma}$ is a sigmoidal logistic activation function to constrain the rates to lie within $[0, 1]$. Since the multi-rate message-passing scheme (2) using (3) does not necessarily prevent oversmoothing (for any choice of the coupling function), we need to further constrain the rate matrix $\boldsymbol{\tau}^n$. To this end, we note that the *graph gradient* of scalar node features $\mathbf{y}$ on the underlying graph $\mathcal{G}$ is defined as $(\nabla \mathbf{y})_{ij} = \mathbf{y}_j - \mathbf{y}_i$ at the edge $i \sim j$ (Lim, 2015). Next, we will use graph gradients to obtain the proposed **Gradient Gating** ($\mathbf{G}^2$) framework given by

$$\begin{aligned}
\hat{\boldsymbol{\tau}}^n &= \sigma(\hat{\mathbf{F}}_\theta(\mathbf{X}^{n-1}, \mathcal{G})), \\
\boldsymbol{\tau}^n_{ik} &= \tanh\left(\sum_{j \in \mathcal{N}_i} |\hat{\boldsymbol{\tau}}^n_{jk} - \hat{\boldsymbol{\tau}}^n_{ik}|^p\right), \\
\mathbf{X}^n &= (1 - \boldsymbol{\tau}^n) \odot \mathbf{X}^{n-1} + \boldsymbol{\tau}^n \odot \sigma(\mathbf{F}_\theta(\mathbf{X}^{n-1}, \mathcal{G})),
\end{aligned} \tag{4}$$

where $\hat{\boldsymbol{\tau}}^n_{jk} - \hat{\boldsymbol{\tau}}^n_{ik} = (\nabla \hat{\boldsymbol{\tau}}^n_{*k})_{ij}$ denotes the graph-gradient and $\hat{\boldsymbol{\tau}}^n_{*k}$ is the k-th column of the rate matrix $\hat{\boldsymbol{\tau}}^n$ and $p \geq 0$. Since $\sum_{j \in \mathcal{N}_i} |\hat{\boldsymbol{\tau}}^n_{jk} - \hat{\boldsymbol{\tau}}^n_{ik}|^p \geq 0$ for all $i \in \mathcal{V}$, it follows that $\boldsymbol{\tau}^n \in [0,1]^{v \times m}$ for all n, retaining its interpretation as a matrix of rates. The sum over the neighborhood $\mathcal{N}_i$ in (4) can be replaced by any permutation-invariant aggregation function (e.g., mean or max). Moreover, any standard message-passing procedure can be used to define the coupling functions $\mathbf{F}$ and $\hat{\mathbf{F}}$ (and, in particular, one can set $\hat{\mathbf{F}} = \mathbf{F}$). As an illustration, Fig. 1 shows a schematic diagram of the layer-wise update of the proposed $\mathbf{G}^2$ architecture.

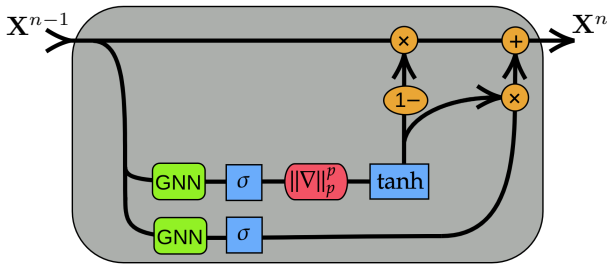

Figure 1: Schematic diagram of $\mathbf{G}^2$ (4) showing the layer-wise update of the latent node features $\mathbf{X}$ (at layer n). The norm of the graph-gradient (i.e., sum in second equation in (4)) is denoted as $\|\nabla\|_p^p$.

The intuitive idea behind gradient gating in (4) is the following: If for any node $i \in \mathcal{V}$ local oversmoothing occurs, i.e., $\lim_{n\to\infty} \sum_{j \in \mathcal{N}_i} \|\mathbf{X}^n_i - \mathbf{X}^n_j\| = 0$, then $\mathbf{G}^2$ ensures that the corresponding rate $\boldsymbol{\tau}^n_i$ goes to zero (at a faster rate), such that the underlying hidden node feature $\mathbf{X}_i$ is no longer updated. This prevents oversmoothing by *early-stopping* of the message passing procedure.

3 Properties of $\mathbf{G}^2$-GNN

$\mathbf{G}^2$ is a flexible framework. An important aspect of $\mathbf{G}^2$ (4) is that it can be considered as a "wrapper" around any specific MPNN architecture. In particular, the hidden layer update for *any* form of message passing (e.g., GCN (Kipf & Welling, 2017), GAT (Velickovic et al., 2018), GIN (Xu et al., 2018) or GraphSAGE (Hamilton et al., 2017)) can be used as the coupling functions $\mathbf{F}, \hat{\mathbf{F}}$ in (4). By setting $\boldsymbol{\tau} \equiv \mathbf{I}$, (4) reduces to

$$\mathbf{X}^n = \sigma\left(\mathbf{F}_\theta(\mathbf{X}^{n-1}, \mathcal{G})\right), \tag{5}$$

a standard (non-residual) MPNN. As we will show in the following, the use of a non-trivial gradient-gated *learnable* rate matrix $\boldsymbol{\tau}$ allows implementing very deep architectures that avoid oversmoothing.

Maximum Principle for node features. Node features produced by $\mathbf{G}^2$ satisfy the following Maximum Principle.

Proposition 3.1. *Let $\mathbf{X}^n$ be the node feature matrix generated by iteration formula* (4)*. Then, the features are bounded as follows:*

$$\min\left(-1, \underline{\sigma}\right) \leq \mathbf{X}^n_{ik} \leq \max\left(1, \overline{\sigma}\right), \quad \forall 1 \leq n, \tag{6}$$

where the scalar activation function is bounded by $\underline{\sigma} \leq \sigma(z) \leq \overline{\sigma}$ for all $z \in \mathbb{R}$.

The proof follows readily from writing (4) component-wise and using the fact that $0 \leq \boldsymbol{\tau}^n_{ik} \leq 1$, for all $1 \leq i \leq v$, $1 \leq k \leq m$ and $1 \leq n$.

Continuous limit of $\mathbf{G}^2$. It has recently been shown (see Avelar et al. (2019); Poli et al. (2019); Zhuang et al. (2020); Xhonneux et al. (2020); Chamberlain et al. (2021a); Eliasof et al. (2021); Chamberlain et al. (2021b); Topping et al. (2021); Rusch et al. (2022a) and references therein)

that interesting properties of GNNs (with residual connections) can be understood by taking the continuous (infinite-depth) limit and analyzing the resulting differential equations.

In this context, we can derive a continuous version of (4) by introducing a small-scale $0 < \Delta t < 1$ and rescaling the rate matrix $\boldsymbol{\tau}^n$ to $\Delta t \boldsymbol{\tau}^n$ leading to

$$\mathbf{X}^n = (1 - \Delta t \boldsymbol{\tau}^n) \odot \mathbf{X}^{n-1} + \Delta t \boldsymbol{\tau}^n \odot \sigma\left(\mathbf{F}_\theta(\mathbf{X}^{n-1}, \mathcal{G})\right). \tag{7}$$

Rearranging the terms in (7), we obtain

$$\frac{\mathbf{X}^n - \mathbf{X}^{n-1}}{\Delta t} = \boldsymbol{\tau}^n \odot \left(\sigma\left(\mathbf{F}_\theta(\mathbf{X}^{n-1}, \mathcal{G})\right) - \mathbf{X}^{n-1}\right). \tag{8}$$

Interpreting $\mathbf{X}^n \approx \mathbf{X}(n\Delta t) = \mathbf{X}(t^n)$, i.e., marching in time, corresponds to increasing the number of hidden layers. Letting $\Delta t \to 0$, one obtains the following system of graph-coupled ordinary differential equations (ODEs):

$$\begin{aligned}
\frac{d\mathbf{X}(t)}{dt} &= \boldsymbol{\tau}(t) \odot \left(\sigma\left(\mathbf{F}_\theta(\mathbf{X}(t), \mathcal{G})\right) - \mathbf{X}(t)\right), \quad \forall t \geq 0, \\
\boldsymbol{\tau}_{ik}(t) &= \tanh\left(\sum_{j \in \mathcal{N}_i} |\hat{\boldsymbol{\tau}}_{ik}(t) - \hat{\boldsymbol{\tau}}_{jk}(t)|^p\right), \\
\hat{\boldsymbol{\tau}}(t) &= \hat{\sigma}(\hat{\mathbf{F}}_{\hat{\theta}}(\mathbf{X}^{n-1}, \mathcal{G})).
\end{aligned} \tag{9}$$

We observe that the iteration formula (4) acts as a *forward Euler* discretization of the ODE system (9). Hence, one can follow Chamberlain et al. (2021a) and design more general (e.g., higher-order, adaptive, or implicit) discretizations of the ODE system (9). All these can be considered as design extensions of (4).

Oversmoothing. Using the interpretation of (4) as a discretization of the ODE system (9), we can adapt the mathematical framework recently proposed in Rusch et al. (2022a) to study the oversmoothing problem. In order to formally define oversmoothing, we introduce the *Dirichlet energy* defined on the node features $\mathbf{X}$ of an undirected graph $\mathcal{G}$ as

$$\mathcal{E}(\mathbf{X}) = \frac{1}{v} \sum_{i \in \mathcal{V}} \sum_{j \in \mathcal{N}_i} \|\mathbf{X}_i - \mathbf{X}_j\|^2. \tag{10}$$

Following Rusch et al. (2022a), we say that the scheme (9) *oversmoothes* if the Dirichlet energy decays *exponentially fast*,

$$\mathcal{E}(\mathbf{X}(t)) \leq C_1 e^{-C_2 t}, \quad \forall t > 0, \tag{11}$$

for some $C_{1,2} > 0$. In particular, the discrete version of (11) implies that oversmoothing happens when the Dirichlet energy, decays exponentially fast as the number of hidden layers increases ((Rusch et al., 2022a) Definition 3.2).

Next, one can prove the following proposition further characterizing oversmoothing with the standard terminology of dynamical systems (Wiggins, 2003).

Proposition 3.2. *The oversmoothing problem occurs for the ODEs* (9) *iff the hidden states* $\mathbf{X}_i^* = \mathbf{c}$*, for all* $i \in \mathcal{V}$ *are* exponentially stable steady states (fixed points) *of the ODE* (9)*, for some* $\mathbf{c} \in \mathbb{R}^m$.

In other words, for the oversmoothing problem to occur for this system, all the trajectories of the ODE (9) that start within the corresponding basin of attraction have to converge exponentially fast in time (according to (11)) to the corresponding steady state $\mathbf{c}$. Note that the basins of attraction will be different for different values of $\mathbf{c}$. The proof of this Proposition is a straightforward adaptation of the proof of Proposition 3.3 of Rusch et al. (2022a).

Given this precise formulation of oversmoothing, we will investigate whether and how gradient gating in (9) can prevent oversmoothing. For simplicity, we set $m = 1$ to consider only scalar node features (extension to vector node features is straightforward). Moreover, we assume coupling functions of the form $\mathbf{F}(\mathbf{X}) = \mathbf{A}(\mathbf{X})\mathbf{X}$, expressed element-wise as (see also Chamberlain et al. (2021a); Rusch et al. (2022a)),

$$(\mathbf{F}(\mathbf{X}))_i = \sum_{j \in \mathcal{N}_i} \mathbf{A}(\mathbf{X}_i, \mathbf{X}_j)\mathbf{X}_j. \tag{12}$$

Here, $\mathbf{A}(\mathbf{X})$ is a matrix-valued function whose form covers many commonly used coupling functions stemming from the graph attention (GAT, where $\mathbf{A}_{ij} = \mathbf{A}(\mathbf{X}_i, \mathbf{X}_j)$ is learnable) or convolution operators (GCN, where $\mathbf{A}_{ij}$ is fixed). Furthermore, the matrices are *right stochastic*, i.e., the entries satisfy

$$0 \leq \mathbf{A}_{ij} \leq 1, \ \sum_{j \in \mathcal{N}_i} \mathbf{A}_{ij} \quad = 1. \tag{13}$$

Finally, as the multi-rate feature of (9) has no direct bearing on the oversmoothing problem, we focus on the contribution of the gradient feedback term. To this end, we deactivate the multi-rate aspects and assume that $\hat{\boldsymbol{\tau}}_i = \mathbf{X}_i$ for all $i \in \mathcal{V}$, leading to the following form of (9):

$$\begin{aligned} \frac{d\mathbf{X}_i(t)}{dt} &= \boldsymbol{\tau}_i(t) \left(\sigma \left(\sum_{j \in \mathcal{N}_i} \mathbf{A}_{ij} \mathbf{X}_j(t) \right) - \mathbf{X}_i(t) \right), \quad \forall t \geq 0, \\ \boldsymbol{\tau}_i(t) &= \tanh \left(\sum_{j \in \mathcal{N}_i} \|\mathbf{X}_j(t) - \mathbf{X}_i(t)\|^p \right). \end{aligned} \tag{14}$$

Lack of $\mathbf{G}^2$ can lead to oversmoothing. We first consider the case where the Gradient Gating is switched off by setting $p = 0$ in (14). This yields a standard GNN in which node features are evolved through message passing between neighboring nodes, without any explicit information about graph gradients. We further assume that the activation function is ReLU i.e., $\sigma(x) = \max(x, 0)$. Given this setting, we have the following proposition on oversmoothing:

Proposition 3.3. *Assume the underlying graph $\mathcal{G}$ is connected. For any $c \geq 0$, let $\mathbf{X}_i^* \equiv c$, for all $i \in \mathcal{V}$ be a (zero-Dirichlet energy) steady state of the ODEs* (14)*. Moreover, assume no Gradient Gating ($p = 0$ in* (14)*) and*

$$\mathbf{A}_{ij}(c, c) = \mathbf{A}_{ji}(c, c), \text{ and } \mathbf{A}_{ij}(c, c) \geq \underline{a}, \quad 1 \leq i, j \leq v, \tag{15}$$

with $0 < \underline{a} \leq 1$ and that there exists at least one node denoted w.l.o.g. with index 1 such that $\mathbf{X}_1(t) \equiv c$, for all $t \geq 0$. Then, the steady state $\mathbf{X}_i^* = \mathbf{c}$, for all $i \in \mathcal{V}$, of (14) *is* exponentially stable.

Proposition 3.2 implies that without gradient gating ($\mathbf{G}^2$), (9) can lead to oversmoothing. The proof, presented in **SM** C.1 relies on analyzing the time-evolution of small perturbations around the steady state $\mathbf{c}$ and showing that these perturbations decay exponentially fast in time (see (20)).

$\mathbf{G}^2$ prevents oversmoothing. We next investigate the effect of Gradient Gating in the same setting of Proposition 3.3. The following Proposition shows that gradient gating prevents oversmoothing:

Proposition 3.4. *Assume the underlying graph $\mathcal{G}$ is connected. For any $c \geq 0$ and for all $i \in \mathcal{V}$, let $\mathbf{X}_i^* \equiv c$ be a (zero-Dirichlet energy) steady state of the ODEs* (14)*. Moreover, assume Gradient Gating ($p > 0$) and that the matrix $\mathbf{A}$ in* (14) *satisfies* (15) *and that there exists at least one node denoted w.l.o.g. with index 1 such that $\mathbf{X}_1(t) \equiv c$, for all $t \geq 0$. Then, the steady state $\mathbf{X}_i^* = \mathbf{c}$, for all $i \in \mathcal{V}$ is* not exponentially stable.

The proof, presented in **SM** C.2 clearly elucidates the role of gradient gating by showing that the energy associated with the quasi-linearized evolution equations (**SM** Eqn. (21)) is balanced by two terms (**SM** Eqn. (23)), both resulting from the introduction of gradient gating by setting $p > 0$ in (14). One of them is of indefinite sign and can even cause *growth* of perturbations around a steady state $\mathbf{c}$. The other decays initial perturbations. However, the rate of this decay is at most *polynomial* (**SM** Eqn. (28)). For instance, the decay is merely linear for $p = 2$ and slower for higher values of p. Thus, the steady state $\mathbf{c}$ cannot be exponentially stable and oversmoothing is prevented. This justifies the intuition behind gradient gating, namely, if oversmoothing occurs around a node i, i.e., $\lim_{n \to \infty} \sum_{j \in \mathcal{N}_i} \|\mathbf{X}_i^n - \mathbf{X}_j^n\| = 0$, then the corresponding rate $\boldsymbol{\tau}_i^n$ goes to zero (at a faster rate), such that the underlying hidden node feature $\mathbf{X}_i$ stops getting updated.

4 Experimental Results

In this section, we present an experimental study of $\mathbf{G}^2$ on both synthetic and real datasets. We use $\mathbf{G}^2$ with three different coupling functions: GCN (Kipf & Welling, 2017), GAT (Velickovic et al., 2018) and GraphSAGE (Hamilton et al., 2017).

Effect of $\mathbf{G}^2$ on Dirichlet energy. Given that oversmoothing relates to the decay of Dirichlet energy (11), we follow the experimental setup proposed by Rusch et al. (2022a) to probe the dynamics of the Dirichlet energy of Gradient-Gated GNNs, defined on a 2-dimensional 10×10 regular grid with 4-neighbor connectivity. The node features $\mathbf{X}$ are randomly sampled from $\mathcal{U}([0, 1])$ and then propagated through a 1000-layer GNN with random weights. We compare GAT, GCN and their gradient-gated versions ($\mathbf{G}^2$-GAT and $\mathbf{G}^2$-GCN) in this experiment. Fig. 2 depicts on log-log scale the Dirichlet energy of each layer's output with respect to the layer number. We clearly observe that GAT and GCN *oversmooth* as the underlying Dirichlet energy converges exponentially fast to zero, resulting in the node features becoming indistinguishable. In practice, the Dirichlet energy for these architectures is ≈ 0 after just ten hidden layers. On the other hand, and as suggested by the theoretical results of the previous section, adding $\mathbf{G}^2$ decisively prevents this behavior and the Dirichlet energy remains (near) constant, even for very deep architectures (up to 1000 layers).

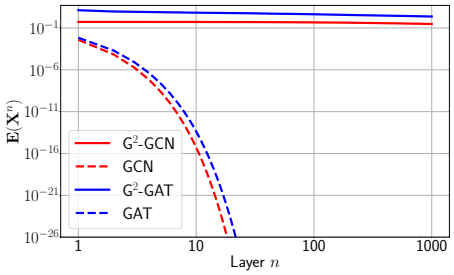

Figure 2: Dirichlet energy $\mathcal{E}(\mathbf{X}^n)$ of layer-wise node features $\mathbf{X}^n$ propagated through a GAT, GCN and their gradient gated versions ($\mathbf{G}^2$-GAT, $\mathbf{G}^2$-GCN).

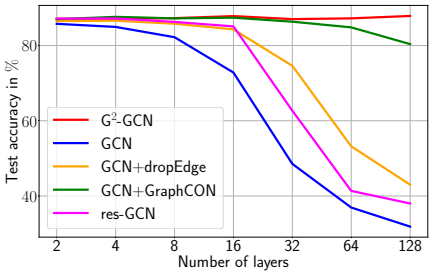

Figure 3: Test accuracies of GCN with gradient gating ($\mathbf{G}^2$-GCN) as well as plain GCN and GCN combined with other methods on the Cora dataset for increasing number of layers.

$\mathbf{G}^2$ for very deep GNNs. Oversmoothing inhibits the use of large number of GNN layers. As $\mathbf{G}^2$ is designed to alleviate oversmoothing, it should allow very deep architectures. To test this assumption, we reproduce the experiment considered in Chamberlain et al. (2021a): a node-level classification task on the Cora dataset using increasingly deeper GCN architectures. In addition to $\mathbf{G}^2$, we also compare with two recently proposed mechanisms to alleviate oversmoothing, DropEdge (Rong et al., 2020) and GraphCON (Rusch et al., 2022a). The results are presented in Fig. 3, where we plot the test accuracy for all the models with the number of layers ranging from 2 to 128. While a plain GCN seems to suffer the most from oversmoothing (with the performance rapidly deteriorating after 8 layers), GCN+DropEdge as well as GCN+GraphCON are able to mitigate this behavior to some extent, although the performance eventually starts dropping (after 16 and 64 layers, respectively). In contrast, $\mathbf{G}^2$-GCN exhibits a small but noticeable *increase* in performance for increasing number of layers, reaching its peak performance for 128 layers. This experiment suggests that $\mathbf{G}^2$ can indeed be used in conjunction with deep GNNs, potentially allowing performance gains due to depth.

$\mathbf{G}^2$ for multi-scale node-level regression. We test the multi-rate nature of $\mathbf{G}^2$ on node-level regression tasks, where the target node values exhibit multiple scales. Due to a lack of widely available node-level regression tasks, we propose regression experiments based on the Wikipedia article networks Chameleon and Squirrel, (Rozemberczki et al., 2021). While Chameleon and Squirrel are already widely used as heterophilic node-level classification tasks, the original datasets consist of continuous node targets (average monthly web-page traffic). We normalize the provided webpage traffic values for every node between 0 and 1 and note that the resulting node values exhibit values on a wide range of different scales ranging between 10^{-5} and 1 (see Fig. 4). Table 1 shows the test normalized mean-square error (mean and standard deviation based on the ten pre-defined splits in Pei et al. (2020)) for two standard GNN architectures (GCN and GAT) with and without $\mathbf{G}^2$. We observe from

Table 1: Normalized test MSE on multi-scale node-level regression tasks.

	Chameleon	**Squirrel**
#Nodes	2,277	5,201
#Edges	31,421	198,493
GCNII	$\mathbf{0.170 \pm 0.034}$	$\mathbf{0.093 \pm 0.031}$
PairNorm	0.207 ± 0.038	0.140 ± 0.040
GCN	0.207 ± 0.039	0.143 ± 0.039
GAT	0.207 ± 0.038	0.143 ± 0.039
$\mathbf{G}^2$-GCN	$\mathbf{0.137 \pm 0.033}$	$\mathbf{0.070 \pm 0.028}$
$\mathbf{G}^2$-GAT	$\mathbf{0.136 \pm 0.029}$	$\mathbf{0.069 \pm 0.029}$

Table 1 that adding $\mathbf{G}^2$ to the baselines significantly reduces the error, demonstrating the advantage of using multiple update rates.

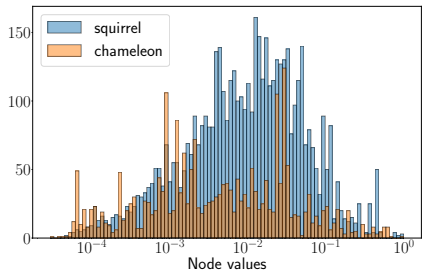

Figure 4: Histogram of the target node values of the Chameleon and Squirrel node-level regression tasks.

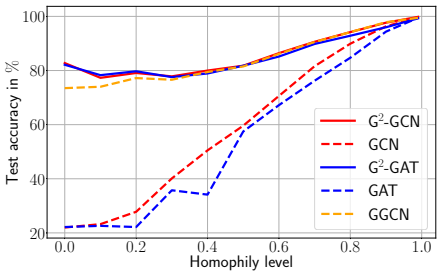

Figure 5: Test accuracy of GCN and GAT with / without gradient gating ($\mathbf{G}^2$) on synthetic Cora with a varying level of true label homophily.

$\mathbf{G}^2$ for varying homophily (Synthetic Cora). We test $\mathbf{G}^2$ on a node-level classification task with varying levels of homophily on the synthetic Cora dataset Zhu et al. (2020). Standard GNN models are known to perform poorly in heterophilic settings. This can be seen in Fig. 5, where we present the classification accuracy of GCN and GAT on the synthetic-Cora dataset with a level of homophily varying between 0 and 0.99. While these models succeed in the homophilic case (reaching nearly perfect accuracy), their performance drops to $\approx 20\%$ when the level of homophily approaches 0. Adding $\mathbf{G}^2$ to GCN or GAT mitigates this phenomenon: the resulting models reach a test accuracy of over 80%, even in the most heterophilic setting, thus leading to a four-fold increase in the accuracy of the underlying GCN or GAT models. Furthermore, we notice an increase in performance even in the homophilic setting. Moreover, we compare with a state-of-the-art model GGCN (Yan et al., 2021), which has been recently proposed to explicitly deal with heterophilic graphs. From Fig. 5 we observe that $\mathbf{G}^2$ performs on par and slightly better than GGCN in strongly heterophilic settings.

Table 2: Results on heterophilic graphs. The three best performing methods are highlighted in **red** (First), **blue** (Second), and **violet** (Third).

	Texas	**Wisconsin**	**Film**	**Squirrel**	**Chameleon**	**Cornell**
Hom level	**0.11**	**0.21**	**0.22**	**0.22**	**0.23**	**0.30**
#Nodes	183	251	7,600	5,201	2,277	183
#Edges	295	466	26,752	198,493	31,421	280
#Classes	5	5	5	5	5	5
GGCN	**84.86 ± 4.55**	86.86 ± 3.29	**37.54 ± 1.56**	55.17 ± 1.58	**71.14 ± 1.84**	85.68 ± 6.63
GPRGNN	78.38 ± 4.36	82.94 ± 4.21	34.63 ± 1.22	31.61 ± 1.24	46.58 ± 1.71	80.27 ± 8.11
H2GCN	**84.86 ± 7.23**	**87.65 ± 4.98**	35.70 ± 1.00	36.48 ± 1.86	60.11 ± 2.15	82.70 ± 5.28
FAGCN	82.43 ± 6.89	82.94 ± 7.95	34.87 ± 1.25	42.59 ± 0.79	55.22 ± 3.19	79.19 ± 9.79
F^2GAT	82.70 ± 5.95	**87.06 ± 4.13**	36.65 ± 1.13	47.32 ± 2.43	67.81 ± 2.05	83.51 ± 6.70
MixHop	77.84 ± 7.73	75.88 ± 4.90	32.22 ± 2.34	43.80 ± 1.48	60.50 ± 2.53	73.51 ± 6.34
GCNII	77.57 ± 3.83	80.39 ± 3.40	**37.44 ± 1.30**	38.47 ± 1.58	63.86 ± 3.04	77.86 ± 3.79
Geom-GCN	66.76 ± 2.72	64.51 ± 3.66	31.59 ± 1.15	38.15 ± 0.92	60.00 ± 2.81	60.54 ± 3.67
PairNorm	60.27 ± 4.34	48.43 ± 6.14	27.40 ± 1.24	50.44 ± 2.04	62.74 ± 2.82	58.92 ± 3.15
LINKX	74.60 ± 8.37	75.49 ± 5.72	36.10 ± 1.55	**61.81 ± 1.80**	68.42 ± 1.38	77.84 ± 5.81
GloGNN	84.32 ± 4.15	**87.06 ± 3.53**	**37.35 ± 1.30**	**57.54 ± 1.39**	**69.78 ± 2.42**	83.51 ± 4.26
GraphSAGE	82.43 ± 6.14	81.18 ± 5.56	34.23 ± 0.99	41.61 ± 0.74	58.73 ± 1.68	75.95 ± 5.01
ResGatedGCN	80.00 ± 5.57	81.57 ± 5.35	36.02 ± 1.19	37.60 ± 1.80	49.82 ± 2.71	73.51 ± 4.95
GCN	55.14 ± 5.16	51.76 ± 3.06	27.32 ± 1.10	31.52 ± 0.71	38.44 ± 1.92	60.54 ± 5.30
GAT	52.16 ± 6.63	49.41 ± 4.09	27.44 ± 0.89	36.77 ± 1.68	48.36 ± 1.58	61.89 ± 5.05
MLP	80.81 ± 4.75	85.29 ± 3.31	36.53 ± 0.70	28.77 ± 1.56	46.21 ± 2.99	81.89 ± 6.40
$\mathbf{G}^2$-GAT	**84.59 ± 5.55**	**87.65 ± 4.64**	37.30 ± 0.87	46.48 ± 1.41	64.12 ± 1.96	**87.30 ± 4.84**
$\mathbf{G}^2$-GCN	**84.86 ± 3.24**	**87.06 ± 3.19**	37.09 ± 1.16	39.62 ± 2.91	55.83 ± 2.88	**86.49 ± 5.27**
$\mathbf{G}^2$-GraphSAGE	**87.57 ± 3.86**	**87.84 ± 3.49**	37.14 ± 1.01	**64.26 ± 2.38**	**71.40 ± 2.38**	**86.22 ± 4.90**

Heterophilic datasets. In Table 2, we test the proposed framework on several real-world heterophilic graphs (with a homophily level of ≤ 0.30) (Pei et al., 2020; Rozemberczki et al., 2021) and benchmark it against baseline models GraphSAGE (Hamilton et al., 2017), GCN (Kipf & Welling,

2017), GAT (Velickovic et al., 2018) and MLP (Goodfellow et al., 2016), as well as recent state-of-the-art models on heterophilic graph datasets, i.e., GGCN (Yan et al., 2021), GPRGNN (Chien et al., 2020), H2GCN (Zhu et al., 2020), FAGCN (Bo et al., 2021), F^2GAT (Wei et al., 2022), MixHop (Abu-El-Haija et al., 2019), GCNII (Chen et al., 2020b), Geom-GCN (Pei et al., 2020), PairNorm (Zhao & Akoglu, 2019). We can observe that $\mathbf{G}^2$ added to GCN, GAT or GraphSAGE outperforms all other methods (in particular recent methods such as GGCN, GPRGNN, H2GCN that were explicitly designed to solve heterophilic tasks). Moreover, adding $\mathbf{G}^2$ to the underlying base GNN model improves the results on average by 45.75% for GAT, 45.4% for GCN and 18.6% for GraphSAGE.

Large-scale graphs. Given the exceptional performance of $\mathbf{G}^2$-GraphSAGE on small and medium sized heterophilic graphs, we test the proposed $\mathbf{G}^2$ (applied to GraphSAGE, i.e., $\mathbf{G}^2$-GraphSAGE) on large-scale datasets. To this end, we consider three different experiments based on large graphs from Lim et al. (2021), which range from highly heterophilic (homophily level of 0.07) to fairly homophilic (homophily level of 0.61). The sizes range from large graphs with $\sim$170K nodes and $\sim$1M edges to a very large graph with $\sim$3M nodes and $\sim$14M edges.

Table 3: Results on large-scale datasets.

	snap-patents	**arXiv-year**	**genius**
Hom level	**0.07**	**0.22**	**0.61**
#Nodes	2,923,922	169,343	421,961
#Edges	13,975,788	1,166,243	984,979
#Classes	5	5	2
MLP	31.34 ± 0.05	36.70 ± 0.21	86.68 ± 0.09
GCN	45.65 ± 0.04	46.02 ± 0.26	87.42 ± 0.37
GAT	45.37 ± 0.44	46.05 ± 0.51	55.80 ± 0.87
MixHop	52.16 ± 0.09	51.81 ± 0.17	90.58 ± 0.16
LINKX	$\mathbf{61.95 \pm 0.12}$	$\mathbf{56.00 \pm 1.34}$	$\mathbf{90.77 \pm 0.27}$
LINK	60.39 ± 0.07	53.97 ± 0.18	73.56 ± 0.14
GCNII	37.88 ± 0.69	47.21 ± 0.28	90.24 ± 0.09
APPNP	32.19 ± 0.07	38.15 ± 0.26	85.36 ± 0.62
GloGNN	$\mathbf{62.09 \pm 0.27}$	$\mathbf{54.68 \pm 0.34}$	$\mathbf{90.66 \pm 0.11}$
GPR-GNN	40.19 ± 0.03	45.07 ± 0.21	90.05 ± 0.31
ACM-GCN	55.14 ± 0.16	47.37 ± 0.59	80.33 ± 3.91
$\mathbf{G}^2$-GraphSAGE	$\mathbf{69.50 \pm 0.39}$	$\mathbf{63.30 \pm 1.84}$	$\mathbf{90.85 \pm 0.64}$

Table 3 shows the results of $\mathbf{G}^2$-GraphSAGE together with other standard GNNs, as well as recent state-of-the-art models, i.e., MLP(Goodfellow et al., 2016), GCN (Kipf & Welling, 2017), GAT (Velickovic et al., 2018), MixHop (Abu-El-Haija et al., 2019), LINK(X) (Lim et al., 2021), GCNII (Chen et al., 2020b), APPNP (Klicpera et al., 2018), GloGNN (Li et al., 2022), GPR-GNN (Chien et al., 2020) and ACM-GCN (Luan et al., 2021). We can see that $\mathbf{G}^2$-GraphSAGE significantly outperforms current state-of-the-art (by up to 13%) on the two heterophilic graphs (i.e., snap-patents and arXiv-year). Moreover, $\mathbf{G}^2$-GraphSAGE is on-par with the current state-of-the-art on the homophilic graph dataset genius.

We conclude that the proposed gradient gating method can successfully be scaled up to large graphs, reaching state-of-the-art performance, in particular on heterophilic graph datasets.

5 Related Work

Gating. Gating is a key component of our proposed framework. The use of gating (i.e., the modulation between 0 and 1) of hidden layer outputs has a long pedigree in neural networks and sequence modeling. In particular, classical recurrent neural network (RNN) architectures such as LSTM (Hochreiter & Schmidhuber, 1997) and GRU (Cho et al., 2014) rely on gates to modulate information propagation in the RNN. Given the connections between RNNs and early versions of GNNs (Zhou et al., 2019), it is not surprising that the idea of gating has been used in designing GNNs Bresson & Laurent (2017); Li et al. (2016); Zhang et al. (2018). However, to the best of our knowledge, the use of local graph-gradients to further modulate gating in order to alleviate the oversmoothing problem is novel, and so is its theoretical analysis.

Multi-scale methods. The multi-rate gating procedure used in $\mathbf{G}^2$ is a particular example of *multi-scale* mechanisms. The use of multi-scale neural network architectures has a long history. An early example is Hinton & Plaut (1987), who proposed a neural network with each connection having a fast changing weight for temporary memory and a slow changing weight for long-term learning. The classical convolutional neural networks (CNNs, LeCun et al. (1989)) can be viewed as multi-scale architectures for processing multiple *spatial* scales in images (Bai et al., 2020). Moreover, there is a close connection between our multi-rate mechanism (4) and the use of multiple time scales in recently proposed sequence models such as UnICORNN (Rusch & Mishra, 2021) and long expressive memory (LEM) (Rusch et al., 2022b).

Neural differential equations. Ordinary and partial differential equations (ODEs and PDEs) are playing an increasingly important role in designing, interpreting, and analyzing novel graph machine learning architectures Avelar et al. (2019); Poli et al. (2019); Zhuang et al. (2020); Xhonneux et al. (2020). Chamberlain et al. (2021a) designed attentional GNNs by discretizing parabolic diffusion-type PDEs. Di Giovanni et al. (2022) interpreted GCNs as gradient flows minimizing a generalized version of the Dirichlet energy. Chamberlain et al. (2021b) applied a non-Euclidean diffusion equation ("Beltrami flow") yielding a scheme with adaptive spatial derivatives ("graph rewiring"), and Topping et al. (2021) studied a discrete geometric PDE similar to Ricci flow to improve information propagation in GNNs. Eliasof et al. (2021) proposed a GNN framework arising from a mixture of parabolic (diffusion) and hyperbolic (wave) PDEs on graphs with convolutional coupling operators, which describe dissipative wave propagation. Finally, Rusch et al. (2022a) used systems of nonlinear oscillators coupled through the associated graph structure to rigorously overcome the oversmoothing problem. In line with these works, one contribution of our paper is a continuous version of $\mathbf{G}^2$ (9), which we used for a rigorous analysis of the oversmoothing problem. Understanding whether this system of ODEs has an interpretation as a known physical model is a topic for future research.

6 Discussion

We have proposed a novel framework, termed $\mathbf{G}^2$, for efficient learning on graphs. $\mathbf{G}^2$ builds on standard MPNNs, but seeks to overcome their limitations. In particular, we focus on the fact that for standard MPNNs such as GCN or GAT, each node (in every hidden layer) is updated at the same *rate*. This might inhibit efficient learning of tasks where different node features would need to be updated at different rates. Hence, we equip a standard MPNN with *gates* that amount to a multi-rate modulation for the hidden layer output in (4). This enables multiple rates (or scales) of flow of information across a graph. Moreover, we leverage local (graph) gradients to further constrain the gates. This is done to alleviate oversmoothing where node features become indistinguishable as the number of layers is increased.

By combining these ingredients, we present a very flexible framework (dubbed $\mathbf{G}^2$) for graph machine learning wherein *any* existing MPNN hidden layer can be employed as the coupling function and the multi-rate gradient gating mechanism can be built on top of it. Moreover, we also show that $\mathbf{G}^2$ corresponds to a time-discretization of a system of ODEs (9). By studying the (in)-stability of the corresponding zero-Dirichlet energy steady states we rigorously prove that gradient gating can mitigate the oversmoothing problem, paving the way for the use of very deep GNNs within the $\mathbf{G}^2$ framework. In contrast, the lack of gradient gating is shown to lead to oversmoothing.

We also present an extensive empirical evaluation to illustrate different aspects of the proposed $\mathbf{G}^2$ framework. Starting with synthetic, small-scale experiments, we demonstrate that i) $\mathbf{G}^2$ can prevent oversmoothing by keeping the Dirichlet energy constant, even for a very large number of hidden layers, ii) this feature allows us to deploy very deep architectures and to observe that the accuracy of classification tasks can *increase* with increasing number of hidden layers, iii) the multi-rate mechanism significantly improves performance on node regression tasks when the node features are distributed over a range of scales, and iv) $\mathbf{G}^2$ is very accurate at classification on *heterophilic* datasets, witnessing an increasing gain in performance with increasing heterophily.

This last feature was more extensively investigated, and we observed that $\mathbf{G}^2$ can significantly outperform baselines as well as recently proposed methods on both benchmark medium-scale and large-scale heterophilic datasets, achieving state-of-the-art performance. Thus, by a combination of theory and experiments, we demonstrate that the $\mathbf{G}^2$-framework is a promising approach for learning on graphs.

Future work. As future work, we would like to better understand the continuous limit of $\mathbf{G}^2$, i.e., the ODEs (9), especially in the zero spatial-resolution limit and investigate if the resulting continuous equations have interesting geometric and analytical properties. Moreover, we would like to use $\mathbf{G}^2$ for solving scientific problems, such as in computational chemistry or the numerical solutions of PDEs. Finally, the promising results for $\mathbf{G}^2$ on large-scale graphs encourage us to use it for even larger industrial-scale applications.

ACKNOWLEDGEMENTS

The research of TKR and SM was performed under a project that has received funding from the European Research Council (ERC) under the European Union's Horizon 2020 research and innovation programme (Grant Agreement No. 770880). MWM would like to acknowledge the IARPA (contract W911NF20C0035), NSF, and ONR for providing partial support of this work. MB is supported in part by ERC Grant No. 724228 (LEMAN). The authors thank Emmanuel de Bézenac (ETH) for his constructive suggestions.

REFERENCES

Sami Abu-El-Haija, Bryan Perozzi, Amol Kapoor, Nazanin Alipourfard, Kristina Lerman, Hrayr Harutyunyan, Greg Ver Steeg, and Aram Galstyan. Mixhop: Higher-order graph convolutional architectures via sparsified neighborhood mixing. In *international conference on machine learning*, pp. 21–29. PMLR, 2019.

Uri Alon and Eran Yahav. On the bottleneck of graph neural networks and its practical implications. In *ICML*, 2021.

P. H. C. Avelar, A. R. Tavares, , M. Gori, and L. C. Lamb. Discrete and continuous deep residual learning over graphs. *arXiv preprint*, 2019.

Shaojie Bai, Vladlen Koltun, and J Zico Kolter. Multiscale deep equilibrium models. In *Advances in Neural Information Processing Systems*, pp. 770–778, 2020.

Deyu Bo, Xiao Wang, Chuan Shi, and Huawei Shen. Beyond low-frequency information in graph convolutional networks. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, pp. 3950–3957, 2021.

Xavier Bresson and Thomas Laurent. Residual gated graph convnets. *arXiv:1711.07553*, 2017.

Michael M Bronstein, Joan Bruna, Taco Cohen, and Petar Veličković. Geometric deep learning: Grids, groups, graphs, geodesics, and gauges. *arXiv:2104.13478*, 2021.

Joan Bruna, Wojciech Zaremba, Arthur Szlam, and Yann LeCun. Spectral networks and locally connected networks on graphs. In *2nd International Conference on Learning Representations, ICLR 2014*, 2014.

Ben Chamberlain, James Rowbottom, Maria I. Gorinova, Michael M. Bronstein, Stefan Webb, and Emanuele Rossi. GRAND: graph neural diffusion. In *Proceedings of the 38th International Conference on Machine Learning*, volume 139, pp. 1407–1418. PMLR, 2021a.

Benjamin Chamberlain, James Rowbottom, Davide Eynard, Francesco Di Giovanni, Xiaowen Dong, and Michael Bronstein. Beltrami flow and neural diffusion on graphs. In *NeurIPS*, 2021b.

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, 2020a.

Ming Chen, Zhewei Wei, Zengfeng Huang, Bolin Ding, and Yaliang Li. Simple and deep graph convolutional networks. In *International Conference on Machine Learning*, pp. 1725–1735. PMLR, 2020b.

Eli Chien, Jianhao Peng, Pan Li, and Olgica Milenkovic. Adaptive universal generalized pagerank graph neural network. *arXiv preprint arXiv:2006.07988*, 2020.

Kyunghyun Cho, B van Merrienboer, Caglar Gulcehre, F Bougares, H Schwenk, and Yoshua Bengio. Learning phrase representations using rnn encoder-decoder for statistical machine translation. In *Conference on Empirical Methods in Natural Language Processing (EMNLP 2014)*, 2014.

Michaël Defferrard, Xavier Bresson, and Pierre Vandergheynst. Convolutional neural networks on graphs with fast localized spectral filtering. *Advances in neural information processing systems*, 29:3844–3852, 2016.

Austin Derrow-Pinion, Jennifer She, David Wong, Oliver Lange, Todd Hester, Luis Perez, Marc Nunkesser, Seongjae Lee, Xueying Guo, Peter W Battaglia, Vishal Gupta, Ang Li, Zhongwen Xu, Alvaro Sanchez-Gonzalez, Yujia Li, and Petar Veličković. Traffic Prediction with Graph Neural Networks in Google Maps. 2021.

Francesco Di Giovanni, James Rowbottom, Benjamin P Chamberlain, Thomas Markovich, and Michael M Bronstein. Graph neural networks as gradient flows. *arXiv:2206.10991*, 2022.

Moshe Eliasof, Eldad Haber, and Eran Treister. Pde-gcn: Novel architectures for graph neural networks motivated by partial differential equations. In *NeurIPS*, 2021.

Lawrence C Evans. *Partial differential equations*, volume 19. American Mathematical Soc., 2010.

Paolo Frasconi, Marco Gori, and Alessandro Sperduti. A general framework for adaptive processing of data structures. *IEEE Trans. Neural Networks*, 9(5):768–786, 1998.

Thomas Gaudelet, Ben Day, Arian R Jamasb, Jyothish Soman, Cristian Regep, Gertrude Liu, Jeremy BR Hayter, Richard Vickers, Charles Roberts, Jian Tang, et al. Utilizing graph machine learning within drug discovery and development. *Briefings in Bioinformatics*, 22(6), 2021.

Justin Gilmer, Samuel S Schoenholz, Patrick F Riley, Oriol Vinyals, and George E Dahl. Neural message passing for quantum chemistry. In *ICML*, 2017.

Christoph Goller and Andreas Kuchler. Learning task-dependent distributed representations by backpropagation through structure. In *ICNN*, 1996.

Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep learning*. MIT press, 2016.

Marco Gori, Gabriele Monfardini, and Franco Scarselli. A new model for learning in graph domains. In *IJCNN*, 2005.

Will Hamilton, Zhitao Ying, and Jure Leskovec. Inductive representation learning on large graphs. *Advances in neural information processing systems*, 30, 2017.

G.E. Hinton and D.C. Plaut. Using fast weights to deblur old memories. In *Proceedings of the ninth annual conference of the cognitive science society*, pp. 177–186, 1987.

Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8): 1735–1780, 1997.

Thomas N. Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. In *ICLR*, 2017.

Johannes Klicpera, Aleksandar Bojchevski, and Stephan Günnemann. Predict then propagate: Graph neural networks meet personalized pagerank. *arXiv preprint arXiv:1810.05997*, 2018.

Yann LeCun, Bernhard Boser, John S Denker, Donnie Henderson, Richard E Howard, Wayne Hubbard, and Lawrence D Jackel. Backpropagation applied to handwritten zip code recognition. *Neural computation*, 1(4):541–551, 1989.

Xiang Li, Renyu Zhu, Yao Cheng, Caihua Shan, Siqiang Luo, Dongsheng Li, and Weining Qian. Finding global homophily in graph neural networks when meeting heterophily. *arXiv preprint arXiv:2205.07308*, 2022.

Yujia Li, Daniel Tarlow, Marc Brockschmidt, and Richard S. Zemel. Gated graph sequence neural networks. In Yoshua Bengio and Yann LeCun (eds.), *ICLR*, 2016.

Derek Lim, Felix Hohne, Xiuyu Li, Sijia Linda Huang, Vaishnavi Gupta, Omkar Bhalerao, and Ser Nam Lim. Large scale learning on non-homophilous graphs: New benchmarks and strong simple methods. *Advances in Neural Information Processing Systems*, 34:20887–20902, 2021.

Lek-Heng Lim. Hodge laplacians on graphs. *arXiv preprint arXiv:1507.05379*, 2015.

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 to do node classification? *arXiv preprint arXiv:2109.05641*, 2021.

Federico Monti, Davide Boscaini, Jonathan Masci, Emanuele Rodola, Jan Svoboda, and Michael M Bronstein. Geometric deep learning on graphs and manifolds using mixture model cnns. In *CVPR*, 2017.

H Nt and T Maehara. Revisiting graph neural networks: all we have is low pass filters. *arXiv:1812.08434v4*, 2019.

K Oono and T Suzuki. Graph neural networks exponentially lose expressive power for node classification. In *ICLR*, 2020.

Hongbin Pei, Bingzhe Wei, Kevin Chen-Chuan Chang, Yu Lei, and Bo Yang. Geom-gcn: Geometric graph convolutional networks. *arXiv preprint arXiv:2002.05287*, 2020.

Michael Poli, Stefano Massaroli, Junyoung Park, Atsushi Yamashita, Hajime Asama, and Jinkyoo Park. Graph neural ordinary differential equations. pp. 6571–6583, 2019.

Yu Rong, Wenbing Huang, Tingyang Xu, and Junzhou Huang. Towards deep graph convolutional networks on node classification. In *ICLR*, 2020.

Benedek Rozemberczki, Carl Allen, and Rik Sarkar. Multi-scale attributed node embedding. *Journal of Complex Networks*, 9(2):cnab014, 2021.

T. Konstantin Rusch and Siddhartha Mishra. Unicornn: A recurrent model for learning very long time dependencies. In *Proceedings of the 38th International Conference on Machine Learning*, volume 139 of *Proceedings of Machine Learning Research*, pp. 9168–9178. PMLR, 2021.

T. Konstantin Rusch, Ben Chamberlain, James Rowbottom, Siddhartha Mishra, and Michael Bronstein. Graph-coupled oscillator networks. In *Proceedings of the 39th International Conference on Machine Learning*, volume 162 of *Proceedings of Machine Learning Research*, pp. 18888–18909. PMLR, 2022a.

T. Konstantin Rusch, Siddhartha Mishra, N. Benjamin Erichson, and Michael. W Mahoney. Long expressive memory for sequence modeling. In *International Conference on Learning Representations*, 2022b.

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

Jonathan Shlomi, Peter Battaglia, and Jean-Roch Vlimant. Graph neural networks in particle physics. *Machine Learning: Science and Technology*, 2(2):021001, 2020.

Alessandro Sperduti. Encoding labeled graphs by labeling RAAM. In *NIPS*, 1994.

Alessandro Sperduti and Antonina Starita. Supervised neural networks for the classification of structures. *IEEE Trans. Neural Networks*, 8(3):714–735, 1997.

Jake Topping, Francesco Di Giovanni, Benjamin Paul Chamberlain, Xiaowen Dong, and Michael M Bronstein. Understanding over-squashing and bottlenecks on graphs via curvature. *arXiv:2111.14522*, 2021.

Petar Velickovic, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, and Yoshua Bengio. Graph attention networks. In *6th International Conference on Learning Representations, ICLR*, 2018.

Lanning Wei, Huan Zhao, and Zhiqiang He. Designing the topology of graph neural networks: A novel feature fusion perspective. In *Proceedings of the ACM Web Conference 2022*, pp. 1381–1391, 2022.

S. Wiggins. *Introduction to nonlinear dynamical systems and chaos*. Springer, 2003.

Louis-pascal A C Xhonneux, Meng Qu, and Jian Tang. Continuous graph neural networks. In *ICML*, 2020.

Keyulu Xu, Weihua Hu, Jure Leskovec, and Stefanie Jegelka. How powerful are graph neural networks? *arXiv preprint arXiv:1810.00826*, 2018.

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. *arXiv preprint arXiv:2102.06462*, 2021.

Rex Ying, Ruining He, Kaifeng Chen, Pong Eksombatchai, William L Hamilton, and Jure Leskovec. Graph convolutional neural networks for web-scale recommender systems. In *KDD*, 2018.

Jiani Zhang, Xingjian Shi, Junyuan Xie, Hao Ma, Irwin King, and Dit-Yan Yeung. Gaan: Gated attention networks for learning on large and spatiotemporal graphs. *arXiv preprint arXiv:1803.07294*, 2018.

Lingxiao Zhao and Leman Akoglu. Pairnorm: Tackling oversmoothing in gnns. *arXiv preprint arXiv:1909.12223*, 2019.

Jie Zhou, Ganqu Cui, Zhengyan Zhang, Cheng Yang, Zhiyuan Liu, Lifeng Wang, , Changcheng Li, and Maosong Sun. Graph neural networks: a review of methods and applications. *arXiv:1812.08434v4*, 2019.

Jiong Zhu, Yujun Yan, Lingxiao Zhao, Mark Heimann, Leman Akoglu, and Danai Koutra. Beyond homophily in graph neural networks: Current limitations and effective designs. *Advances in Neural Information Processing Systems*, 33:7793–7804, 2020.

J. Zhuang, N. Dvornek, X. Li, and J. S. Duncan. Ordinary differential equations on graph networks. *Technical Report*, 2020.

Supplementary Material for:
Gradient Gating for Deep Multi-Rate Learning on Graphs

A Additional Experiments

In this section, we describe additional empirical results to complement those in the main text.

On the multi-rate effect of $\mathbf{G}^2$. Here, we analyze the performance of $\mathbf{G}^2$ on the multi-scale node-level regression task of the main text. As we see in the main text, $\mathbf{G}^2$ applied to GCN or GAT outperforms their plain counterparts (GCN and GAT) on the multi-scale node-level regression task by more than 50% on Chameleon and more than 100% on Squirrel. The question therefore arises whether this better performance can be explained by the multi-rate nature of gradient gating.

To empirically analyse this, we begin by adding a control parameter α to $\mathbf{G}^2$ (4) as follows,

$$\mathbf{X}^n = (1 - (\boldsymbol{\tau}^n)^\alpha) \odot \mathbf{X}^{n-1} + (\boldsymbol{\tau}^n)^\alpha \odot \sigma(\mathbf{F}_\theta(\mathbf{X}^{n-1}, \mathcal{G})).$$

Clearly, setting $\alpha = 1$ recovers the original gradient gating message-passing update,

$$\mathbf{X}^n = (1 - \boldsymbol{\tau}^n) \odot \mathbf{X}^{n-1} + \boldsymbol{\tau}^n \odot \sigma(\mathbf{F}_\theta(\mathbf{X}^{n-1}, \mathcal{G})),$$

while setting $\alpha = 0$ disables any explicit multi-rate behavior and a plain message-passing scheme is recovered,

$$\mathbf{X}^n = \sigma(\mathbf{F}_\theta(\mathbf{X}^{n-1}, \mathcal{G})).$$

Note that by continuously changing α from 0 to 1 controls the level of multi-rate behavior in the proposed gradient gating method.

In Fig. 6 we plot the test NMSE of the best performing $\mathbf{G}^2$-GCN and $\mathbf{G}^2$-GAT on the Chameleon multi-scale node-level regression task for increasing values of $\alpha \in [10^{-3}, 1]$ in log-scale. We can see that the test NMSE monotonically decreases (lower error means better performance) for both $\mathbf{G}^2$-GCN and $\mathbf{G}^2$-GAT for increasing values of α, i.e., increasing level of multi-rate behavior. We can conclude that the multi-rate behavior of $\mathbf{G}^2$ is instrumental in successfully learning multi-scale regression tasks.

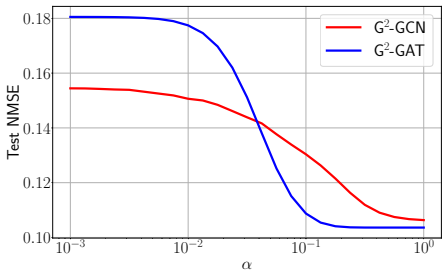

Figure 6: Test NMSE on the multi-scale chameleon node-level regression task of $\mathbf{G}^2$-GCN and $\mathbf{G}^2$-GAT for continuously decreasing level of multi-rate behavior.

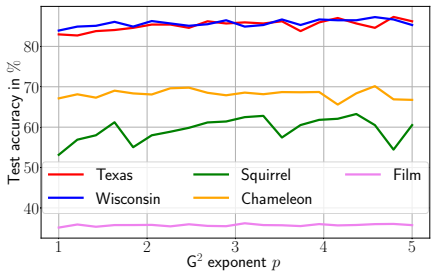

Figure 7: Test accuracies of $\mathbf{G}^2$-GraphSAGE on Texas, Squirrel, Film, Wisconsin and Chameleon graph datasets for varying values of p in (4).

On the sensitivity of performance of $\mathbf{G}^2$ to the hyperparameter p. The proposed gradient gating model implicitly depends on the hyperparameter p, which defines the multiple rates $\boldsymbol{\tau}$, i.e.,

$$\hat{\boldsymbol{\tau}}^n = \hat{\sigma}(\hat{\mathbf{F}}_\theta(\mathbf{X}^{n-1}, \mathcal{G})),$$

$$\boldsymbol{\tau}_{ik}^n = \tanh\left(\sum_{j \in \mathcal{N}_i} |\hat{\tau}_{ik}^n - \hat{\tau}_{jk}^n|^p\right).$$

While any value $p > 0$ can be used in practice, a standard hyperparameter tuning procedure (see B for the training details) on p has been applied in every experiment included in this paper. Thus,

it is natural to ask how sensitive the performance of $\mathbf{G}^2$ is with respect to different values of the hyperparameter p.

To answer this question, we trained different $\mathbf{G}^2$-GraphSAGE models on a variety of different graph datasets (i.e., Texas, Squirrel, Film, Wisconsin and Chameleon) for different values of $p \in [1, 5]$. Fig. 7 shows the resulting performance of $\mathbf{G}^2$-GraphSAGE. We can see that different values of p do not significantly change the performance of the model. However, including the hyperparameter p to the hyperparameter fine-tuning procedure will further improve the overall performance of $\mathbf{G}^2$.

On the sensitivity of performance of $\mathbf{G}^2$ to the number of parameters. All results of $\mathbf{G}^2$ provided in the main paper are obtained using standard hyperparameter tuning (i.e., random search). Those hyperparameters include the number of hidden channels for each hidden node of the graph, which directly correlates with the total number of parameters used in $\mathbf{G}^2$. It is thus natural to ask how $\mathbf{G}^2$ performs compared to its plain counter-version (e.g. $\mathbf{G}^2$-GCN to GCN) for the exact same number of total parameters of the underlying model. To this end, Fig. 8 shows the test accuracies of $\mathbf{G}^2$-GCN and GCN for increasing number of total parameters in its corresponding model. We can see that first, using more parameters has only a slight effect on the overall performance of both models. Second, and most importantly, $\mathbf{G}^2$-GCN constantly reaches significantly higher test accuracies for the exact same number of total parameters. We can thus rule out that the better performance of $\mathbf{G}^2$ compared to its plain counter-versions is explained by the usage of more parameters.

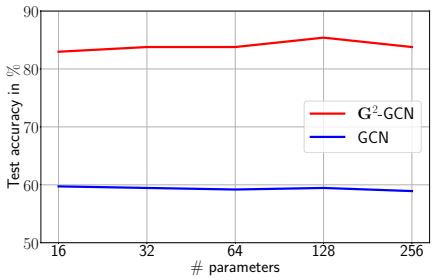

Figure 8: Test accuracies of plain GCN and $\mathbf{G}^2$-GCN on Texas for varying number of total parameters in the GNN.

Ablation of $\hat{\mathbf{F}}_\theta$ in $\mathbf{G}^2$. In its most general form $\mathbf{G}^2$ (4) uses an additional GNN $\hat{\mathbf{F}}_\theta$ to construct the multiple rates $\boldsymbol{\tau}^n$. Is this additional GNN needed ? To answer this question, Table 4 shows in which of the provided experiments (using $\mathbf{G}^2$-GraphSAGE) we actually used an additional GNN $\hat{\mathbf{F}}_\theta$ (as suggested by our hyperparameter tuning protocol). We can see that on small-scale experiments having an additional GNN is not needed. However, on the considered medium and large-scale graph datasets it is beneficial to use it. Motivated by this, Table 5 shows the results for $\mathbf{G}^2$-GraphSAGE on the three medium-scale graph datasets (Film, Squirrel and Chameleon) without using an additional GNN in (4) (i.e., $\mathbf{F}_\theta = \hat{\mathbf{F}}_\theta$) as well as with using an additional GNN (i.e., the results provided in the main paper). We can see that while $\mathbf{G}^2$-GraphSAGE without an additional GNN (i.e., w/o $\hat{\mathbf{F}}_\theta$) yields competitive results, using an additional GNN is needed in order to obtain state-of-the-art results on these three datasets.

Table 4: Usage of $\hat{\mathbf{F}}_\theta$ in $\mathbf{G}^2$ (4) for each result with $\mathbf{G}^2$-GraphSAGE presented in the main paper (YES indicates the usage of $\hat{\mathbf{F}}_\theta$, while NO indicates that no additional GNN is used to construct the multiple rates, i.e., $\mathbf{F}_\theta = \hat{\mathbf{F}}_\theta$)

Texas	Wisconsin	Film	Squirrel	Chameleon	Cornell	snap-patents	arXiv-year	genius
NO	NO	YES	YES	YES	NO	YES	YES	YES

Table 5: Test accuracies of $\mathbf{G}^2$-GraphSAGE with and without additional GNN (i.e., w/ $\hat{\mathbf{F}}_\theta$ and w/o $\hat{\mathbf{F}}_\theta$ in (4)) on Film, Squirrel and Chameleon graph dataset.

	Film	**Squirrel**	**Chameleon**
$\mathbf{G}^2$-GraphSAGE w/ $\hat{\mathbf{F}}_\theta$	37.14 ± 1.01	64.26 ± 2.38	71.40 ± 2.38
$\mathbf{G}^2$-GraphSAGE w/o $\hat{\mathbf{F}}_\theta$	36.83 ± 1.26	55.78 ± 1.61	65.04 ± 2.27

Ablation of multi-rate channels in $\mathbf{G}^2$. The corner stone of the proposed $\mathbf{G}^2$ is the multi-rate matrix $\boldsymbol{\tau}^n$ in (4), which automatically solves the oversmoothing issue for any given GNN (Proposition 3.3). This multi-rate matrix learns different rates for every node but also for every channel of every node. It is thus natural to ask if the multi-rate property for the channels is necessary, or if having multiple rates for the different nodes is sufficient, i.e., having a **multi-rate vector** $\boldsymbol{\tau}^n \in \mathbb{R}^v$. A direct construction of such multi-rate vector (derived from our proposed $\mathbf{G}^2$) is:

$$\begin{aligned}
\hat{\boldsymbol{\tau}}^n &= \sigma(\hat{\mathbf{F}}_\theta(\mathbf{X}^{n-1}, \mathcal{G})), \\
\boldsymbol{\tau}_i^n &= \tanh\left(\sum_{j \in \mathcal{N}_i} \|\hat{\boldsymbol{\tau}}_j^n - \hat{\boldsymbol{\tau}}_i^n\|_p\right), \\
\mathbf{X}^n &= (1 - \boldsymbol{\tau}^n) \odot \mathbf{X}^{n-1} + \boldsymbol{\tau}^n \odot \sigma(\mathbf{F}_\theta(\mathbf{X}^{n-1}, \mathcal{G})).
\end{aligned} \tag{16}$$

Note that the only difference to our proposed $\mathbf{G}^2$ is in the second equation of (16), where we sum over the node-wise p-norms of the differences of adjacent nodes. This way, we compute a single scalar $\boldsymbol{\tau}_i^n$ for every node $i \in \mathcal{V}$.

Table 6 shows the results of our proposed $\mathbf{G}^2$-GraphSAGE as well as the single-rate channels ablation of $\mathbf{G}^2$ (eq. (16)) on the Film, Squirrel and Chameleon graph datasets. As a baseline, we also include the results of a plain GraphSAGE. We can see that while $\mathbf{G}^2$ with single-scale channels outperforms the base GraphSAGE model, our proposed $\mathbf{G}^2$ with multi-rate channels vastly outperforms the single-rate channels version of $\mathbf{G}^2$.

Table 6: Test accuracies of plain GraphSAGE, $\mathbf{G}^2$-GraphSAGE with multi-rate channels for each node (i.e., standard $\mathbf{G}^2$ (4)) as well as with only a single rate for every channel on Film, Squirrel and Chameleon.

	Film	**Squirrel**	**Chameleon**
GraphSAGE	34.23 ± 0.99	41.61 ± 0.74	58.73 ± 1.68
$\mathbf{G}^2$-GraphSAGE w/ multi-rate channels $\mathbf{G}^2$ (4)	37.14 ± 1.01	64.26 ± 2.38	71.40 ± 2.38
$\mathbf{G}^2$-GraphSAGE w/ single-rate channels $\mathbf{G}^2$ (16)	36.67 ± 0.56	44.03 ± 1.01	60.29 ± 3.45

Alternative measures of oversmoothing. The proof of Proposition 3.2 and Proposition 3.3 as well as the first experiment in the main paper is based on the definition of oversmoothing using the Dirichlet energy, which was proposed in Rusch et al. (2022a). However, there exist alternative measures to describe the oversmoothing phenomenon in deep GNNs. One such measure is the mean average distance (MAD), which was proposed in Chen et al. (2020a). In order to check if our proposed $\mathbf{G}^2$ mitigates oversmoothing defined through the MAD measure we repeat the first experiment in the main paper and plot the MAD instead of the Dirichlet energy for increasing number of layers in Fig. 9. We can see that while the MAD of a plain GCN and GAT converges exponentially with increasing number of layers, it remains constant for $\mathbf{G}^2$-GCN and $\mathbf{G}^2$-GAT. We can thus conclude that $\mathbf{G}^2$ mitigates oversmoothing defined through the MAD measure.

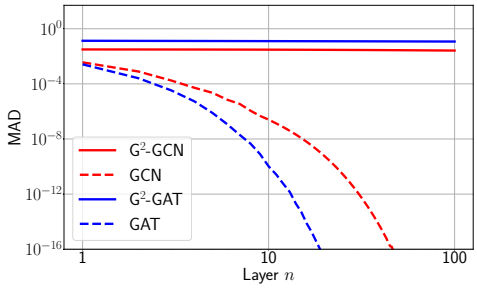

Figure 9: Mean average distance (MAD) of layer-wise node features $\mathbf{X}^n$ propagated through a GAT, GCN and their gradient gated versions ($\mathbf{G}^2$-GAT, $\mathbf{G}^2$-GCN).

B TRAINING DETAILS

All small and medium-scale experiments have been run on NVIDIA GeForce RTX 2080 Ti, GeForce RTX 3090, TITAN RTX and Quadro RTX 6000 GPUs. The large-scale experiments have been run on Nvidia Tesla A100 (40GiB) GPUs.

All hyperparameters were tuned using random search. Table 7 shows the ranges of each hyperparameter as well as the random distribution used to randomly sample from it. Moreover, Table 8 shows the rounded hyperparameter p in $\mathbf{G}^2$ (4) of each best performing network.

Table 7: Hyperparameter ranges.

	range	**rand. distribution**
learning rate	$[10^{-4}, 10^{-2}]$	log uniform
hidden size m	$\{32, 64, 128, 256, 512\}$	disc. uniform
dropout input	$[0, 0.9]$	uniform
dropout output	$[0, 0.9]$	uniform
weight decay	$[10^{-8}, 10^{-2}]$	log uniform
$\mathbf{G}^2$-exponent p	$[1, 5]$	uniform
Usage of $\hat{\mathbf{F}}_\theta$ in (4)	{YES, NO}	disc. uniform

Table 8: Rounded hyperparameter p in $\mathbf{G}^2$ of each best performing network.

	Texas	**Wisconsin**	**Film**	**Squirrel**	**Chameleon**	**Cornell**	**snap-patents**	**arXiv-year**	**genius**
$\mathbf{G}^2$-GAT	3.06	1.68	1.23	3.48	3.54	3.54	/	/	/
$\mathbf{G}^2$-GCN	3.93	2.92	3.79	1.99	1.08	3.87	/	/	/
$\mathbf{G}^2$-GraphSAGE	4.47	1.14	2.89	3.04	2.00	3.27	1.60	3.40	4.40

C MATHEMATICAL DETAILS

In this section, we provide proofs for Propositions 3.3 and 3.4 in the main text. We start with the following technical result which is necessary in the subsequent proofs.

A Poincare Inequality on Connected Graphs. Poincare inequalities for functions (Evans, 2010) bound function values in terms of their gradients. Similar bounds on node values in terms of graph-gradients can be derived and a particular instance is given below,

Proposition C.1. *Let $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ be a connected graph and the corresponding (scalar) node features are denoted by $\mathbf{y}_i \in \mathbb{R}$, for all $i \in \mathcal{V}$. Let $\mathbf{y}_1 = 0$. Then, the following bound holds,*

$$\sum_{i\in\mathcal{V}} \mathbf{y}_i^2 \leq \overline{d}\Delta_1 \sum_{i\in\mathcal{V}} \sum_{j\in\mathcal{N}_i} |\mathbf{y}_j - \mathbf{y}_i|^2, \tag{17}$$

where $\overline{d} = \max_{i\in\mathcal{V}} \deg(i)$ and Δ_1 is the eccentricity of the node 1.

Proof. Fix a node $i \in \mathcal{V}$. By assumption, the graph $\mathcal{G}$ is connected. Hence, there exists a path connecting i and the node 1. Denote the shortest path as $\mathcal{P}(i, 1)$. This path can be expressed in terms of the nodes $\ell_{i,1}$ with $0 \leq \ell \leq \delta$, where $0_{i,1} = 1$ and $\delta_{i,1} = i$. For any ℓ, we require $\ell_{i,1} \sim (\ell + 1)_{i,1}$. Moreover, $\delta_{i,1}$ is the graph distance between the nodes i and 1 and $\Delta_1 = \max_{i\in\mathcal{V}} \delta_{i,1}$ is the eccentricity of the node 1.

Given the node feature $\mathbf{y}_i$, we can rewrite it as,

$$\mathbf{y}_i = \mathbf{y}_1 + \sum_{\ell=0}^{\delta-1} \mathbf{y}_{(\ell+1)_{i,1}} - \mathbf{y}_{\ell_{i,1}} = \sum_{\ell=0}^{\delta-1} \mathbf{y}_{(\ell+1)_{i,1}} - \mathbf{y}_{\ell_{i,1}},$$

as by assumption $\mathbf{y}_1 = 0$.

Using Cauchy-Schwartz inequality on the previous identity yields,

$$\mathbf{y}_i^2 \leq \Delta_1 \sum_{\ell=0}^{\delta-1} \left(\mathbf{y}_{(\ell+1)_{i,1}} - \mathbf{y}_{\ell_{i,1}}\right)^2.$$

Summing the above inequality over $i \in \mathcal{V}$ and using the fact that $\ell_{i,1} \sim (\ell + 1)_{i,1}$, we obtain the desired Poincare inequality (17).

□

C.1 Proof of Proposition 3.3 of Main Text

Proof. By the definition of exponential stability, we consider a small perturbation around the steady state $\mathbf{c}$ and study whether this perturbation grows or decays in time. To this end, define the perturbation as,

$$\hat{\mathbf{X}}_i = \mathbf{X}_i - c, \quad 1 \leq i \leq v. \tag{18}$$

A tedious but straightforward calculation shows that these perturbations evolve by the following *linearized* system of ODEs,

$$\frac{d\hat{\mathbf{X}}_i(t)}{dt} = \sum_{j\in\mathcal{N}_i} \mathbf{A}_{ij}(c, c) \left(\hat{\mathbf{X}}_j - \hat{\mathbf{X}}_i\right), \quad \forall t,\ \forall i \in \mathcal{V}. \tag{19}$$

Multiplying $\hat{\mathbf{x}}_i$ to both sides of (19) yields,

$$\begin{aligned} \hat{\mathbf{X}}_i \frac{d\hat{\mathbf{X}}_i(t)}{dt} &= \sum_{j\in\mathcal{N}_i} \mathbf{A}_{ij}(c, c)\hat{\mathbf{X}}_i \left(\hat{\mathbf{X}}_j - \hat{\mathbf{X}}_i\right), \\ \Rightarrow \quad \frac{d\hat{\mathbf{X}}_i^2(t)}{dt} &= \sum_{j\in\mathcal{N}_i} \mathbf{A}_{ij}(c, c) \left(\hat{\mathbf{X}}_j^2 - \hat{\mathbf{X}}_i^2\right) - \sum_{j\in\mathcal{N}_i} \mathbf{A}_{ij}(c, c) \left(\hat{\mathbf{X}}_j - \hat{\mathbf{X}}_i\right)^2. \end{aligned}$$

Summing the above identity over all nodes $i \in \mathcal{V}$ yields,

$$\begin{aligned}
\frac{d}{dt}\sum_{i\in\mathcal{V}}\hat{\mathbf{X}}_i^2(t) &= \sum_{i\in\mathcal{V}}\sum_{j\in\mathcal{N}_i}\mathbf{A}_{ij}(c,c)\left(\hat{\mathbf{X}}_j^2-\hat{\mathbf{X}}_i^2\right) - \sum_{i\in\mathcal{V}}\sum_{j\in\mathcal{N}_i}\mathbf{A}_{ij}(c,c)\left(\hat{\mathbf{X}}_j-\hat{\mathbf{X}}_i\right)^2 \\
&= \frac{1}{2}\sum_{i\in\mathcal{V}}\sum_{j\in\mathcal{N}_i}\underbrace{(\mathbf{A}_{ij}(c,c)-\mathbf{A}_{j,i}(c,c))}_{=0\ (15)}\left(\hat{\mathbf{X}}_j^2-\hat{\mathbf{X}}_i^2\right) - \frac{1}{2}\sum_{i\in\mathcal{V}}\sum_{j\in\mathcal{N}_i}\underbrace{(\mathbf{A}_{ij}(c,c)+\mathbf{A}_{j,i}(c,c))}_{=2\mathbf{A}_{ij}\ (15)}\left(\hat{\mathbf{X}}_j-\hat{\mathbf{X}}_i\right)^2, \\
&= -\sum_{i\in\mathcal{V}}\sum_{j\in\mathcal{N}_i}\mathbf{A}_{ij}(c,c)\left(\hat{\mathbf{X}}_j-\hat{\mathbf{X}}_i\right)^2, \\
&\leq -\underline{a}\sum_{i\in\mathcal{V}}\sum_{j\in\mathcal{N}_i}\left(\hat{\mathbf{X}}_j-\hat{\mathbf{X}}_i\right)^2, \quad \text{(by (15))}, \\
&\leq -\frac{\underline{a}}{\overline{d}\Delta_1}\sum_{i\in\mathcal{V}}\hat{\mathbf{X}}_i^2.
\end{aligned}$$

Here, the last inequality comes from applying the Poincare inequality (17) for the perturbations $\hat{\mathbf{X}}$ and from the fact that by assumption $\hat{\mathbf{X}}_1 = 0$.

Applying Grönwall's inequality yields,

$$\sum_{i\in\mathcal{V}}\hat{\mathbf{X}}_i^2(t) \leq \sum_{i\in\mathcal{V}}\hat{\mathbf{X}}_i^2(0)e^{-\frac{\underline{a}}{\overline{d}\Delta_1}t}. \tag{20}$$

Thus, the initial perturbations around the steady state $\mathbf{c}$ are damped down exponentially fast and the steady state $\mathbf{c}$ is exponentially stable implying that this architecture will lead to oversmoothing. □

C.2 Proof of Proposition 3.4 of Main Text

Proof. As in the proof of Proposition 3.3, we consider small perturbations of form (18) of the steady state $\mathbf{c}$ and investigate how these perturbations evolve in time. Assuming that the initial perturbations are small, i.e., that there exists an $0 < \epsilon << 1$ such that $\max_{i\in\mathcal{V}}|\hat{\mathbf{x}}_i(0)| \leq \epsilon$, we perform a straightforward calculation to obtain that the perturbations (for a short time) evolve with the following *quasi-linearized* system of ODEs,

$$\begin{aligned}
\frac{d\hat{\mathbf{X}}_i(t)}{dt} &= \overline{\boldsymbol{\tau}}_i(t)\sum_{j\in\mathcal{N}_i}\mathbf{A}_{ij}(c,c)\left(\hat{\mathbf{X}}_j-\hat{\mathbf{X}}_i\right),\ \forall i\in\mathcal{V}, \\
\overline{\boldsymbol{\tau}}_i(t) &= \sum_{j\in\mathcal{N}_i}|\hat{\mathbf{X}}_j(t)-\hat{\mathbf{X}}_i(t)|^p,\ \forall i\in\mathcal{V}.
\end{aligned} \tag{21}$$

Note that we have used the fact that $\sigma'(x) = 1$ and $\tanh'(0) = 1$ in obtaining (21) from (14).

Next, we multiply $\hat{\mathbf{x}}_i$ to both sides of (21) to obtain,

$$\begin{aligned}
\hat{\mathbf{X}}_i\frac{d\hat{\mathbf{X}}_i(t)}{dt} &= \sum_{j\in\mathcal{N}_i}\mathbf{A}_{ij}(c,c)\overline{\boldsymbol{\tau}}_i\hat{\mathbf{X}}_i\left(\hat{\mathbf{X}}_j-\hat{\mathbf{X}}_i\right), \\
\Rightarrow \quad \frac{d\hat{\mathbf{X}}_i^2(t)}{dt} &= \sum_{j\in\mathcal{N}_i}\mathbf{A}_{ij}(c,c)\overline{\boldsymbol{\tau}}_i\left(\hat{\mathbf{X}}_j^2-\hat{\mathbf{X}}_i^2\right) - \sum_{j\in\mathcal{N}_i}\mathbf{A}_{ij}(c,c)\overline{\boldsymbol{\tau}}_i\left(\hat{\mathbf{X}}_j-\hat{\mathbf{X}}_i\right)^2
\end{aligned} \tag{22}$$

Trivially,

$$|\hat{\mathbf{X}}_j-\hat{\mathbf{X}}_i|^p \leq \overline{\boldsymbol{\tau}}_i,\ \forall j\in\mathcal{N}_i, \quad \forall i.$$

Applying this inequality to the last line of the identity (22), we obtain,

$$\frac{d\hat{\mathbf{X}}_i^2(t)}{dt} \leq \sum_{j\in\mathcal{N}_i}\mathbf{A}_{ij}(c,c)\overline{\boldsymbol{\tau}}_i\left(\hat{\mathbf{X}}_j^2-\hat{\mathbf{X}}_i^2\right) - \sum_{j\in\mathcal{N}_i}\mathbf{A}_{ij}(c,c)\left|\hat{\mathbf{X}}_j-\hat{\mathbf{X}}_i\right|^{p+2}.$$

Summing the above inequality over $i \in \mathcal{V}$ leads to,

$$\begin{aligned}\frac{d}{dt}\sum_{i\in\mathcal{V}}\hat{\mathbf{X}}_i^2(t) &\leq \sum_{i\in\mathcal{V}}\sum_{j\in\mathcal{N}_i}\mathbf{A}_{ij}(c,c)\overline{\boldsymbol{\tau}}_i\left(\hat{\mathbf{X}}_j^2-\hat{\mathbf{X}}_i^2\right)-\sum_{i\in\mathcal{V}}\sum_{j\in\mathcal{N}_i}\mathbf{A}_{ij}(c,c)\left|\hat{\mathbf{X}}_j-\hat{\mathbf{X}}_i\right|^{p+2}\\ &\leq \frac{1}{2}\sum_{i\in\mathcal{V}}\sum_{j\in\mathcal{N}_i}\mathbf{A}_{ij}(c,c)\left(\overline{\boldsymbol{\tau}}_i-\overline{\boldsymbol{\tau}}_j\right)\left(\hat{\mathbf{X}}_j^2-\hat{\mathbf{X}}_i^2\right)\quad(\mathbf{A}_{ij}=\mathbf{A}_{j,i})\\ &-\underline{a}\sum_{i\in\mathcal{V}}\sum_{j\in\mathcal{N}_i}\left|\hat{\mathbf{X}}_j-\hat{\mathbf{X}}_i\right|^{p+2}\quad(\text{from (15)}).\end{aligned}$$

Therefore, we have the following inequality,

$$\begin{aligned}&\frac{d}{dt}\sum_{i\in\mathcal{V}}\hat{\mathbf{X}}_i^2(t)\leq T_1-T_2,\\ &T_1=\frac{1}{2}\sum_{i\in\mathcal{V}}\sum_{j\in\mathcal{N}_i}\mathbf{A}_{ij}(c,c)\left(\overline{\boldsymbol{\tau}}_i-\overline{\boldsymbol{\tau}}_j\right)\left(\hat{\mathbf{X}}_j^2-\hat{\mathbf{X}}_i^2\right)\\ &T_2=\underline{a}\sum_{i\in\mathcal{V}}\sum_{j\in\mathcal{N}_i}\left|\hat{\mathbf{X}}_j-\hat{\mathbf{X}}_i\right|^{p+2}.\end{aligned}\tag{23}$$

We analyze the differential inequality (23) by starting with the term T_1 in (23). We observe that this term does not have a definite sign and can be either positive or negative. However, we can upper bound this term in the following manner. Given that the right-hand side of the ODE system (21) is Lipschitz continuous, the well-known Cauchy-Lipschitz theorem states that the solutions $\hat{\mathbf{x}}$ depend continuously on the initial data. Given that $\max_{i\in\mathcal{V}}|\hat{\mathbf{X}}_i(0)|\leq\epsilon<<1$ and the bounds on the hidden states (1), there exists a time $\overline{t}>0$ such that

$$\max_{i\in\mathcal{V}}|\hat{\mathbf{X}}_i(t)|\leq 1,\forall t\in[0,\overline{t}].$$

Using the definitions of $\overline{\boldsymbol{\tau}}$ and the right stochasticity of the matrix $\mathbf{A}$, we easily obtain the following bound,

$$|T_1|\leq 2^{p+1}\overline{d}^2 v,\tag{24}$$

where $\overline{d}=\max_{i\in\mathcal{V}}\deg(i)$.

On the other hand, the term T_2 in (23) is clearly positive. Hence, the solutions of resulting ODE,

$$\frac{d}{dt}\sum_{i\in\mathcal{V}}\hat{\mathbf{X}}_i^2(t)\leq -T_2,\tag{25}$$

will clearly decay in time. The key question is whether or not the decay is *exponentially fast*. We answer this question below.

To this end, we have the following calculation using the Hölder's inequality,

$$\begin{aligned}&\sum_{i\in\mathcal{V}}\sum_{j\in\mathcal{N}_i}\left|\hat{\mathbf{X}}_j-\hat{\mathbf{X}}_i\right|^2\leq\left(\overline{d}v\right)^{\frac{p}{p+2}}\left(\sum_{i\in\mathcal{V}}\sum_{j\in\mathcal{N}_i}\left|\hat{\mathbf{X}}_j-\hat{\mathbf{X}}_i\right|^{p+2}\right)^{\frac{2}{p+2}},\\ \Rightarrow\quad&\frac{1}{\left(\overline{d}v\right)^{\frac{p}{2}}}\left(\sum_{i\in\mathcal{V}}\sum_{j\in\mathcal{N}_i}\left|\hat{\mathbf{X}}_j-\hat{\mathbf{X}}_i\right|^2\right)^{\frac{p+2}{2}}\leq\sum_{i\in\mathcal{V}}\sum_{j\in\mathcal{N}_i}\left|\hat{\mathbf{X}}_j-\hat{\mathbf{X}}_i\right|^{p+2}.\end{aligned}$$

Observing that $\hat{\mathbf{X}}_1=0$ by assumption, we can applying the Poincare inequality (17) in the above inequality to further obtain,

$$\frac{1}{\overline{d}^{p+1}v^{\frac{p}{2}}\Delta_1^{\frac{p+2}{2}}}\left(\sum_{i\in\mathcal{V}}|\hat{\mathbf{X}}_i|^2\right)^{\frac{p+2}{2}}\leq\sum_{i\in\mathcal{V}}\sum_{j\in\mathcal{N}_i}\left|\hat{\mathbf{X}}_j-\hat{\mathbf{X}}_i\right|^{p+2}.$$

Hence, from the definition of T_2 (23), we have,

$$T_2 \geq \frac{\underline{a}}{\overline{d}^{p+1} v^{\frac{p}{2}} \Delta_1^{\frac{p+2}{2}}} \left(\sum_{i \in \mathcal{V}} |\hat{\mathbf{X}}_i|^2 \right)^{\frac{p+2}{2}}. \tag{26}$$

Therefore, the differential inequality (25) now reduces to,

$$\frac{d}{dt} \sum_{i \in \mathcal{V}} \hat{\mathbf{X}}_i^2(t) \leq -\frac{\underline{a}}{\overline{d}^{p+1} v^{\frac{p}{2}} \Delta_1^{\frac{p+2}{2}}} \left(\sum_{i \in \mathcal{V}} |\hat{\mathbf{X}}_i|^2 \right)^{\frac{p+2}{2}}. \tag{27}$$

The differential inequality (27) can be explicitly solved to obtain,

$$\sum_{i \in \mathcal{V}} \hat{\mathbf{X}}_i^2(t) \leq \left(2 + pt \frac{\underline{a}}{\overline{d}^{p+1} v^{\frac{p}{2}} \Delta_1^{\frac{p+2}{2}}} \sum_{i \in \mathcal{V}} \hat{\mathbf{X}}_i^2(0)^{\frac{p}{2}} \right)^{-\frac{2}{p}} \sum_{i \in \mathcal{V}} \hat{\mathbf{X}}_i^2(0). \tag{28}$$

From (28), we see that the initial perturbations decay but only *algebraically* at a rate of $t^{-\frac{2}{p}}$ in time. For instance, the decay is only linear in time for $p = 2$ and even slower for higher value of p.

Combining the analysis of the terms $T_{1,2}$ in the differential inequality (23), we see that the one of the terms can lead to a growth in the initial perturbations whereas the second term only leads to polynomial decay. Even if the contribution of the term $T_1 \equiv 0$, the decay of initial perturbations is only polynomial. Thus, the steady state $\mathbf{c}$ is not exponentially stable.

□

Remark C.2. *We note that the Proposition 3.4 assumes a certain structure of the matrix* $\mathbf{A}$ *in* (14)*. A careful perusal of the proof presented above reveal that this assumptions can be further relaxed. To start with, if the matrix* $\mathbf{A}(c, c)$ *is not symmetric, then there will be an additional term in the inequality* (23)*, which would be proportional to* $\mathbf{A}_{ij} - \mathbf{A}_{ji}$*. This term will be of indefinite sign and can cause further growth in the perturbations of the steady state* c*. In any case, it can only further destabilize the quasi-linearized system. The assumption that the entries of* $\mathbf{A}$ *are uniformly positive amounts to assuming positivity of the weights of the underlying GNN layer. This can be replaced by requiring that the corresponding eigenvalues are uniformly positive. If some eigenvalues are negative, this will cause further instability and only strengthen the conclusion of lack of (exponential) stability. Finally, the assumption that one node is not perturbed during the quasi-linearization is required for the Poincare inequality* (17)*. If this is not true, an additional term, of indefinite sign, is added to the inequality* (23)*. This term can cause further growth of the perturbations and will only add instability to the system. Hence, all the assumptions in Proposition 3.4 can be relaxed and the conclusion of lack of exponential stability of the zero-Dirichlet energy steady state still holds.*