Lipschitz Normalization for Self-Attention Layers with Application to Graph Neural Networks

George Dasoulas [1 2] **Kevin Scaman** [1] **Aladin Virmaux** [1]

Abstract

Attention based neural networks are state of the art in a large range of applications. However, their performance tends to degrade when the number of layers increases. In this work, we show that enforcing Lipschitz continuity by normalizing the attention scores can significantly improve the performance of deep attention models. First, we show that, for deep graph attention networks (GAT), gradient explosion appears during training, leading to poor performance of gradient-based training algorithms. To address this issue, we derive a theoretical analysis of the Lipschitz continuity of attention modules and introduce LipschitzNorm, a simple and parameter-free normalization for self-attention mechanisms that enforces the model to be Lipschitz continuous. We then apply LipschitzNorm to GAT and Graph Transformers and show that their performance is substantially improved in the deep setting (10 to 30 layers). More specifically, we show that a deep GAT model with LipschitzNorm achieves state of the art results for node label prediction tasks that exhibit long-range dependencies, while showing consistent improvements over their unnormalized counterparts in benchmark node classification tasks.

1. Introduction

Over the last few years, attention models became extremely popular in a wide variety of deep learning applications. These architectures made their first appearance in natural language processing and neural machine translation (Bahdanau et al., 2015; Gehring et al., 2017; Vaswani et al., 2017), and gradually became state-of-the-art in multiple machine learning tasks, including sequential data learning (Radford et al., 2018; Luong et al., 2015; Yang et al., 2016), graph classification (Velickovic et al., 2018; Li et al., 2016) and computer vision (Xu et al., 2015). Notably, Vaswani et al.

(2017) showed that efficient deep learning models could be created using attention layers only, leading to the *Transformer* architecture. Compared to convolutional or linear layers, attention layers have the advantage of allowing the selection of key features in the data while being amenable to backpropagation and gradient descent schemes.

Unfortunately, attention models tend to suffer from poor performance when their depth increases, and most applications have a relatively small number of layers (e.g. 6 for the Transformers in Vaswani et al., 2017). While depth is not necessarily synonymous with increased performance, deep architectures showed extremely good performance in many difficult tasks (e.g. image classification) that exhibit complex structural information (Simonyan & Zisserman, 2014). Moreover, for graph neural networks (GNNs) (Hamilton et al., 2017b), the model depth is directly related to the neighborhood size on which the model aggregates information. In such a case, shallow neural networks are fundamentally unable to capture long-range characteristics, and designing deep graph neural networks is thus a subject of extensive research (Li et al., 2018; 2019; 2020; Loukas, 2020).

In this work, we show that enforcing Lipschitz continuity by normalizing the attention scores can significantly improve the performance of deep attention models. To do so, we present *LipschitzNorm*, a normalization scheme for self-attention layers that enforces Lipschitz continuity, and apply this normalization to attention-based GNNs, including graph attention networks (GAT) (Velickovic et al., 2018), and graph transformers (GT) (Yun et al., 2019; Shi et al., 2020). Moreover, we show that, without normalization, gradient explosion appears in these architectures due to a lack of Lipschitz continuity of the original attention mechanism (Kim et al., 2020). Finally, we show that such a normalization allows to build deeper graph neural networks that show good performance for node label prediction tasks that exhibit long-range dependencies.

The remainder of the paper is structured as follows: in Sec. 2, we provide an overview of the related work on attention mechanisms and graph learning models. Then, in Sec. 3, we provide precise definitions for Lipschitz continuity and attention models. In Sec. 4, we present our theoretical analysis and in Sec. 5 we introduce our normal-

---

[1]Noah's Ark Lab, Huawei Technologies France. [2]DaSciM, LIX, École Polytechnique, France.

ization layer, called *LipschitzNorm*. Then, we empirically show in Sec. 6 the connection between Lipschitz continuity and gradient explosion during training. These are followed by the experimental evaluation in Sec. 7.

2. Related Work

Initially designed to extend the capabilities of recurrent neural networks (Bahdanau et al., 2015), attention models rapidly became a highly efficient and versatile model for machine learning tasks in natural language processing (Radford et al., 2018; Yang et al., 2016), computer vision (Xu et al., 2015) and recommender systems (Ying et al., 2018). Recently, novel attention models have been introduced in graph-based systems showing state-of-the-art performance on graph classification (Lee et al., 2018), node classification (Velickovic et al., 2018; Shi et al., 2020) and link prediction (Zhao et al., 2017) tasks.

Attention and Lipschitz Continuity: Although the attention models gain more attraction in the research community, little progress has been made in the theoretical study of the attention. Pérez et al. (2019) showed how attention-based models can be Turing complete and Cordonnier et al. (2020) studied the relationship of self-attention layers and the convolutional networks for image processing. One important direction that can help towards the expressivity of attention models is the analysis of Lipschitz continuity. Even though the computation of tight Lipschitz bounds of neural networks has been proven to be a hard task (Virmaux & Scaman, 2018), a few approaches suggested Lipschitz-based normalization methods for neural networks (Miyato et al., 2018; Gouk et al., 2020). Kim et al. (2020) showed that the standard dot-product self-attention is not Lipschitz continuous, proposing an alternative self-attention layer that satisfies the Lipschitz continuity. However, the Lipschitz variant is only applicable to Transformer-based models.

Attention and Graph Neural Networks: In this work, we study Lipschitz properties of the general form of self-attention from the optimization perspective. We propose a normalization that enforces the attention layer to be Lipschitz and prevents the model from gradient explosion phenomena. Graph Neural Networks (GNNs) is a class of models that suffer from gradient explosion and vanishing as the model depth increases and, thus enforcing the Lipschitz continuity of deep attention-based GNNs can enhance their expressivity. Due to the recent success of GNNs in various real-world applications, there is a growing interest in their expressive power, either investigating how GNNs can be universal approximators (Xu et al., 2019; Dasoulas et al., 2020; Maron et al., 2019) or studying the impact of the depth and width of the models (Li et al., 2018; Loukas, 2020). The second aspect of the depth analysis still has a few unanswered questions, as the majority of the current state-of-the-art models employ shallow GNNs.

Depth in GNNs: Zhao & Akoglu (2020) related the expressivity of graph convolutional networks with the *laplacian oversmoothing* effect and proposed a normalization layer as a way to alleviate it. More recently, Rong et al. (2019) proposed an edge dropping framework on node classification tasks, in order to tackle over-fitting and over-smoothing phenomena and have shown empirically a constant improvement on the original datasets. Li et al. (2019) and Li et al. (2020) introduced frameworks of adaptive residual connections and generalized message-passing aggregators that allow for the training of very deep GCNs. Finally, Loukas (2020) studied the effect of the depth and the width of a graph neural network model and Alon & Yahav (2021) introduced the *oversquashing* phenomenon as a deterioration factor to the performance of the GNNs. However, to our knowledge, no study on the explicit relationship between the gradient explosion and the GNNs has yet been made.

3. Notations and Definitions

In this section, we recall the definitions of attention models as well as Lipschitz continuity. This notion will be central in our analysis and help us understand why gradient explosion appears when training attention models (see Sec. 6).

3.1. Basic Notations

For any matrix $M \in \mathbb{R}^{n \times m}$, we will denote as spectral norm $\|M\|_*$ its largest singular value, $(\infty, 2)$-norm $\|M\|_{(\infty,2)} = \max_i (\sum_j M_{ij}^2)^{1/2}$, and Frobenius norm $\|M\|_F = (\sum_{i,j} M_{ij}^2)^{1/2}$. Moreover, for $\mathbb{X}$ (resp. $\mathbb{Y}$) a vector space equipped with the norm $\| \cdot \|_{\mathbb{X}}$ (resp. $\| \cdot \|_{\mathbb{Y}}$), the operator norm of a linear operator $f : \mathbb{X} \to \mathbb{Y}$ will denote the quantity $\|\|f\|\|_{\mathbb{X},\mathbb{Y}} = \max_{x \in X} \|f(x)\|_{\mathbb{Y}} / \|x\|_{\mathbb{X}}$ and $\|\|f\|\|_{\mathbb{X}} = \|\|f\|\|_{\mathbb{X},\mathbb{X}}$. Finally, the (Fréchet) derivative of a function $f : \mathbb{X} \to \mathbb{Y}$ at $x \in \mathbb{X}$ will denote (when such a function exists) the linear function $\mathbf{D}f_x : \mathbb{X} \to \mathbb{Y}$ such that, $\forall h \in \mathbb{X}$, $f(x + h) - f(x) = \mathbf{D}f_x(h) + o(\|h\|)$.

3.2. Lipschitz Continuity

A function $f : \mathbb{X} \to \mathbb{Y}$ is said to be Lipschitz continuous if there exists a constant L such that, for any $x, y \in \mathbb{X}$, $\|f(x) - f(y)\|_{\mathbb{X}} \leqslant L\|x - y\|_{\mathbb{Y}}$. The Lipschitz constant $L_{\mathbb{X},\mathbb{Y}}(f)$ will denote the smallest of such constants. Moreover, a Lipschitz continuous function f is derivable almost everywhere and (see Federer (1996, Thm 3.1.6))

$$L_{\mathbb{X},\mathbb{Y}}(f) = \sup_{X \in \mathbb{X}} \|\|\mathbf{D}f_X\|\|_{\mathbb{X},\mathbb{Y}} . \qquad (1)$$

The Lipschitz constant controls the perturbation of the output given a bounded input perturbation, and is a direct ex-

tension of the gradient norm to the multi-dimensional case. Indeed, when f is scalar-valued and differentiable, we have $\mathbf{D}f_x(h) = \nabla f(x)^\top h$ and $\|\mathbf{D}f_x\|_F = \|\nabla f(x)\|_2$. In our analysis, we will only consider the Lipschitz constant of attention layers for the Frobenius norm (i.e. the L_2-norm of the *flattened* input and output matrices), and derive upper bounds from the previous formula (see Sec. 4).

3.3. Attention Models

An *attention layer* is a soft selection procedure that uses scores to choose which input vectors to focus on. Before presenting attention layers in their most general form, we first focus on the more simple case with a single vector output in order to provide more intuition to the reader.

Single Output Case: Let $x_1, \ldots, x_n \in \mathbb{R}^d$ be a set of input vectors, and $g : \mathbb{R}^d \to \mathbb{R}$ a score function. Each vector is assigned a score $g(x_i)$ that measures the impact of the input vector on the output through a *softmax* function:

$$\text{Att}(x) = \sum_{i=1}^n \frac{e^{g(x_i)}}{\sum_{j=1}^n e^{g(x_j)}} x_i . \tag{2}$$

In most applications, the score function is linear $g(x) = q^\top x$ where $q \in \mathbb{R}^d$ is a query vector that indicates the direction favored by the attention model.

General Case: In many applications, the output is not a single vector, but a collection of vectors. We thus switch to a matrix notation in order to simplify the definitions. Let $X \in \mathbb{R}^{d \times n}$ be an input matrix whose rows are the input vectors $x_1, \ldots, x_n \in \mathbb{R}^d$. A score function $g : \mathbb{R}^{d \times n} \to \mathbb{R}^{m \times n}$ takes the input matrix and returns scores for each output vector $i \in \{1, \ldots, m\}$ and each input vector $j \in \{1, \ldots, n\}$. This score is usually linear or quadratic ; however, we will see in Sec. 4 that such a generalisation allows to consider more advanced score functions, including overall normalization by a scalar. The probability weights are then computed using a (row-wise) softmax operator softmax $: \mathbb{R}^{m \times n} \to \mathbb{R}^{n \times m}$ taking as input a score matrix $M \in \mathbb{R}^{m \times n}$,

$$\text{softmax}(M)_{ij} = \frac{e^{M_{ij}}}{\sum_{k=1}^n e^{M_{ik}}} . \tag{3}$$

Note that all rows sum to one, and all coordinates are between 0 and 1. Each row can thus be interpreted as a probability distribution over the n input vectors. Finally, the overall attention module $\text{Att} : \mathbb{R}^{d \times n} \to \mathbb{R}^{d \times m}$ returns a matrix whose columns are weighted averages of the inputs:

$$\text{Att}(X) = X \, \text{softmax}(g(X))^\top . \tag{4}$$

Multi-Head Attention: In order to augment the power of attention models, a common trick consists in concatenating multiple independent attention models. These *multi-head* models can thus focus on multiple directions of the input space at the same time, and are generally more powerful in practice. A standard procedure consists in first projecting the input vectors into multiple low-dimensional spaces, and combining the results of all attention layers using a linear function. Let d_I (resp. d_O) be the input (resp. output) dimension, h the number of heads, and $W_1, \ldots, W_h \in \mathbb{R}^{d_I \times d}$ and $W_O \in \mathbb{R}^{d_O \times dh}$ be $h + 1$ matrices, then

$$\text{MultAtt}(X) = W_O \left(\text{Att}(W_1 X) \middle\| \ldots \middle\| \text{Att}(W_h X) \right), \tag{5}$$

where the $\|$ operator denotes row-wise concatenation. In this work, we will consider each attention head separately, using the fact that the Lipschitz constant of multi-head attention can be bounded by that of each attention head.

Theorem 1. *If each attention head is Lipschitz continuous, then multi-head attention as defined in Eq. (5) is Lipschitz continuous and*

$$L_F(\text{MultAtt}) \leqslant L_F(\text{Att}) \|W_O\|_* \sqrt{\sum_{k=1}^h \|W_k\|_*^2} . \tag{6}$$

Transformer Case: For Transformers, $m = n$ and the input matrix is decomposed as $X = (Q\|K\|V)$, where $Q, K, V \in \mathbb{R}^{d \times n}$ represent, respectively, *queries*, *keys* and *values*. The attention model is then

$$\text{Att}(X) = V \text{softmax} \left(\frac{Q^\top K}{\sqrt{d}} \right)^\top . \tag{7}$$

Note that the softmax is not multiplied by the whole input vector X, but only the values V. This is equivalent to projecting the output vectors on a subspace, and thus does not lead to an increase in the Lipschitz constant.

4. The Lipschitz Constant of Attention

As their name suggest, the purpose of attention layers is to select a small number of input vectors (softmax probabilities tend to focus most of their mass on the largest score). Unfortunately, large scores also tend to create large gradients. In order to show this behavior, we first provide a computation of the norm of the derivative of attention layers. Note that the proofs of all lemmas and theorems are provided in the supplementary material.

Derivative of Attention Models: A direct computation using the definition of Eq. (4) and the chain rule gives

$$\mathbf{D}\text{Att}_X(H) = H\text{softmax}(g(X))^\top$$
$$+ X\mathbf{D}\text{softmax}_{g(X)}(\mathbf{D}g_X(H))^\top , \tag{8}$$

where $H \in \mathbb{R}^{d \times n}$ is an input perturbation. We handle both terms separately, leading to the following upper bound on the Lipschitz constant.

Lemma 1. *For any $X \in \mathbb{R}^{d \times n}$, the norm of the derivative of attention models (see Eq. (4)) is upper bounded by:*

$$\|\mathbf{D}\mathrm{Att}_X\|_F \leqslant \|\mathrm{softmax}(g(X))\|_F$$
$$+ \sqrt{2}\|X^\top\|_{(\infty,2)}\|\mathbf{D}g_X\|_{F,(2,\infty)}. \quad (9)$$

Eq. (9) shows that the Lipschitz constant is controlled by two terms: the first one is related to the *uniformity* of the softmax probabilities, while the second one is related to the size of the input and gradient of the score function. In what follows, we will examine these two terms and show that normalizing the scores by a well-chosen scalar allows to control both simultaneously.

Uniformity of the Softmax Probabilities: The first term in Eq. (9) is directly related to *how far* the softmax probabilities are from being uniform. More precisely, we have

$$\|\mathrm{softmax}(g(X))\|_F = \sqrt{\frac{m + \sum_{i=1}^m d_{\chi^2}(S_i, U_n)}{n}}, \quad (10)$$

where S_i is the i-th row of $\mathrm{softmax}(g(X))$, U_n is the uniform distribution over n elements, and $d_{\chi^2}(p,q) = \sum_i q_i(p_i/q_i - 1)^2$ is the χ^2-divergence between p and q (Csiszár & Shields, 2004). Hence, if all attention heads have uniform probabilities, then $d_{\chi^2}(S_i, U_n) = 0$ and $\|\mathrm{softmax}(g(X))\|_F = \sqrt{m/n}$. On the contrary, the distances are maximum when the whole mass of the probabilities is on one element, and in such a case $d_{\chi^2}(S_i, U_n) = n-1$ and $\|\mathrm{softmax}(g(X))\|_F = \sqrt{m}$.

Lemma 2. *For any $M \in \mathbb{R}^{m \times n}$, we have*

$$\sqrt{m/n} \leqslant \|\mathrm{softmax}(M)\|_F \leqslant \sqrt{m}. \quad (11)$$

When $m \gg 1$ (e.g. $m = n$ for Transformers), this implies that the gradients of attention models can be large and lead to the explosive phenomena observed in Sec. 6. Fortunately, controlling the scale of the scores is sufficient to control the uniformity of the probabilities.

Lemma 3. *If all the scores are bounded by $\alpha \geqslant 0$, i.e. for all $i \in \{1, \ldots, m\}$ and $j \in \{1, \ldots, n\}$, $|g(x)_{ij}| \leqslant \alpha$, then*

$$\|\mathrm{softmax}(g(X))\|_F \leqslant e^\alpha \sqrt{\frac{m}{n}}. \quad (12)$$

Hence, the first objective of the normalization is to scale the scores in order to avoid softmax probabilities to put their entire mass on a single vector.

Impact of a Scalar Normalization: Without any additional control, Lemma 1 does not prove the Lipschitz continuity of attention models, as the second term is proportional to the norm of the input matrix $\|X^\top\|_{(\infty,2)}$. For example, if the scores are linear, then their derivative is constant, and the second term in Eq. (9) is not bounded. In order to address this issue, we propose to normalize the score function by a scalar function $c : \mathbb{R}^{d \times n} \to \mathbb{R}_+$:

$$g(X) = \frac{\tilde{g}(X)}{c(X)}, \quad (13)$$

where $\tilde{g}$ is the original score function, and g the normalized one. When $c(X)$ is chosen wisely, this simple normalization is sufficient to obtain a tight bound on the Lipschitz constant of the attention (see Sec. 5).

Theorem 2. *Let $\alpha \geqslant 0$. If, for all $X \in \mathbb{R}^{d \times n}$, we have*

(1) $\|\tilde{g}(X)\|_\infty \leqslant \alpha c(X)$,

(2) $\|X^\top\|_{(\infty,2)}\|\mathbf{D}\tilde{g}_X\|_{F,(2,\infty)} \leqslant \alpha c(X)$,

(3) $\|X^\top\|_{(\infty,2)}\|\mathbf{D}c_X\|_{F,1}\|\tilde{g}(X)\|_{(2,\infty)} \leqslant \alpha c(X)^2$,

then attention models (see Eq. (4)) with score function $g(X) = \tilde{g}(X)/c(X)$ is Lipschitz continuous and

$$L_F(\mathrm{Att}) \leqslant e^\alpha \sqrt{\frac{m}{n}} + \alpha\sqrt{8}. \quad (14)$$

First, note that α controls the scale of all the scores, as assumption (1) implies $\|g(X)\|_\infty \leqslant \alpha$. Thus, when the scores are allowed to reach values of order ≈ 1, and when $m \leqslant n$, Theorem 2 implies that the Lipschitz bound is also of order ≈ 1. The assumptions (1)-(3) of Theorem 2 are rather restrictive, and finding a proper normalization $c(X)$ in the general case is a difficult problem. However, we will see in the next section that a solution can be found in most practical cases of interest.

5. The *LipschitzNorm* normalization

In this section, we present our proposed normalization, the LipschitzNorm in three different settings: Lipschitz, linear and quadratic score functions. These settings cover most of the practical applications, including Transformers, GAT and GT models. All settings are particular instances of a common idea: impose assumptions (1)-(3) of Theorem 2 by dividing by the maximum of input values. A visualization of how LipschitzNorm is applied to an attention model with linear score function is shown in Fig. 1.

Lipschitz scores: When the score function is Lipschitz, the assumptions in Theorem 2 can be met, for $\alpha \geqslant 0$, by

$$g(X) = \frac{\alpha\,\tilde{g}(X)}{\max\left\{\|\tilde{g}(X)\|_{(2,\infty)}, \|X^\top\|_{(\infty,2)}L_{F,(2,\infty)}(\tilde{g})\right\}}. \quad (15)$$
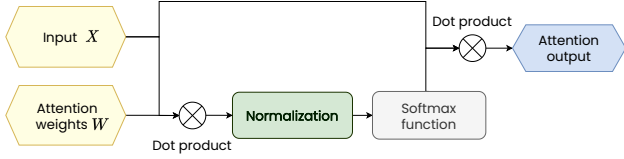
Figure 1. Pipeline of an attention mechanism along with the proposed normalization. For clarity, we assume a linear score function $g(x) = W^\top X$, expressed as a dot product operator.

The denominator is composed of two terms: the first ensures that all the scores are bounded (by α), while the second ensures that the gradient of the normalized scores remains low compared to the scale of the input vector. Note that $\|X^\top\|_{(\infty,2)}$ is the maximum of the norm of input vectors.

Theorem 3. *If the score function $\tilde{g}$ is Lipschitz continuous, then the attention layer with score function as defined in Eq. (15) is Lipschitz continuous and*

$$L_F(\text{Att}) \leqslant e^\alpha \sqrt{\frac{m}{n}} + \alpha\sqrt{8}. \qquad (16)$$

Note that $\alpha = 0$ leads to a uniform distribution and gradient vanishing when $m \ll n$ (for example $m = 1$ when the output is a single vector). On the contrary, a large $\alpha \gg 1$ will lead to very large gradients that may destabilize training. In our experiments, we show that $\alpha = 1$ is a good trade-off that allows to create relatively peaked attention weights, while maintaining a low Lipschitz constant (see Sec. 7).

Linear Score Function: The initial definition of attention layers considers a linear score function $\tilde{g}(X) = Q^\top X$ for $Q \in \mathbb{R}^{d \times m}$. As this score function is Lipschitz continuous, Theorem 3 is directly applicable and leads to the following normalization, called LipschitzNorm,

$$g(X) = \frac{Q^\top X}{\|Q\|_F \|X^\top\|_{(\infty,2)}}. \qquad (17)$$

Note that, contrary to Transformers, the query matrix Q is assumed to be a parameter of the model instead of an input.

Corollary 1. *The attention layer with score function as defined in Eq. (17) is Lipschitz continuous and*

$$L_F(\text{Att}) \leqslant e^1 \sqrt{\frac{m}{n}} + \sqrt{8}. \qquad (18)$$

Transformer Case: For Transformers, the query matrix Q is an input of the model, and the score function is thus quadratic. As quadratic functions are not Lipschitz, Theorem 3 is not applicable. Fortunately, we can adapt the same idea to this setting. As defined in Sec. 3.3, let

$X = (Q\|K\|V)$ be a concatenation of queries, keys and values. Then, the assumptions in Theorem 2 are met by

$$g(X) = \frac{Q^\top K}{\max\{uv, uw, vw\}}, \qquad (19)$$

where $u = \|Q\|_F$, $v = \|K^\top\|_{(\infty,2)}$, and $w = \|V^\top\|_{(\infty,2)}$. Compared to the linear case of Eq. (17), we decompose the input matrix norm $\|X^\top\|_{(\infty,2)}$ into $\|K^\top\|_{(\infty,2)}$ and $\|V^\top\|_{(\infty,2)}$ and return the product between the maximum and second maximum of $\|Q\|_F$, $\|K^\top\|_{(\infty,2)}$, and $\|V^\top\|_{(\infty,2)}$.

Corollary 2. *The attention layer with score function as defined in Eq. (19) is Lipschitz continuous and*

$$L_F(\text{Att}) \leqslant e^{\sqrt{3}} \sqrt{\frac{m}{n}} + 2\sqrt{6}. \qquad (20)$$

Finally, as discussed in Sec. 3.3, we normalize multi-head attention by normalizing each attention head separately. Theorem 1 then directly implies the following bound on the Lipschitz constant of the whole multi-head attention layer:

$$L_F(\text{MultAtt}) \leqslant 11\|W_O\|_* \sqrt{\sum_{k=1}^{h} \|W_k\|_*^2}, \qquad (21)$$

where $W_1, \ldots, W_h$ (resp. W_O) represent the input (resp. output) projection matrices (see Eq. (5)), and $m = n$.

Implementation details: Given an attention model $\mathcal{M}$ with score function g, we define $\mathcal{M}$-Lip as the updated attention model with the application of LipschitzNorm. This normalization requires three steps, that we now provide for the linear and Transformer settings:

1. **Frobenius norm of the queries**: First, we compute the Frobenius norm of Q: $u = \sqrt{\sum_i \|q_i\|_2^2}$.

2. **Input norms**: Then, we compute the maximum 2-norm of the input vectors $v = \max_i \|x_i\|_2$ (or $v = \max_i \|k_i\|_2$ and $w = \max_i \|v_i\|_2$ for Transformers).

3. **Scaling**: Finally, we divide the score function by the product uv (or $\max\{uv, uw, vw\}$ for Transformers).

Each attention head is treated separately, and thus all the norms and maximums are taken *per head*. Moreover, in the case of graph attention, the norms and maximums are computed *neighbor-wise*, i.e. for each node, we compute the maximum of the 2-norms of its neighbors.

6. Gradient Explosion and Vanishing

Similar to the deep neural networks, the design and efficient training of deep attention models has a tight connection with their Lipschitz continuity. In fact, given M
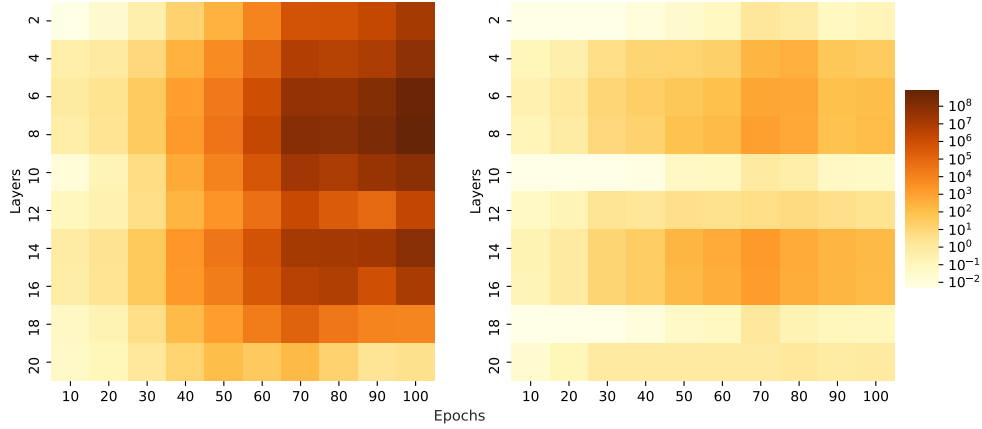
Figure 2. Gradient evolution of attention weights of a 20-layer GAT model for each layer throughout training. Each cell i, j represents the norm of the gradients of the attention weights in the i-th layer and trained until j-th epoch. The left heat map corresponds to the standard GAT without any normalization, where the phenomenon of **gradient explosion** occurs. The right heat map corresponds to the GAT model using LipschitzNorm. The proposed normalization restrains the attention weights from explosion.
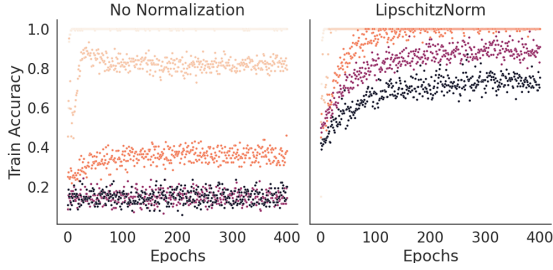

Figure 3. Convergence of train accuracy for a GAT model on node classification task using no normalization (left) and using LipschitzNorm (right).

Lipschitz continuous attention layers $\text{Att}_m(\cdot)$ with Lipschitz constants $l_m = L_F(\text{Att}_m)$, their composition $f = \text{Att}_1 \circ \text{Att}_2 \circ ... \circ \text{Att}_{m-1} \circ \text{Att}_m$ is Lipschitz continuous with Lipschitz constant upper-bounded by

$$L_F(f) \leqslant \prod_{m=1}^{M} L_F(\text{Att}_m). \qquad (22)$$

Eq. (22) implies that there is a multiplicative effect on the gradient flow of an M-layered attention model. Thus, enforcing the attention layer to be Lipschitz continuous with tight Lipschitz bounds can alleviate gradient explosion and allow for the design of deeper attention-based models.

Fig. 2 (left picture) shows that gradient explosion occurs in a deep (20 layers) Graph Attention Network (GAT) (Velickovic et al., 2018) applied to a node classification task on Cora dataset (McCallum et al., 2000). Throughout 100

epochs of training, the gradients of the attention weights in each GAT layer exhibit a steep increase, reaching extremely large value of the order of 10^8. However, Fig. 2 (right picture) shows that LipschitzNorm is able to prevent gradient explosion, and throughout training, the gradients of the attention weights remain stable. Fig. 3 shows that gradient explosion also comes with poor performance, and even a total lack of improvement in training accuracy for a GAT model with 30 layers. Again, LipschitzNorm avoids this behavior and allows for proper training in all regimes, showing that enforcing the Lipschitz continuity of the attention layer can help towards the design of deeper architectures.

7. Experimental Evaluation

We now examine the practical contribution of our normalization LipschitzNorm in real-world and synthetic benchmarks. In Sec. 7.1, we evaluate LipschitzNorm in real-world datasets that present long-range dependencies. In Sec. 7.2, we perform a synthetic study of increasing data and model depth and in Sec. 7.3, we apply LipschitzNorm to attention-based GNNs of increasing model depth in real-world node classification tasks.

7.1. Node Classification with Long-range interactions

In most standard node classification benchmarks, the nodes present short-range dependencies, thus, making the fair evaluation of deeper models a difficult task. Towards a more solid comparison of deep GNNs, Zhao & Akoglu (2020) presented a framework that imposes long-range dependencies in real-world graphs. In particular, for a node classification

Table 1. Classification accuracies for the missing-vector setting. In parentheses we denote the number of layers of the best model chosen for the highlighted accuracy. We denote by '-Pn' the application of PairNorm and by '-Lip' the application of the proposed LipschitzNorm.

	Cora		CiteSeer		Pubmed	
	0%	**100%**	**0%**	**100%**	**0%**	**100%**
GCN	82.5 ± 1.2 (2)	58.8 ± 3.5 (2)	$\mathbf{69.5 \pm 2.1}$ **(2)**	31.3 ± 2.7 (2)	77.9 ± 1.4 (2)	44.9 ± 4.4 (2)
GGNN	81.8 ± 2.0 (2)	68.2 ± 2.5 (6)	68.5 ± 1.9 (3)	40.5 ± 1.4 (5)	78.4 ± 2.1 (4)	56.6 ± 1.9 (4)
GAT	82.3 ± 2.3 (2)	65.3 ± 2.1 (4)	69.3 ± 1.6 (2)	42.8 ± 1.6 (4)	77.4 ± 0.5 (6)	63.1 ± 0.7 (4)
GAT-Pn	78.8 ± 0.6 (4)	73.8 ± 1.2 (12)	67.2 ± 0.8 (4)	$\mathbf{51.7 \pm 1.1}$ **(10)**	77.6 ± 1.6 (8)	70.4 ± 1.1 (12)
GAT-Lip	$\mathbf{83.1 \pm 0.5}$ **(5)**	$\mathbf{75.3 \pm 0.9}$ **(11)**	69.1 ± 1.5 (3)	50.9 ± 1.9 (9)	$\mathbf{78.9 \pm 1.3}$ **(5)**	$\mathbf{73.3 \pm 1.4}$ **(15)**

task let a node attributed graph $\mathcal{D} = (V_u \cup V_l, E, X, U)$, where V_u, V_l are the node sets of the unlabeled and the labeled nodes respectively, E is the edge set, $X \in \mathbb{R}^{n \times d}$ is the node attribute matrix and $U \in \mathbb{N}^{n \times m}$ is the label matrix. For an unlabeled node subset $\mathcal{M} \subseteq V_u$ we remove its node attributes: $\{X_j | j \in \mathcal{M}\}$ and we call this framework *missing-vector setting* with fraction $p = \frac{|\mathcal{M}|}{|V_u|}$.

Dataset and Model Setup: We used three standard node classification datasets Cora, CiteSeer and PubMed (McCallum et al., 2000; Giles et al., 1998). The train/validation/test splits were the same as in (Kipf & Welling, 2017). Following Zhao & Akoglu (2020), for each dataset we had 2 node feature setups: the 0% setup, where no attribute were removed and the 100% setup, where all attributes of the unlabeled nodes were removed. We experimented with 3 models: 1) **GCN**: Graph Convolutional Network (Kipf & Welling, 2017), 2) **GGNN**: Gated Graph Neural Network (Li et al., 2016) and 3) **GAT**: Graph Attention Network (Velickovic et al., 2018). A full description of the experimentation details is provided in the supplementary material.

Results: Table 1 shows the average classification accuracy achieved in the standard and the long-range setting. LipschitzNorm enables the training of deeper GAT layers, as in the missing-vector setting with $p = 100\%$, GAT-Lip (i.e. the GAT model with LipschitzNorm) achieves state-of-the-art classification accuracies in four out of the six setups. Moreover, it is noteworthy that GAT-Lip exhibits a solid performance for both the short-range and the long-range scenarios, outperforming PairNorm.

7.2. Model Depth in Synthetic Trees

An intuitive way to show the ability of a GNN model to capture long-range interactions is to generate synthetic graphs with nodes that are distant and have the same behavior. Thus, following Alon & Yahav (2021), we create the TREES dataset. That is a set of directed trees (from the root to the leaves) of labeled nodes with increasing depth $d \in \{2, ..., 10\}$, where the leaves of the tree are colored *blue*, the root of the tree and the predecessors of the leaves are colored *green* and the rest of the nodes remain uncolored.
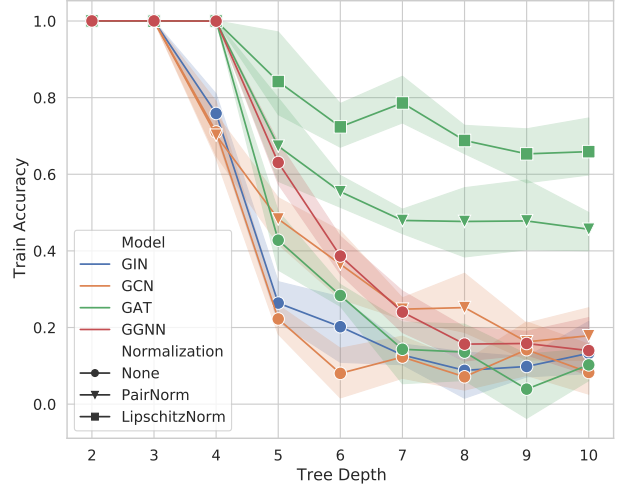

Figure 4. Train accuracies of four GNN models on the TREES dataset. We compare the model performance using: no normalization (circular dots), *PairNorm* (triangular dots) and the proposed *LipschitzNorm* (square dots).

The task is to predict the label of the tree's root green node, according to the label of the other green nodes.

TREES Dataset: The generated tree structure simulate the exponential growth of the receptive field of the nodes, so that the information passes between two distant nodes. For this goal, we created for every tree depth 5000 binary trees and we run each experiment 10 times. Following Alon & Yahav (2021), we did not use explicitly extra blue neighbors, but, instead, we encoded their existence with 1-hot vectors of their cardinality as node attributes of the green nodes.

Model Setup: We compared the performance of Graph Convolutional Network (Kipf & Welling, 2017), Graph Isomorphism Network (Xu et al., 2019), Gated Graph Neural Network (Li et al., 2016) and Graph Attention Network (Velickovic et al., 2018). Each model was implemented with $d + 1$ graph layers, where $d =$ tree depth and the hidden units size is set to 32. Moreover, we used either
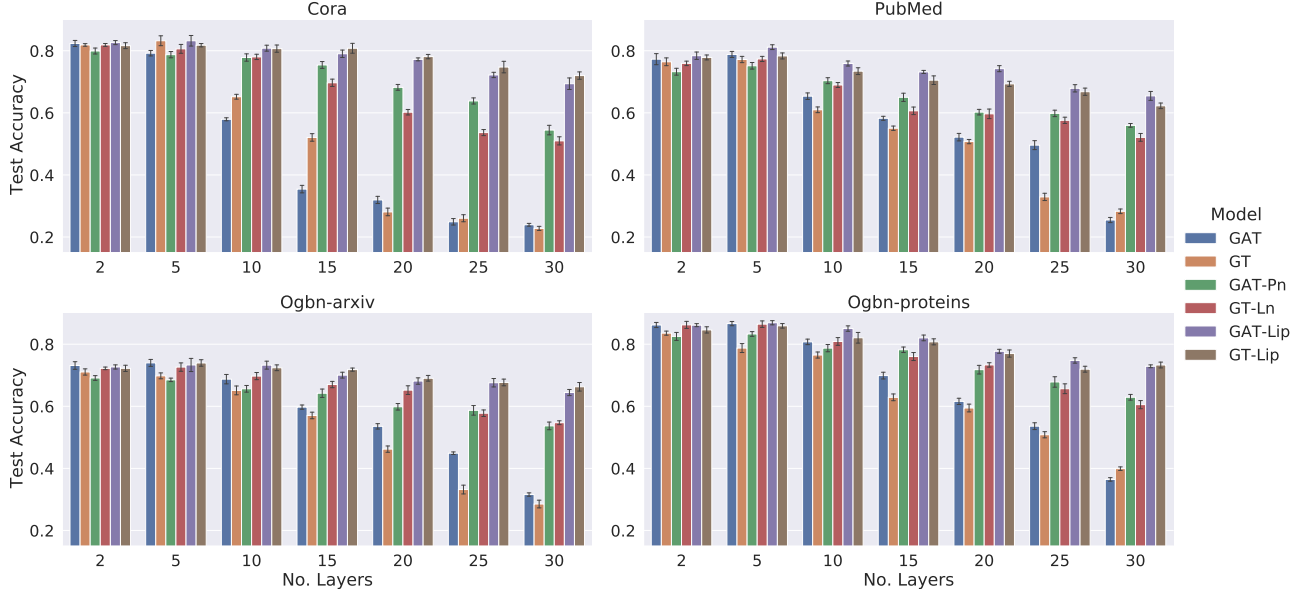
Figure 5. Test accuracies of a Graph Attention Network (GAT) and a Graph Transformer (GT). By '-Lip' we denote the application of *LipschitzNorm*, by '-Ln' the *LayerNorm* and by '-Pn' the *PairNorm*. In Ogbn-proteins dataset, the observed metric is ROC-AUC instead.

no normalization (None case in Fig. 4), PairNorm (Zhao & Akoglu, 2020), or our proposed LipschitzNorm.

Discussion: Fig. 4 shows the train accuracy of the GNN models as the tree depth increases. Aligned with the previously found results, GAT and GGNN exhibit a better behavior with respect to the model depth. Moreover, the application of normalization methods has a significant impact on the performance of deeper models. GAT using the proposed normalization clearly outperforms the other architectures (for tree depth=10, GAT-Lip achieves 68.3% training accuracy, while GAT-Pn achieves 47.8% and all other variants achieve < 25%), showcasing the contribution of LipschitzNorm to the design of deeper architectures.

7.3. Model Depth in Real-World Datasets

In this section, we measure the behavior of LipschitzNorm with respect to an increasing number of layers in real-world datasets. We apply the proposed normalization to two types of attention-based GNNs. We use the well-examined datasets *Cora* and *PubMed* (McCallum et al., 2000) and two datasets from Open Graph Benchmark (Hu et al., 2020): *Ogbn-arxiv* and *Ogbn-proteins*. Details and statistics of the datasets are provided in the supplementary material.

Experimentation Setup: We used two attention-based graph neural networks, as described in Sec. 5: Graph Attention Network (Velickovic et al., 2018) and Graph Transformer (Shi et al., 2020). For the two models, we compared

the contribution of LipschitzNorm with other normalization methods: PairNorm (Zhao & Akoglu, 2020) and LayerNorm (Ba et al., 2016). The number of used graph attention layers was $l \in \{2, 5, 10, 15, 20, 25, 30\}$. We performed cross validation, where the train/validation/test splits in Cora and PubMed were the same as in Kipf & Welling (2017), and for Ogbn-arxiv and Ogbn-proteins we used the same splitting methods as used in Hu et al. (2020). The full experimental setup is described in the supplementary material.

Discussion: In Fig. 5, we highlight the impact of LipschitzNorm on graph neural networks with respect to the model depth. For all four datasets LipschitzNorm enables both GAT and GT to learn and maintain information throughout layers. Even for a large number of layers ($l > 15$), where the models without any normalization fail to converge, the variants using the LipschitzNorm achieve comparable to the state-of-the-art results in the node classification tasks. More importantly, LipschitzNorm outperforms PairNorm and LayerNorm, as it enhances the performance of shallow architectures and maintains it to deeper architectures.

8. Conclusion

In this work, we introduced a novel normalization layer for attention-based neural networks, called LipschitzNorm. We proved that the application of LipschitzNorm enforces the Lipschitz continuity of self-attention layers. In an empirical study, we showed that Lipschitz continuous modules can

prevent from gradient explosion phenomena and, thus, can improve the performance of deep attention models. Focusing on Graph Neural Networks (GNNs), where designing deep models is still a challenging task, we applied LipschitzNorm to standard attention-based GNNs. We showed that LipschitzNorm allows to build deep GNN architectures with strong performance on node classification tasks that exhibit long-range interactions.

References

Alon, U. and Yahav, E. On the bottleneck of graph neural networks and its practical implications. In *International Conference on Learning Representations*, 2021.

Ba, J. L., Kiros, J. R., and Hinton, G. E. Layer normalization. *CoRR*, 2016.

Bahdanau, D., Cho, K., and Bengio, Y. Neural machine translation by jointly learning to align and translate. In *International Conference on Learning Representations*, 2015.

Cordonnier, J.-B., Loukas, A., and Jaggi, M. On the relationship between self-attention and convolutional layers. In *International Conference on Learning Representations*, 2020.

Csiszár, I. and Shields, P. C. Information theory and statistics: A tutorial. *Commun. Inf. Theory*, 1(4):417–528, December 2004.

Dasoulas, G., Dos Santos, L., Scaman, K., and Virmaux, A. Coloring graph neural networks for node disambiguation. In *Proceedings of the Twenty-Ninth International Joint Conference on Artificial Intelligence, IJCAI-20*, pp. 2126–2132, 7 2020. Main track.

Federer, H. *Geometric measure theory*. Grundlehren der mathematischen Wissenschaften (Springer) ; 153. Springer-Verlag Berlin Heidelberg, Berlin, 1996.

Gehring, J., Auli, M., Grangier, D., and Dauphin, Y. A convolutional encoder model for neural machine translation. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 123–135, 2017.

Giles, C., Bollacker, K., and Lawrence, S. Citeseer: an automatic citation indexing system. In *Proceedings of the ACM International Conference on Digital Libraries*, pp. 89–98, 1998.

Gouk, H., Frank, E., Pfahringer, B., and Cree, M. Regularisation of neural networks by enforcing lipschitz continuity. *Machine Learning*, December 2020.

Hamilton, W. L., Ying, R., and Leskovec, J. Inductive representation learning on large graphs. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, pp. 1025–1035, 2017a.

Hamilton, W. L., Ying, R., and Leskovec, J. Representation learning on graphs: Methods and applications. *IEEE Data Eng. Bull.*, 40(3):52–74, 2017b.

Hu, W., Fey, M., Zitnik, M., Dong, Y., Ren, H., Liu, B., Catasta, M., and Leskovec, J. Open graph benchmark: Datasets for machine learning on graphs. *CoRR*, 2020.

Kim, H., Papamakarios, G., and Mnih, A. The lipschitz constant of self-attention. *CoRR*, 2020.

Kingma, D. P. and Ba, J. Adam: A method for stochastic optimization. In *3rd International Conference on Learning Representations (ICLR)*, 2015.

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

Lee, J. B., Rossi, R., and Kong, X. Graph classification using structural attention. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pp. 1666–1674, 2018.

Li, G., Müller, M., Thabet, A., and Ghanem, B. Deepgcns: Can gcns go as deep as cnns? In *The IEEE International Conference on Computer Vision (ICCV)*, 2019.

Li, G., Xiong, C., Thabet, A., and Ghanem, B. Deepergcn: All you need to train deeper gcns. *CoRR*, 2020.

Li, Q., Han, Z., and Wu, X.-M. Deeper insights into graph convolutional networks for semi-supervised learning. In *AAAI*, 2018.

Li, Y., Zemel, R., Brockschmidt, M., and Tarlow, D. Gated graph sequence neural networks. In *International Conference on Learning Representations*, April 2016.

Loukas, A. What graph neural networks cannot learn: depth vs width. In *International Conference on Learning Representations*, 2020.

Luong, T., Pham, H., and Manning, C. D. Effective approaches to attention-based neural machine translation. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pp. 1412–1421, September 2015.

Maron, H., Ben-Hamu, H., Shamir, N., and Lipman, Y. Invariant and equivariant graph networks. In *International Conference on Learning Representations*, 2019.

McCallum, A. K., Nigam, K., Rennie, J., and Seymore, K. Automating the construction of internet portals with machine learning. *Inf. Retr.*, 3(2):127–163, July 2000.

Miyato, T., Kataoka, T., Koyama, M., and Yoshida, Y. Spectral normalization for generative adversarial networks. In *International Conference on Learning Representations*, 2018.

Pérez, J., Marinković, J., and Barceló, P. On the turing completeness of modern neural network architectures. In *International Conference on Learning Representations*, 2019.

Radford, A., Wu, J., Child, R., Luan, D., Amodei, D., and Sutskever, I. Language models are unsupervised multitask learners. 2018.

Rong, Y., Huang, W., Xu, T., and Huang, J. The truly deep graph convolutional networks for node classification. *CoRR*, 2019.

Shi, Y., Huang, Z., Wang, W., Zhong, H., Feng, S., and Sun, Y. Masked label prediction: Unified message passing model for semi-supervised classification. *CoRR*, 2020.

Simonyan, K. and Zisserman, A. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L. u., and Polosukhin, I. Attention is all you need. In *Advances in Neural Information Processing Systems*, volume 30, pp. 5998–6008, 2017.

Velickovic, P., Cucurull, G., Casanova, A., Romero, A., Liò, P., and Bengio, Y. Graph attention networks. In *International Conference on Learning Representations*, 2018.

Virmaux, A. and Scaman, K. Lipschitz regularity of deep neural networks: analysis and efficient estimation. In *Advances in Neural Information Processing Systems*, volume 31, pp. 3835–3844, 2018.

Xu, K., Ba, J., Kiros, R., Cho, K., Courville, A., Salakhudinov, R., Zemel, R., and Bengio, Y. Show, attend and tell: Neural image caption generation with visual attention. In *Proceedings of the 32nd International Conference on Machine Learning*, volume 37, pp. 2048–2057, 2015.

Xu, K., Hu, W., Leskovec, J., and Jegelka, S. How powerful are graph neural networks? In *International Conference on Learning Representations*, 2019.

Yang, Z., Yang, D., Dyer, C., He, X., Smola, A., and Hovy, E. Hierarchical attention networks for document classification. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pp. 1480–1489, June 2016.

Ying, H., Zhuang, F., Zhang, F., Liu, Y., Xu, G., Xie, X., Xiong, H., and Wu, J. Sequential recommender system based on hierarchical attention networks. In *Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence, IJCAI-18*, pp. 3926–3932, 2018. Main track.

Yun, S., Jeong, M., Kim, R., Kang, J., and Kim, H. J. Graph transformer networks. In *Advances in Neural Information Processing Systems*, volume 32, pp. 11983–11993, 2019.

Zhao, L. and Akoglu, L. Pairnorm: Tackling oversmoothing in gnns. In *International Conference on Learning Representations*, 2020.

Zhao, Z., Gao, B., Zheng, V. W., Cai, D., He, X., and Zhuang, Y. Link prediction via ranking metric dual-level attention network learning. In *Proceedings of the Twenty-Sixth International Joint Conference on Artificial Intelligence, IJCAI-17*, pp. 3525–3531, 2017. Main track.

Supplementary material

9. Proofs

9.1. Proof of Theorem 1

Theorem 4. *If each attention head is Lipschitz continuous, then multi-head attention as defined in Eq. (5) of the paper is Lipschitz continuous and*

$$L_F(\text{MultAtt}) \leqslant L_F(\text{Att}) \|W_O\|_* \sqrt{\sum_{k=1}^{h} \|W_k\|_*^2} . \tag{23}$$

Proof. As $\text{MultAtt}(X) = W_O \left(\text{Att}(W_1 X) \Big\| \dots \Big\| \text{Att}(W_h X) \right)$, we have, for any matrix $H \in \mathbb{R}^{d \times n}$,

$$\mathbf{D}\text{MultAtt}_X(H) = W_O \left(\mathbf{D}\text{Att}_{W_1 X}(W_1 H) \Big\| \dots \Big\| \mathbf{D}\text{Att}_{W_h X}(W_h H) \right) . \tag{24}$$

First, note that, for any matrices $A \in \mathbb{R}^{n \times m}$ and $B \in \mathbb{R}^{m \times l}$, we have $\|AB\|_F^2 = \sum_i \|AB_i\|_2^2 \leqslant \sum_i \|A\|_*^2 \|B_i\|_2^2 = \|A\|_*^2 \|B\|_F^2$ by definition of the spectral norm $\|A\|_*$. Hence, we have, for any matrices $X, H \in \mathbb{R}^{d \times n}$,

$$\|\mathbf{D}\text{MultAtt}_X(H)\|_F^2 \leqslant \|W_O\|_*^2 \left\| \left(\mathbf{D}\text{Att}_{W_1 X}(W_1 H) \Big\| \dots \Big\| \mathbf{D}\text{Att}_{W_h X}(W_h H) \right) \right\|_F^2$$

$$= \|W_O\|_*^2 \sum_{k=1}^{h} \|\mathbf{D}\text{Att}_{W_k X}(W_k H)\|_F^2$$

$$\leqslant \|W_O\|_*^2 \sum_{k=1}^{h} \|\mathbf{D}\text{Att}_{W_k X}\|_F^2 \|W_k H\|_F^2$$

$$\leqslant \|W_O\|_*^2 \sum_{k=1}^{h} L_F(\text{Att})^2 \|W_k\|_*^2 \|H\|_F^2 ,$$

which leads to the desired result, as $L_F(f) = \max_X \|\mathbf{D}f_X\|_F = \max_{X,H} \|\mathbf{D}f_X(H)\|_F / \|H\|_F$. $\qquad\square$

9.2. Proof of Lemma 1

Lemma 4. *For any $X \in \mathbb{R}^{d \times n}$, the norm of the derivative of attention models (see Eq. (4) of the paper) is upper bounded by:*

$$\|\mathbf{D}\text{Att}_X\|_F \leqslant \|\text{softmax}(g(X))\|_F + \sqrt{2} \|X^\top\|_{(\infty,2)} \|\mathbf{D}g_X\|_{F,(2,\infty)} . \tag{25}$$

Proof. Using the chain rule on the derivative of $\text{Att}(X) = X\text{softmax}(g(X))^\top$, we immediately obtain, for any $H \in \mathbb{R}^{d \times n}$,

$$\mathbf{D}\text{Att}_X(H) = H\text{softmax}(g(X))^\top + X\mathbf{D}\text{softmax}_{g(X)}(\mathbf{D}g_X(H))^\top , \tag{26}$$

and thus

$$\|\mathbf{D}\text{Att}_X(H)\|_F \leqslant \|H\text{softmax}(g(X))^\top\|_F + \|X\mathbf{D}\text{softmax}_{g(X)}(\mathbf{D}g_X(H))^\top\|_F . \tag{27}$$

First, we have $\|H\text{softmax}(g(X))^\top\|_F \leqslant \|H\|_F \|\text{softmax}(g(X))\|_F$ by multiplicativity of the Frobenius norm. The second term follows from the bound, for any matrices $A \in \mathbb{R}^{d \times n}$ and $B \in \mathbb{R}^{m \times n}$,

$$\|A\mathbf{D}_X\text{softmax}(B)^\top\|_F \leqslant \sqrt{2} \|A^\top\|_{(\infty,2)} \|B\|_{(2,\infty)} , \tag{28}$$

that we will prove below. Assuming that Eq. (28) holds, we have, for any $H \in \mathbb{R}^{d \times n}$,

$$\|\mathbf{D}\text{Att}_X(H)\|_F \leqslant \left(\|\text{softmax}(g(X))\|_F + \sqrt{2} \|X^\top\|_{(\infty,2)} \|\mathbf{D}g_X\|_{F,(2,\infty)} \right) \|H\|_F , \tag{29}$$

and the desired result. Eq. (28) is proven as follows: the derivative of the softmax is given by

$$\mathbf{D}_X \mathrm{softmax}(B)_{ij} = \sum_k \mathrm{softmax}(X)_{ij} \mathrm{softmax}(X)_{ik}(B_{ij} - B_{ik}), \tag{30}$$

and thus

$$
\begin{aligned}
(A\mathbf{D}_X \mathrm{softmax}(B)^\top)_{ij} &= \sum_k A_{ik}\mathbf{D}_X\mathrm{softmax}(B)_{jk} \\
&= \sum_{k,l} A_{ik}\mathrm{softmax}(X)_{jk}\mathrm{softmax}(X)_{jl}(B_{jk} - B_{jl}) \\
&= \sum_k A_{ik}\mathrm{softmax}(X)_{jk}B_{jk} - \sum_l \left(\sum_k A_{ik}\mathrm{softmax}(X)_{jk}\right)\mathrm{softmax}(X)_{jl}B_{jl} \\
&= \sum_k A_{ik}\mathrm{softmax}(X)_{jk}B_{jk} - \sum_l (A\mathrm{softmax}(B)^\top)_{ij}\mathrm{softmax}(X)_{jl}B_{jl} \\
&= \sum_k \mathrm{softmax}(X)_{jk}B_{jk}\left(A_{ik} - (A\mathrm{softmax}(B)^\top)_{ij}\right).
\end{aligned}
$$

Inserting this last equality within the Frobenius norm, we get

$$
\begin{aligned}
\|A(\mathbf{D}_X\mathrm{softmax}(B))^\top\|_F^2 &= \sum_{i,j}\left(\sum_k \mathrm{softmax}(X)_{jk}B_{jk}(A_{ik} - (A\mathrm{softmax}(B)^\top)_{ij})\right)^2 \\
&\leqslant \sum_{i,j,k}\mathrm{softmax}(X)_{jk}B_{jk}^2\left(A_{ik} - (A\mathrm{softmax}(B)^\top)_{ij}\right)^2 \\
&= \sum_{j,k}\mathrm{softmax}(X)_{jk}B_{jk}^2\|A_k^\top - (A\mathrm{softmax}(B)^\top)_j^\top\|_2^2,
\end{aligned}
$$

where the inequality comes from Jensen's inequality applied to the square function (i.e. $\mathbb{E}[Z]^2 \leqslant \mathbb{E}[Z^2]$ for any r.v. Z) and the fact that $\mathrm{softmax}(X)_j$ is a probability distribution. Finally, as $(A\mathrm{softmax}(B)^\top)_j^\top$ is a weighted average of the vectors $A_k^\top$, and is thus in their convex hull, we have $\|A_k^\top - (A\mathrm{softmax}(B)^\top)_j^\top\|_2^2 \leqslant 2\max_k \|A_k^\top\|_2^2$, and thus

$$
\begin{aligned}
\|A(\mathbf{D}_X\mathrm{softmax}(B))^\top\|_F^2 &\leqslant 2\sum_{j,k}\mathrm{softmax}(X)_{jk}B_{jk}^2\|A^\top\|_{(\infty,2)}^2 \\
&\leqslant 2\sum_j \|\mathrm{softmax}(X)_j^\top\|_1\|B_j^\top\|_\infty^2\|A^\top\|_{(\infty,2)}^2 \\
&= 2\|B\|_{(2,\infty)}^2\|A^\top\|_{(\infty,2)}^2,
\end{aligned}
$$

where the second inequality uses the Hölder inequality and the last line is due to $\|\mathrm{softmax}(X)_j^\top\|_1 = 1$. This finishes the proof and leads to the desired inequality. □

9.3. Proof of Lemma 2

Lemma 5. *For any $M \in \mathbb{R}^{m \times n}$, we have*

$$\sqrt{m/n} \leqslant \|\mathrm{softmax}(M)\|_F \leqslant \sqrt{m}. \tag{31}$$

Proof. Let $M \in \mathbb{R}^{m \times n}$, we have

$$\|\mathrm{softmax}(M)\|_F = \left(\sum_{i,j}\mathrm{softmax}(M)_{ij}^2\right)^{1/2}.$$

First, using $\mathrm{softmax}(M)_{ij} \in [0,1]$ and $\sum_j \mathrm{softmax}(M)_{ij} = 1$ for all i, we obtain

$$\|\mathrm{softmax}(M)\|_F = \sqrt{\sum_{i,j}\mathrm{softmax}(M)_{ij}^2} \leqslant \sqrt{\sum_{i,j}\mathrm{softmax}(M)_{ij}} = \sqrt{m}. \tag{32}$$

Then, using Eq.(10) of the paper, we have

$$\|\text{softmax}(g(X))\|_F = \sqrt{\frac{m + \sum_{i=1}^m d_{\chi^2}(S_i, U_n)}{n}} \geqslant \sqrt{\frac{m}{n}}, \tag{33}$$

as the χ^2-divergences are positive (i.e. $d_{\chi^2}(S_i, U_n) \geqslant 0$). $\square$

9.4. Proof of Lemma 3

Lemma 6. *If all the scores are bounded by $\alpha \geqslant 0$, i.e. for all $i \in \{1, \ldots, m\}$ and $j \in \{1, \ldots, n\}$, $|g(x)_{ij}| \leqslant \alpha$, then*

$$\|\text{softmax}(g(X))\|_F \leqslant e^\alpha \sqrt{\frac{m}{n}}. \tag{34}$$

Proof. If, for all $i \in \{1, \ldots, m\}$ and $j \in \{1, \ldots, n\}$, we have $|g(x)_{ij}| \leqslant \alpha$, then

$$\text{softmax}(g(X))_{ij} = \frac{e^{g(X)_{ij}}}{\sum_k e^{g(X)_{ik}}} \leqslant \frac{e^\alpha}{e^\alpha + (n-1)e^{-\alpha}} = \frac{1}{1 + (n-1)e^{-2\alpha}} \leqslant \frac{e^{2\alpha}}{n}. \tag{35}$$

Hence, we have

$$\|\text{softmax}(g(X))\|_F = \sqrt{\sum_{i,j} \text{softmax}(g(X))_{ij}^2} \leqslant \sqrt{\sum_{i,j} \text{softmax}(g(X))_{ij} \frac{e^{2\alpha}}{n}} = e^\alpha \sqrt{\frac{m}{n}}. \tag{36}$$

$\square$

9.5. Proof of Theorem 2

Theorem 5. *Let $\alpha \geqslant 0$. If, for all $X \in \mathbb{R}^{d \times n}$, we have*

(1) $\|\tilde{g}(X)\|_\infty \leqslant \alpha c(X)$,

(2) $\|X^\top\|_{(\infty,2)} \|\mathbf{D}\tilde{g}_X\|_{F,(2,\infty)} \leqslant \alpha c(X)$,

(3) $\|X^\top\|_{(\infty,2)} \|\mathbf{D}c_X\|_{F,1} \|\tilde{g}(X)\|_{(2,\infty)} \leqslant \alpha c(X)^2$,

then attention models (see Eq. (4) of the paper) with score function $g(X) = \tilde{g}(X)/c(X)$ is Lipschitz continuous and

$$L_F(\text{Att}) \leqslant e^\alpha \sqrt{\frac{m}{n}} + \alpha \sqrt{8}. \tag{37}$$

Proof. Using Lemma 1 and Lemma 3 and the assumptions (1), we have,

$$\|\mathbf{D}\text{Att}_X\|_F \leqslant \|\text{softmax}(g(X))\|_F + \sqrt{2}\|X^\top\|_{(\infty,2)} \|\mathbf{D}g_X\|_{F,(2,\infty)}$$

$$\leqslant e^\alpha \sqrt{\frac{m}{n}} + \sqrt{2}\|X^\top\|_{(\infty,2)} \|\mathbf{D}g_X\|_{F,(2,\infty)},$$

where the first inequality is due to Lemma 1 and the second inequality is due to Lemma 3 and assumption (1) (as then $\|g(X)\|_\infty \leqslant \alpha$). Moreover, the derivative of the score function $g(X) = \tilde{g}(X)/c(X)$ gives

$$\mathbf{D}g_X(H) = \frac{\mathbf{D}\tilde{g}_X(H)}{c(X)} - \frac{\mathbf{D}c_X(H)\tilde{g}(X)}{c(X)^2}, \tag{38}$$

and thus,

$$\|\mathbf{D}g_X\|_{F,(2,\infty)} \leqslant \frac{\|\mathbf{D}\tilde{g}_X\|_{F,(2,\infty)}}{c(X)} + \frac{\|\mathbf{D}c_X\|_{F,1} \|\tilde{g}(X)\|_{(2,\infty)}}{c(X)^2}. \tag{39}$$

Finally, using this equation and assumption (2) and (3), we have

$$\|\mathbf{D}\mathrm{Att}_X\|_F \leqslant e^\alpha\sqrt{\frac{m}{n}} + \frac{\sqrt{2}\|X^\top\|_{(\infty,2)}\|\mathbf{D}\tilde{g}_X\|_{F,(2,\infty)}}{c(X)} + \frac{\sqrt{2}\|X^\top\|_{(\infty,2)}\|\mathbf{D}c_X\|_{F,1}\|\tilde{g}(X)\|_{(2,\infty)}}{c(X)^2}$$

$$\leqslant e^\alpha\sqrt{\frac{m}{n}} + \sqrt{2}\alpha + \sqrt{2}\alpha$$

$$\leqslant e^\alpha\sqrt{\frac{m}{n}} + \alpha\sqrt{8}\,,$$

and the desired result. $\qquad\square$

Remark 1. *Note that Theorem 2 still holds if* $\mathrm{Att}(X) = h(X)\,\mathrm{softmax}(g(X))^\top$ *and* $L_F(h) \leqslant 1$ *(i.e. the function* h *is contractive). In such a case, the assumptions become:*

(1) $\|\tilde{g}(X)\|_\infty \leqslant \alpha c(X),$

(2) $\|h(X)^\top\|_{(\infty,2)}\|\mathbf{D}\tilde{g}_X\|_{F,(2,\infty)} \leqslant \alpha c(X),$

(3) $\|h(X)^\top\|_{(\infty,2)}\|\mathbf{D}c_X\|_{F,1}\|\tilde{g}(X)\|_{(2,\infty)} \leqslant \alpha c(X)^2.$

9.6. Proof of Theorem 3

Theorem 6. *If the score function* $\tilde{g}$ *is Lipschitz continuous, then the attention layer with score function as defined in Eq. (15) of the paper is Lipschitz continuous and*

$$L_F(\mathrm{Att}) \leqslant e^\alpha\sqrt{\frac{m}{n}} + \alpha\sqrt{8}\,. \tag{40}$$

Proof. First, as $c(X) = \max\left\{\|\tilde{g}(X)\|_{(2,\infty)}, \|X^\top\|_{(\infty,2)}L_{F,(2,\infty)}(\tilde{g})\right\}/\alpha$, we have $\alpha c(X) \geqslant \|\tilde{g}(X)\|_{(2,\infty)} \geqslant \|\tilde{g}(X)\|_\infty$ and assumption (1) of Theorem 2 is verified. Second, we have $\alpha c(X) \geqslant \|X^\top\|_{(\infty,2)}L_{F,(2,\infty)}(\tilde{g}) \geqslant \|X^\top\|_{(\infty,2)}\|\mathbf{D}\tilde{g}_X\|_{F,(2,\infty)}$ and assumption (2) of Theorem 2 is also verified. Finally, we have

$$\alpha|\mathbf{D}c_X(H)| \leqslant \max\left\{\left|\mathbf{D}\|\tilde{g}(\cdot)\|_{(2,\infty)\,X}(H)\right|, \left|\mathbf{D}\|\cdot\|^\top_{(\infty,2)\,X}(H)\right|L_{F,(2,\infty)}(\tilde{g})\right\}$$

$$\leqslant \max\left\{\|\mathbf{D}\tilde{g}_X(H)\|_{(2,\infty)}, \|H^\top\|_{(\infty,2)}L_{F,(2,\infty)}(\tilde{g})\right\}$$

$$\leqslant \max\left\{\|\|\mathbf{D}\tilde{g}_X\|_{F,(2,\infty)}\|H\|_F, \|H\|_F L_{F,(2,\infty)}(\tilde{g})\right\}$$

$$\leqslant L_{F,(2,\infty)}(\tilde{g})\|H\|_F\,,$$

where the second inequality follows from the triangle inequality $\left|\|X + H\|_{(\infty,2)} - \|X\|_{(\infty,2)}\right| \leqslant \|H\|_{(\infty,2)}$, implying that $|\mathbf{D}\|\cdot\|_{(\infty,2)\,X}(H)| \leqslant \|H\|_{(\infty,2)}$. As a result, we have $\|X^\top\|_{(\infty,2)}\|\mathbf{D}c_X\|_{F,1}\|\tilde{g}(X)\|_{(2,\infty)} \leqslant \|X^\top\|_{(\infty,2)}L_{F,(2,\infty)}(\tilde{g})\|\tilde{g}(X)\|_{(2,\infty)}/\alpha \leqslant \alpha c(X)^2$ (using assumption (1) and (2)) and assumption (3) of Theorem 2 is also verified. We can thus apply Theorem 2 and obtain the desired result. $\qquad\square$

9.7. Proof of Corollary 1

Corollary 3. *The attention layer with score function as defined in Eq. (17) of the paper is Lipschitz continuous and*

$$L_F(\mathrm{Att}) \leqslant e^1\sqrt{\frac{m}{n}} + \sqrt{8}\,. \tag{41}$$

Proof. First, note that replacing $L_{F,(2,\infty)}(\tilde{g})$ in Theorem 2 by any upper bound $M \geqslant L_{F,(2,\infty)}(\tilde{g})$ does not change the result and, as $L_{F,(2,\infty)}(\tilde{g}) = \|Q\|_*$ is hard to compute, we instead prefer the upper bound $\|Q\|_F \geqslant \|Q\|_*$ that is simple and fast to compute. As $\tilde{g}(X) = Q^\top X$ is Lipschitz, we can directly apply Theorem 3 with $\alpha = 1$ and $c(X) = \max\left\{\|Q^\top X\|_{(2,\infty)}, \|X^\top\|_{(\infty,2)}\|Q\|_F\right\}$ to get the desired result. Moreover, the normalization simplifies to $c(X) = \|Q\|_F\|X^\top\|_{(\infty,2)}$, as $\|Q^\top X\|_{(2,\infty)} \leqslant \|Q\|_F\|X^\top\|_{(\infty,2)}$. $\qquad\square$

9.8. Proof of Corollary 2

Corollary 4. *The attention layer with score function as defined in Eq. (19) of the paper is Lipschitz continuous and*

$$L_F(\text{Att}) \leqslant e^{\sqrt{3}}\sqrt{\frac{m}{n}} + 2\sqrt{6}\,. \tag{42}$$

Proof. As defined in Sec. 3.3 of the paper, let $X = (Q\|K\|V)$ be a concatenation of queries, keys and values, and $\text{Att}(X) = V\,\text{softmax}\,(g(X))^\top$. First, note that $\text{Att}(X) = h(X)\,\text{softmax}\,(g(X))^\top$, where $h : X = (Q\|K\|V) \mapsto V$ is a projection. As projections are contractive, Remark 1 implies that Theorem 2 can be used in such a case if we replace $\|X^\top\|_{(\infty,2)}$ by $\|V^\top\|_{(\infty,2)}$ in assumptions (1)-(3). As proposed in Eq.(19) of the paper, let $g(X) = \tilde{g}(X)/c(X)$ where $\tilde{g}(X) = Q^\top K$, $c(X) = \max\{uv, uw, vw\}$, $u = \|Q\|_F$, $v = \|K^\top\|_{(\infty,2)}$, and $w = \|V^\top\|_{(\infty,2)}$. Then, we have

$$\|Q^\top K\|_\infty \leqslant \|Q^\top K\|_{(2,\infty)} \leqslant \|Q\|_F \|K^\top\|_{(\infty,2)} = uv \leqslant c(X)\,,$$

and assumption (1) is verified (with $\alpha = 1$). Moreover, for any perturbation $H = (H_Q\|H_K\|H_V)$, where H_Q, H_K and H_V are the perturbations associated to, respectively, Q, K and V, we have

$$\begin{aligned}
\|D\tilde{g}_X(H)\|_{(2,\infty)} &\leqslant \|Q^\top H_K\|_{(2,\infty)} + \|H_Q^\top K\|_{(2,\infty)} \\
&\leqslant \|Q\|_F\|H_K^\top\|_{(\infty,2)} + \|H_Q\|_F\|K^\top\|_{(\infty,2)} \\
&\leqslant u\|H_K\|_F + v\|H_Q\|_F \\
&\leqslant \sqrt{u^2 + v^2}\|H\|_F\,,
\end{aligned}$$

where the last inequality is due to the Cauchy-Schwarz inequality. Hence, we have

$$\|V^\top\|_{(\infty,2)}\|\mathbf{D}\tilde{g}_X\|_{F,(2,\infty)} \leqslant w\sqrt{u^2+v^2} \leqslant \sqrt{2}c(X)\,,$$

and assumption (2) is verified (with $\alpha = \sqrt{2}$). Finally, we have

$$\begin{aligned}
|\mathbf{D}c_X(H)| &\leqslant \max\{v,w\}\|H_Q\|_F + \max\{u,w\}\|H_K^\top\|_{(\infty,2)} + \max\{u,v\}\|H_V^\top\|_{(\infty,2)} \\
&\leqslant \max\{v,w\}\|H_Q\|_F + \max\{u,w\}\|H_K\|_F + \max\{u,v\}\|H_V\|_F \\
&\leqslant \sqrt{\max\{v,w\}^2 + \max\{u,w\}^2 + \max\{u,v\}^2}\|H\|_F\,,
\end{aligned}$$

where the last inequality is due to the Cauchy-Schwarz inequality, and thus

$$\|\tilde{g}(X)\|_{(2,\infty)}\|V^\top\|_{(\infty,2)}\|\mathbf{D}c_X\|_{F,1} \leqslant uvw\sqrt{\max\{v,w\}^2 + \max\{u,w\}^2 + \max\{u,v\}^2} \leqslant \sqrt{3}uvw\max\{u,v,w\} \leqslant \sqrt{3}c(X)^2\,,$$

and assumption (3) is verified (with $\alpha = \sqrt{3}$). Hence, Theorem 2 with $\alpha = \sqrt{3}$ is applicable and immediately provides the desired result. $\square$

10. Experiments

In this section we report the dataset and experimentation setup in Section 7.1 and Section 7.3.

10.1. Datasets

Here, we present the details of the examined real-world datasets, that were used in Section 7.1 and Section 7.3.

- **Cora, CiteSeer and PubMed** are citation networks (McCallum et al., 2000; Giles et al., 1998). Nodes correspond to research publications and edges encode citation links. All three datasets contain node attributes, that are sparse bag-of-words representations for each document (1433,3703, and 500-dimensional respectively).

- **Ogbn-proteins:** is a proteins interactions network, where each node representts a protein and each edge indicate biological interactions between proteins (e.g homology, co-expression, etc.) (Hu et al., 2020). The dataset contains 8-dimensional edge attributes, where each dimension corresponds to the strength of the interaction type and 8-dimensional node attributes, that is one-hot encodings of the 8 species that a protein comes from.

- **Ogbn-arxiv** is a citation network with directed edges, where each node corresponds to an arXiv paper and the edges denote citations from one paper to another (Hu et al., 2020). The dataset contains node attributes, that are averaged word embeddings of the titles and the abstracts of dimensionality 128. The label of each node is the subject area of the paper and can take 40 values.

In Table 2, we report the statistics of the datasets.

Table 2. Datasets statistics. All datasets consist of a single graph. All node classification tasks are single-label, except *Ogbn-proteins*, which is multi-label. Attributes correspond to node features, except for *Ogbn-proteins* dataset, where attributes are a summation over the node and edge features.

Dataset	# Nodes	# Edges	# Attributes	# Classes
Cora	2,708	5,429	1433	7
CiteSeer	3,327	4,732	3703	6
PubMed	19,717	44,338	500	3
Ogbn-arxiv	169,343	1,166,243	128	40
Ogbn-proteins	132,534	39,561,252	16	2 (112-label)

10.2. Experimentation details for missing-vector setting

Next, we present the experimentation setup that was followed in Section 7.1. This experiment corresponds to a **node classification task** under the *missing vector* setting, as suggested in Zhao & Akoglu, 2020. In our experiments, we used the Adam optimizer (Kingma & Ba, 2015) with a weight decay $L = 5 * 10^{-4}$ and the initial learning rate was set in $\{0.1, 0.01, 0.005, 0.001\}$. We have run each experiment 5 times for 1000 epochs. The evaluation metric is the standard validation classification accuracy for all three datasets.

Model Selection: For all three GNN models, i.e GCN, GGNN, GAT and the normalization scenarios we performed cross-validation with predefined train/validation/test splits. For a fair comparison we used the same splits for all three datasets (Cora, CiteSeer and PubMed) as reported and used in Kipf & Welling, 2017.

Hyper-parameter tuning: We performed grid-search to tune the hyper parameters. The hyper-parameters that were tuned are the following:

- **Number of GNN layers**: For all models and datasets, we used l GNN layers where $l \in \{1, 2, 3, .., 20\}$.

- **Hidden units size**: The dimensionality of the hidden units in all models was in $\{8, 16, 32, 64, 128\}$.

- **Attention heads**: In the case of the GAT model, the attention heads that we used were in $\{1, 2, 4, 8\}$.

- **Dropout ratio**: The dropout ratio was set in $\{0, 0.5\}$.

10.3. Experimentation details for real-world datasets with respect to the model depth

In this section, we present the setup of the experimentation in Section 7.3. Same with Section 7.1, this experiment is a **node classification task**, where we evaluate the performance of GNN models with respect to increasing model depth. We used again the Adam optimizer (Kingma & Ba, 2015) with a weight decay $L = 5 * 10^{-4}$ and the initial learning rate was set in $\{0.1, 0.01, 0.005, 0.001\}$.

Model Selection: We performed, again, for all models and datasets cross-validation with predefined train/validation/test splits and reported the best achieved validation accuracy. For Cora and PubMed, as in Section 7.1, we used the same splits as in Kipf & Welling, 2017. For the other two datasets we have:

1. *Ogbn-arxiv*: We used the same splitting method as used in Hu et al., 2020. Specifically, the train split corresponds to the papers published until 2017, the validation split to the ones published in 2018 and the test split to the ones published in 2019. We used a full-batch training.

2. *Ogbn-proteins* For this dataset, we used, also, the same splitting method as in Hu et al., 2020. That is we split the nodes according to the node labels and in particular grouping according to the protein species. Similar to Shi et al., 2020, we used neighbor sampling (Hamilton et al., 2017a) as a sampling method, due to the size of the graph.

Model Depth: In order to examine the model behavior under the depth increase, for each architecture we used models consisting of l GNN layers, where $l \in \{2, 5, 10, 15, 20, 25, 30\}$. We run each experiment 5 times and we keep the configuration with the best average accuracy.

Hyper-parameter tuning: For each model depth and GNN model, we performed grid-search for hyper-parameter tuning. The hyper-parameters that were tuned are the following:

1. **Graph Attention Network** (Velickovic et al., 2018): The dimensionality of the hidden units was set in $\{8, 16, 64, 128\}$. The number of attention heads was selected between $\{1, 2, 4, 8\}$ and we experimented over two standard aggregators of the attention heads: a) *concatenation* and b) *averaging* of the attention heads. The dropout of the attention weights was set in $\{0, 0.2, 0.5\}$.

2. **Graph Transformer** from the UNIMP framework (Shi et al., 2020): The hidden dimensionality was selected from $\{8, 16, 64, 128\}$ and the number of attentions heads from $\{1, 2, 4\}$. We tested *concatenation* and *averaging* of the attention heads and the dropout of the attention weights was set in $\{0, 0.5\}$.