CauchyGCN: Preserving Local Smoothness in Graph Convolutional Networks via a Cauchy-Based Message-Passing Scheme and Clustering Analysis

Peiyu Liang[1,2], Hongchang Gao[1], and Xubin He[1]

[1] Temple University, Philadelphia PA 19122, USA
[2] peiyu.liang@temple.edu

Abstract. In Graph Convolutional Networks (GCNs), a message-passing scheme explicitly learns and reasons node representations via aggregation and propagation of neighboring information over the graph topology. Most existing message-passing schemes are grounded in Laplacian smoothing, which seeks to maintain the similarity of node representations in the hidden feature space (local smoothness) among neighboring nodes, ensuring their labeling consistency (global smoothness). This often leads to Laplacian smoothing imposing strict penalization on distant neighbors. Because some distant neighbors are inter-class or represent some necessary intra-class patterns, strict penalization of distant neighbors can fail to preserve local smoothness effectively as expected thus introducing noise, mixing representations, and failing to capture valuable hidden patterns. Although recent research has introduced various strategies, including graph filters, k-hop jumps, and bounded penalties to tackle this issue, these methods often fall short of explicitly capturing and preserving the local smoothness over the original topology. In this paper, we present CauchyGCN, which enhances preserving local smoothness in a more interpretable approach. CauchyGCN comprises two key components: 1) a Cauchy smoothing message-passing scheme that explains and preserves local smoothness in each hidden layer, and 2) an unsupervised clustering analysis that simultaneously improves the classifier's capacity to learn both local and global smoothness. We conduct comprehensive experiments using five benchmark datasets to assess the performance of CauchyGCN in semi-supervised node classification tasks compared to state-of-the-art GCNs.

Keywords: Graph Neural Networks · Message-Passing Scheme

1 Introduction

Graph Convolutional Networks (GCNs) have gained increasing attention in studying entities (nodes), and entities' relationships (edges) in various graph-structured large data networks, such as citation networks [27], social media networks [22], and recommendation systems [6]. In GCNs, the layer-wise message-passing scheme

performs learning and reasoning in node representations by aggregating and propagating neighboring information over the graph topology. Recent studies [7, 10, 12, 25, 26] have emphasized the importance of the layer-wise message-passing scheme for node information denoising and topological information smoothing to better enhance downstream task performance.

Despite the numerous advancements in message-passing schemes, most still heavily rely on Laplacian smoothing [1, 4]. Laplacian smoothing operates under the assumption that neighboring nodes belong to the same class, and it enforces strict penalization on distant neighbors intending to maintain the proximity of representations among neighboring nodes (local smoothness) to ensure labeling consistency (global smoothness). However, the strict penalization of distant neighbors often results in Laplacian smoothing being less effective than expected at preserving local smoothness. Several factors contribute to this ineffectiveness. First, distant neighbors could belong to different classes [14]. For instance, in widely used graph datasets like Cora [16] and CiteSeer [8], the inter-class neighbors account for approximately 19% and 26%, respectively. Message passing between inter-class neighbors introduces noise and leads to the mixing of representations [3, 11]. Second, some distant neighbors may represent critical intra-class patterns in sparsely connected regions of the graph [12]. Excessive penalization of these patterns fails to capture valuable hidden proximal patterns in local smoothness. Designing a new message-passing scheme beyond Laplacian smoothing is thus essential to solving these issues.

To tackle the abovementioned challenges in Laplacian smoothing-based GCNs, we propose CauchyGCN which focuses on capturing and preserving local smoothness in an interpretable approach. CauchyGCN achieves local smoothness preservation from two key components in GCNs: the layer-wise message-passing scheme and unsupervised clustering analysis. Specifically, we design Cauchy smoothing, which leverages the desired properties of Cauchy distribution, to capture and preserve the proximal patterns and similarity relationships over the underlying graph topology. In contrast to the emphasis on the variant of the distant neighbors in Laplacian smoothing, Cauchy smoothing emphasizes the variant of the close neighbors. We further combine Cauchy smoothing and Laplacian smoothing and strike a balance between them as a new message-passing scheme in CauchyGCN. Our message-passing scheme reduces noise that might lead to the mixing of inter-class node representations and mitigates the penalization of non-smooth intra-class variations, while also enhancing the capture and preservation of valuable local smoothness, encompassing both proximal patterns and similarity relationships. Moreover, we introduce an end-to-end unsupervised clustering analysis, which simultaneously enhances global smoothness and improves both inter-cluster distinction and intra-cluster cohesion.

Our main contributions are as follows: 1) This is the first approach of leveraging Cauchy distribution to preserve local smoothness in GNNs through a new design message-passing scheme. 2) We jointly learn node representations and clustering analysis to improve both local and global smoothness. 3) Extensive

experiments demonstrate that CauchyGCN achieves competitive performance in semi-supervised node classification compared to existing methods.

2 Graph Notations and Related Work

We begin by providing the graph notations and a general graph convolutional layer, then discuss the recent advancements in message-passing schemes in GCNs.

2.1 Graph Notations

Let $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ denotes an undirected graph that contains a node set $\mathcal{V} = \{v_i : i = 1, \ldots, |\mathcal{V}|\}$, and edge set $(i, j) \in \mathcal{E}$. We use $|\cdot|$ to denote the size of a set. To represent $\mathcal{G}$, there is a node matrix $\mathbf{x} = (x_1, \ldots, x_{|\mathcal{V}|})$ with node v_i has a node representation x_i, and an adjacency matrix $\mathbf{A} \in \mathbb{R}^{|\mathcal{V}| \times |\mathcal{V}|}$ represents edges, where $A_{ij} = 1$ denotes a neighboring node pair (v_i, v_j), $i \neq j$. The adjacency matrix with a self-loop effect is denoted as $\tilde{\mathbf{A}} = \mathbf{A} + \mathbf{I}$, where $\mathbf{I}$ is an identity matrix of size $|\mathcal{V}|$. A diagonal degree matrix $\tilde{\boldsymbol{D}} = diag(\tilde{d}_1, \ldots, \tilde{d}_N)$ is based on $\tilde{\mathbf{A}}$, where $\tilde{d}_i = \sum_j \tilde{A}_{ij}$. Then the normalized Laplacian matrix that represents the graph $\mathcal{G}$ is denoted as $\tilde{\boldsymbol{L}} = \mathbf{I} - \hat{\tilde{\mathbf{A}}}$, where $\hat{\tilde{\mathbf{A}}} = \tilde{\boldsymbol{D}}^{-\frac{1}{2}} \tilde{\mathbf{A}} \tilde{\boldsymbol{D}}^{-\frac{1}{2}}$ is the normalized adjacency matrix.

2.2 Graph Convolutional Layer

Considering a node representation x_v, a general graph convolutional layer in GCNs involves projection and propagation layers [20],

$$\begin{aligned} x_v^{(l)} &= x_v^{(l-1)} \boldsymbol{\Theta}^{(l)} \qquad \text{(Projection)} \ , \\ x_v^{(l+1)} &= f^{(l)} \left(x_v^{(l)}, x_{\mathcal{N}(v)}^{(l)} \right) \quad \text{(Propagation)} \ , \end{aligned} \tag{1}$$

where $l \in [1, L]$ stands for the layer index with L being the last layer. Here, the projection layer updates node representation $x_v^{(l-1)} \in \mathbb{R}^{d_{l-1}}$ from the previous layer $(l-1)$ through a layer-wise trainable projection matrix $\boldsymbol{\Theta}^{(l)} \in \mathbb{R}^{d_{l-1} \times d_l}$ to a new representation $x_v^{(l)} \in \mathbb{R}^{d_l}$, where d_{l-1} and d_l are the hidden feature space at layer $l-1$ and l, respectively. The propagation layer leverages a specified message-passing scheme $f(\cdot)$, further updates $x_v^{(l)}$ by incorporating neighboring information $x_{\mathcal{N}(v)}^{(l)}$ within the current hidden feature space, where $\mathcal{N}(v)$ denotes the set of neighbor nodes of the central node v. The corresponding layer-wise propagation rule [10],

$$\mathcal{O} = \arg\min_{x_v^{(l+1)}} \underbrace{\|x_v^{(l+1)} - x_v^{(l)}\|^2}_{\mathcal{L}_0} + \lambda \underbrace{F^{(l)} \left(x_v^{(l)}, x_{\mathcal{N}(v)}^{(l)} \right)}_{\mathcal{L}_{reg}}, \tag{2}$$

optimizes the network to capture significant graph relations within the current hidden feature space. In Eq. (2), $\mathcal{L}_0$ enforces the network to maintain label consistency by controlling the distance between the node representation in the two consecutive layers (layer l and layer $l+1$), which can be beneficial for tasks like node classification. $\mathcal{L}_{reg}$ with the hyperparameter $\lambda \in [0,1]$ ensures that the message-passing process adheres to certain smoothness patterns and denoising constraints. In the forward pass, obtaining an optimal or sub-optimal solution for $x_v^{(l+1)}$ via $\mathcal{L}_{reg}$ requires an appropriate optimization technique for node representations x. For example, when F is convex and differentiable, the message-passing scheme can be represented as $f(x) = \frac{\partial F}{\partial x}$. In the following sections, we use $\mathbf{x}$ (equivalent to $\mathbf{x}^{(l)}$) to denote the input representation and $\mathbf{h}$ (equivalent to $\mathbf{x}^{(l+1)}$) to denote the output representation in layer l.

2.3 Laplacian smoothing-based GCNs

Laplacian smoothing [1,4] is designed to enhance the smoothness of node representation over the graph topology:

$$tr(\mathbf{h}^T \tilde{L}\mathbf{h}) = \frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n} \hat{\tilde{A}}_{ij}(\mathbf{h}_i - \mathbf{h}_j)^2 \ , \tag{3}$$

where the magnitude of $tr(\mathbf{h}^T \tilde{L}\mathbf{h})$ is largely influenced by the differences between distant neighbors. When applied to GCNs, let $\mathcal{L}_{reg}$ in Eq. (2) adheres to Laplacian smoothing [10], such that,

$$\mathcal{O}_{\text{GCN}} = \arg\min_{\mathbf{h}} \|\mathbf{h} - \mathbf{x}\|^2 + \lambda \cdot tr(\mathbf{h}^T \tilde{\boldsymbol{L}}\mathbf{h}) \ . \tag{4}$$

This encourages the local smoothness of node representations during the message-passing process at each layer, with particular emphasis on the smoothness between distant neighbors (see Proposition 1). Eq. (4) corresponds to the propagation rule in GCN [10], denoted as $\mathcal{O}_{\text{GCN}}$, and stands as the fast and vanilla benchmark in realm of GCNs.

Proposition 1. (Laplacian Smoothing [4]) *Laplacian smoothing focuses on preserving local smoothness between the node i and its distant neighbor j during gradient descent. The update rule is given by: $(\boldsymbol{h}_i - \boldsymbol{h}_j)^* = (\boldsymbol{h}_i - \boldsymbol{h}_j) - 2\eta\tilde{\boldsymbol{L}}(\boldsymbol{h}_i - \boldsymbol{h}_j)$, where η is the learning rate. The correction term $2\eta\tilde{\boldsymbol{L}}(\boldsymbol{h}_i - \boldsymbol{h}_j)$ penalizes the difference between the node representations of node i and j. Consequently, the strict penalization is imposed on the substantial difference between $\boldsymbol{h}_i$ and $\boldsymbol{h}_j$.*

The message-passing scheme corresponding to the propagation rule in Eq. (4) can be obtained by performing a single gradient descent step at $\mathbf{x}$ [10],

$$\begin{aligned} \mathbf{h}^*_{\text{GCN}} &= \mathbf{x} - \eta \frac{\partial \mathcal{O}_{\text{GCN}}}{\partial \mathbf{h}}|_{\mathbf{h}=\mathbf{x}} \\ &= x - \eta\left(2\left(\mathbf{h} - \mathbf{x}\right) + 2L\mathbf{h}\right) \\ &= (D^{-1/2}\tilde{A}D^{-1/2} + L - 2\eta L)\mathbf{x} = \hat{\tilde{\mathbf{A}}}\mathbf{x} \ , \end{aligned} \tag{5}$$

with a step size of $\eta = \frac{1}{2}$. As Laplacian smoothing represents a smooth and convex l_2-based regularization, the updated node representation $\mathbf{h}^*$ can be obtained as a closed-form solution.

2.4 Other Advancements

Because neighboring nodes can belong to different classes or be distantly embedded, the fast and vanilla framework offered by GCN [10] introduces noise and mixes representations during the message-passing process as it assumes the constant importance of neighboring information. To tackle this issue, a line of studies relax this extreme assumption and strive to preserve local smoothness based on the original data's locality from either spectral or spatial approaches.

Spectral approaches are theoretically based on graph signal processing, where advancements delve into refining graph filter definitions to enhance the processing of the graph's frequency domain through graph Fourier transforms. PPNP/ APPNP [7] involves adjusting a node's neighborhood through teleport probability using personalized PageRank [18] on the graph filter. GNN-LF/HF [28] further refine the graph filter in PPNP/APPNP into low-/high-pass filtering kernels for k-hop neighborhoods, thereby enhancing its adaptability to accommodate arbitrary coefficients of polynomial filters by introducing more adjustable factors.

Spatial approaches directly exploit the graph topology, where advancements aim at improving the capture of connectivity among neighboring nodes. GAT [23] leverages attention mechanisms to emphasize the importance between neighbors. GraphSAGE [9] sampling on the neighbors and learns different local patterns through different aggregation methods. JKNet [25] extends its reach beyond 1-hop neighborhoods using hierarchical neighborhood information. UGNN [15] employs the node's intra-class rate as a neighbor-information scalar. ElasticGNN [12] introduces a smoothing strategy based on the l_1-norm with a soft-thresholding operator to restrict distant neighboring information.

Different from these advancements, CauchyGCN distinguishes itself by capturing and preserving local smoothness of the nonlinearity but proximity pattern among closely embedded and mutually connected neighbors, adhering to the Cauchy distribution. We also introduce a weight factor following a Gaussian distribution to explain which neighbors should preserve their local smoothness.

3 CauchyGCN

In this section, we begin by introducing Cauchy smoothing for capturing and preserving local smoothness in an interpretable approach through a new message-passing scheme. We then detail the new message-passing scheme, which involves both Cauchy smoothing and Laplacian smoothing and strikes a balance between these two smoothing strategies. After that, we present an end-to-end unsupervised clustering analysis aimed at further improving both global and local smoothness. Lastly, we elucidate the optimization process of CauchyGCN.

3.1 Cauchy Smoothing

While Laplacian smoothing prioritizes smoothness between distant neighbors, it overlooks smoothness between closely embedded and more mutually connected neighbors. We propose a complementary strategy to address this limitation and enhance local smoothness.

Cauchy distribution has found applications in various domains [2,13] to align objects that carry proximal representations,

$$\phi(x, x_0, \gamma) = \frac{1}{\pi}\left[\frac{\gamma}{(x - x_0)^2 + \gamma^2}\right] . \tag{6}$$

Here, x_0 signifies the location factor, $\gamma > 0$ is a scale parameter, and $\frac{1}{\pi}$ is a constant that is omitted in our proposed method. Inspired by its desirable nonlinear properties and a crucial proposition (see Proposition 2), we introduce Cauchy smoothing to preserve the local smoothness of the nonlinearity but proximity pattern among closely embedded and mutually connected neighbors. Consider a central node i and j as one of its neighboring nodes. Our objective is to maximize the magnitude of the Cauchy distribution, aiming to minimize the distance between their representations ($\mathbf{h}_i$ and $\mathbf{h}_j$),

$$\max_{\mathbf{h}_i} \frac{\gamma}{(\mathbf{h}_i - \mathbf{h}_j)^2 + \gamma^2} . \tag{7}$$

Proposition 2. (Cauchy Distribution) *Let γ represent a positive constant. In the Cauchy distribution ϕ, the maximum occurs at $\frac{1}{\pi\gamma}$ when $x = x_0$. The magnitude of ϕ decreases as the difference between x and x_0 increases.*

Additionally, we introduce a weight factor w_{ij} to reason and determine which neighbors should have their local smoothness preserved. This assessment takes into account both the graph topology and the hidden topological pattern,

$$w_{ij} = A_{ij} \cdot e^{-d_{ij}/\sigma^2} . \tag{8}$$

The term A_{ij} reflects the connectivity of neighbors in the graph topology. $d_{ij} = \|\mathbf{h}_i - \mathbf{h}_j\|^2$ measures the distance between neighboring nodes under the hidden topological pattern, adhering to a Gaussian distribution. Because the Gaussian distribution decays more rapidly than the Cauchy distribution when their parameters γ in Eq. (7) and σ in Eq. (8) are identical, the weight factor compels Cauchy smoothing to preserve local smoothness among closely embedded neighbors following the Gaussian distribution but ensures proper strict penalizations following the Cauchy distribution.

As a result, the Cauchy smoothing is defined as,

$$\max_{\mathbf{h}_i} \frac{w_{ij} \cdot \gamma}{(\mathbf{h}_i - \mathbf{h}_j)^2 + \gamma^2} \equiv \min_{\mathbf{h}_i} \frac{-w_{ij} \cdot \gamma}{(\mathbf{h}_i - \mathbf{h}_j)^2 + \gamma^2} . \tag{9}$$

Proposition 3 illustrates the insights into preserving local smoothness among closely embedded and mutually connected neighbors via Cauchy smoothing.

Proposition 3. (Cauchy smoothing) *Cauchy smoothing focuses on preserving local smoothness between node i and its close neighbor j during gradient descent. The update rule is given by:* $(\boldsymbol{h}_i - \boldsymbol{h}_j)^* = (\boldsymbol{h}_i - \boldsymbol{h}_j) + 2\eta \frac{w_{ij} \cdot \gamma(\boldsymbol{h}_i - \boldsymbol{h}_j)}{((\boldsymbol{h}_i - \boldsymbol{h}_j)^2 + \gamma^2)^2}$, *with* η *representing the learning rate. The correction term* $2\eta \frac{w_{ij} \cdot \gamma(\boldsymbol{h}_i - \boldsymbol{h}_j)}{((\boldsymbol{h}_i - \boldsymbol{h}_j)^2 + \gamma^2)^2}$ *penalizes the disparity between the node representations of node i and j only when there is a sufficiently small difference between* $\boldsymbol{h}_i$ *and* $\boldsymbol{h}_j$*, with the condition that the weight factor* $1 \geq w_{ij} > 0$ *follows a Gaussian distribution.*

3.2 Message-Passing Scheme in CauchyGCN

In terms of preserving local smoothness, Laplacian smoothing emphasizes the variance among distant neighbors, while Cauchy smoothing emphasizes the variance among close neighbors. It is reasonable to incorporate both Laplacian smoothing (Eq. (3)) and Cauchy smoothing (Eq. (7)) to capture local smoothness from distinct perspectives while necessitating a balancing estimator to help mitigate the strict penalization from either aspect. Therefore, the layer-wise propagation rule in CauchyGCN is defined as:

$$\mathcal{O}_c = \arg\min_{\mathbf{h}} \|\mathbf{h} - \mathbf{x}\|^2 + \lambda_1 \cdot tr(\mathbf{h}^T \tilde{\boldsymbol{L}} \mathbf{h}) - \lambda_2 \sum_i \sum_j \frac{w_{ij} \cdot \gamma}{\|\mathbf{h}_i - \mathbf{h}_j\|^2 + \gamma^2} \ , \tag{10}$$

where $\lambda_1 \in [0, 1]$ and $\lambda_2 \in [0, 1]$ are parameters that scale the penalties from Laplacian smoothing and Cauchy smoothing, respectively.

Now, we need to solve the layer-wise propagation rule in Eq. (10) to illustrate the message-passing scheme of CauchyGCN. However, the difficulty is introduced since the two smoothing strategies are coupled by the components of $\mathbf{h}$. To address this issue, we find inspiration in splitting methods [19], particularly those designed for solving optimization problems of the form $f(x) = l(x) + u(x) + \varphi(Ax)$. This problem is equivalent to the one presented in Eq. (10). By reformulating the problem, we convert it from a three-function problem to a more manageable two-function form, expressed as $f(x) = \mathcal{L}(x) + \epsilon\varphi(x)$, where $\mathcal{L}(x) = l(x) + u(x)$ and $\epsilon \in [0, 1]$ is a scalar. Following the splitting method, Eq. (10) is reformulated to: $\mathcal{L}(\mathbf{h}) = \|\mathbf{h} - \mathbf{x}\|^2 + \lambda_1 \mathrm{tr}(\mathbf{h}^T \tilde{\boldsymbol{L}} \mathbf{h})$ and $\varphi(\mathbf{h}) = \lambda_2 \sum_i \sum_j \frac{w_{ij} \cdot \gamma}{\|\mathbf{h}_i - \mathbf{h}_j\|^2 + \gamma^2}$, aiming to find an optimal solution of $\mathbf{h}$. This allows us to approach the optimization step-by-step. First, we solve the optimization problem in $\mathcal{L}(\mathbf{h})$, which intriguingly aligns with solving the Laplacian smoothing problem outlined in Eq. (5). The one gradient descent step at $\mathbf{x}$ is:

$$\begin{aligned} \mathbf{f}^* &= \mathbf{x} - \eta \frac{\partial \mathcal{L}(\mathbf{h})}{\partial \mathbf{h}} | \mathbf{h} = \mathbf{x} \\ &= \mathbf{x} - \eta \lambda_1 L \mathbf{x} \\ &= \frac{\lambda_1}{(\lambda_1 + \lambda_2)} \hat{\tilde{\mathbf{A}}} \mathbf{x} + (\frac{\lambda_2}{(\lambda_1 + \lambda_2)} \mathbf{x} \ . \end{aligned} \tag{11}$$

where the step size $\eta = \frac{1}{2(\lambda_1 + \lambda_2)}$. The optimal solution $\mathbf{f}^*$ obtained through λ_1-scaled Laplacian smoothing is subsequently utilized to solve the second func-

tion, $\varphi(\mathbf{f}^*)$. Before delving into the search for the optimal solution of Cauchy smoothing, it is prudent to calculate the derivative of $\varphi(\mathbf{f}^*)$ with respect to $\mathbf{f}^*$.

$$\boldsymbol{z}^* = \frac{\partial \varphi(\mathbf{f}^*)}{\partial \mathbf{f}^*} = -2\lambda_2 \sum_i \sum_j \frac{w_{ij} \cdot \gamma(\mathbf{f}_i^* - \mathbf{f}_j^*)}{\left(\|\mathbf{f}_i^* - \mathbf{f}_j^*\|^2 + \gamma^2\right)^2} \ , \tag{12}$$

where w_{ij} has $d_{ij} = \|\mathbf{f}_i^* - \mathbf{f}_j^*\|^2$. Finally, let us take another gradient descent on $\mathbf{f}^*$ for Cauchy smoothing,

$$\begin{aligned} \mathbf{h}^* &= \mathbf{f}^* - \eta\epsilon \frac{\partial \varphi(\mathbf{f}^*)}{\partial \mathbf{f}^*} = \mathbf{f}^* - \eta\epsilon \boldsymbol{z}^* \\ &= \lambda \hat{\tilde{\mathbf{A}}} \mathbf{x} + (1-\lambda) \left(\mathbf{x} - \epsilon \sum_i \sum_j \frac{w_{ij} \cdot \gamma(\mathbf{f}_i^* - \mathbf{f}_j^*)}{\left(\|\mathbf{f}_i^* - \mathbf{f}_j^*\|^2 + \gamma^2\right)^2} \right) \ , \end{aligned} \tag{13}$$

where the step size $\eta = \frac{1}{2(\lambda_1+\lambda_2)}$, $\epsilon \in [0,1]$ is a scalar from the splitting method, and $\mathbf{h}^*$ denotes the output node representation at the current layer.

Let $\lambda = \frac{\lambda_1}{\lambda_1+\lambda_2}$, we introduce a more scalable parameter λ to enhance the equilibrium between the two smoothing strategies. For special cases, when $\lambda_2 = 0$ leads a fully Laplacian smoothing as $\lambda = 1$. Conversely, when $\lambda_1 = 0$ leads a complete Cauchy smoothing as $\lambda = 0$. Furthermore, it is important to note that all node representations in this context are normalized by the square root of the degree matrix ($\tilde{D}$). We denote the distance matrix of all neighboring nodes as $\Delta\mathbf{H}$, as such $\Delta\mathbf{H} = \sum_{A_{ij}\neq 0} \frac{\mathbf{H}_i}{\sqrt{\tilde{d}_i}} - \frac{\mathbf{H}_j}{\sqrt{\tilde{d}_j}}$. The graph convolution process in CauchyGCN follows the general framework of graph convolution outlined in Eq. (1), and a concise process is provided in Algorithm 1.

3.3 Clustering Analysis

We introduce a clustering analysis at the output layer to smooth both inter-class clustering and intra-class clustering, simultaneously enhancing the learning of local and global smoothness. Specifically, we employ the unsupervised deep clustering techniques in [24], which utilize the KL divergence to compare the clustering distribution (Q) with an auxiliary self-training target distribution (P). The KL divergence is calculated as follows:

$$KL(P\|Q) = \sum_{i=1}^{N} \sum_{c=1}^{C} p_{ic} \cdot log \frac{p_{ic}}{q_{ic}}, \tag{14}$$

where $c \in [1, C]$ denotes a class, with C being the number of classes in the dataset, and $i \in [1, |\mathcal{V}|]$ denotes a node.

The clustering distribution (Q) captures the similarity between the node representation $\mathbf{x}_i^{(L)}$ and cluster centroid $\{\mu_c\}_{c=1}^{C}$. Since the node representations are partially learned from the Cauchy distribution, we utilize the Cauchy distribution to compute the clustering distribution, enabling it to also capture heavy

Algorithm 1: Graph Convolution in CauchyGCN

Input: Node Representation $\mathbf{x}$; Adjacency Matrix $\hat{\tilde{\mathbf{A}}}$; Layer-specific trainable weight matrix Θ; Activation Function $\sigma(.)$; Hyper-parameters $\lambda \in [0,1]$, $\gamma \geq 0$, and $\epsilon \in [0,1]$

Output: Output Node Representation $\mathbf{H}$

Initialize $\mathbf{x}^{(0)} \leftarrow \mathbf{x}$, $k \leftarrow 1$, $\Theta^{(1)} \leftarrow \Theta$;

for $l \in [1, L]$ **do**

  Projection layer: $\mathbf{x}^{(l)} \leftarrow \mathbf{x}^{(l-1)}\Theta^{(l)}$

  if $\lambda \neq 0$ **then**

    Laplacian smoothing: $\mathbf{F}^{(l)} \leftarrow \lambda\hat{\tilde{\mathbf{A}}}\mathbf{x}^{(l)} + (1-\lambda)\mathbf{x}^{(l)}$

  else

    $\mathbf{F}^{(l)} \leftarrow \mathbf{x}^{(l)}$

  end

  if $\lambda \neq 1$ **then**

    $\Delta\mathbf{F}^{(l)} \leftarrow \sum_{A_{ij}\neq 0} \frac{\mathbf{F}_i^{(l)}}{\sqrt{\tilde{d}_i}} - \frac{\mathbf{F}_j^{(l)}}{\sqrt{\tilde{d}_j}}$

    $\boldsymbol{w}^{(l)} \leftarrow \hat{\tilde{\mathbf{A}}} \cdot e^{-\Delta\mathbf{F}^{(l)}/\sigma^2}$

    Cauchy smoothing: $\mathbf{Z}^{(l)} \leftarrow \frac{\gamma \cdot \boldsymbol{w}^{(l)} \Delta\mathbf{F}^{(l)}}{\left((\Delta\mathbf{F}^{(l)})^2 + \gamma^2\right)^2}$

  else

    $\mathbf{Z}^{(l)} \leftarrow 0$

  end

  Message-passing: $\mathbf{H}^{(l)} \leftarrow \lambda\hat{\tilde{\mathbf{A}}}\mathbf{x}^{(l)} + (1-\lambda)(\mathbf{x}^{(l)} - \epsilon\mathbf{Z}^{(l)})$

  Activation + Dropout layer: $\mathbf{H}^{(l)} \leftarrow dropout(\sigma(\mathbf{H}^{(l)}))$

  Update $l \leftarrow l + 1$

end

tails. The probability q_{ic} of assigning node i to cluster c is computed as follows:

$$q_{ic} = \frac{\left(1 + \|\mathbf{h}_i^{(L)} - \mu_c\|^2\right)^{-1}}{\sum c' \left(1 + \|\mathbf{h}_i^{(L)} - \mu_{c'}\|^2\right)^{-1}}, \tag{15}$$

where we assume the scale hyper-parameter $\gamma = 1$ for the Cauchy distribution. Recall, $\mathbf{h}^{(L)}$ is the node representation from the last layer.

The target distribution P is designed to enhance the performance of the classifier and correct centroids. The distribution P is defined as follows:

$$p_{ic} = \frac{q_{ic}^2 / f_c}{\sum_{c'} q_{ic'}^2 / f_{c'}} \tag{16}$$

where $f_c = \sum_i q_{ic}$ represents the soft cluster frequencies.

The challenge lies in determining the initial values for centroids μ and updating them throughout each training epoch in CauchyGCN. The initialization strategy of centroids, as outlined in [24], assumes a high accuracy of the classifier

during the initial training epoch. However, this assumption presents a challenge for CauchyGCN, given that the classifier's accuracy at the first epoch is typically quite low. Moreover, while the K-means clustering method seems intuitive, its lack of differentiability complicates the optimization problem during backpropagation. To circumvent the challenges, we compute the centroids μ_c by averaging the node representations associated with high confidence in each class label c. At the first training epoch, we initialize centroids μ_c as follows:

$$\mu_c = \frac{1}{N_c} \sum_{i=1}^{N_c} \mathbf{h}_{i \in c}^{(L)} \, . \tag{17}$$

In this approach, N_c denotes the number of nodes in cluster c. The condition $i \in c^L$ for node representation $\mathbf{h}^L$ indicates that node i is assigned to class c due to having the highest confidence in this class among all potential classes. The optimization of μ_c will be discussed in Section 3.4.

3.4 Optimization of CauchyGCN

The objective function of CauchyGCN in the output layer is jointly optimizing the performance of semi-supervised node classification and unsupervised clustering analysis,

$$\mathcal{O}_{CGCN} = \underbrace{- \sum_{l \in \mathcal{Y}_{label}} \sum_{f=1}^{F} Y_{lf} ln H_{lf}^{L}}_{\text{semi-supervised node classification}} + \underbrace{\kappa KL(P \| Q)}_{\text{clustering analysis}} \tag{18}$$

where $\mathcal{Y}_{label}$ is the set of ground truth labels of the labeled nodes and $\kappa \in [0, 1]$ is a hyper-parameter that control the effect of $KL(P \| Q)$. The gradient of $\mathcal{O}_{CGCN}$ with respect to $\mathbf{h}_i^{(L)}$ is computed as:

$$\frac{\partial \mathcal{O}_{CGCN}}{\partial \mathbf{h}_i^{(L)}} = 2 \sum_c (p_{ic} - q_{ic}) \frac{\mathbf{h}_i^{(L)} - \mu_c}{1 + \| \mathbf{h}_i^{(L)} - \mu_c \|^2} \, . \tag{19}$$

This is then passed down to the backpropagation to optimize the projection matrix Θ, i.e., $\partial \mathcal{O}_{CGCN} / \partial \Theta$. The gradient of $\mathcal{O}_{CGCN}$ with respect to cluster centroid μ_c is computed as:

$$\frac{\partial \mathcal{O}_{CGCN}}{\partial \mu_c} = -2 \sum_i (p_{ic} - q_{ic}) \frac{\mathbf{h}_i^{(L)} - \mu_c}{1 + \| \mathbf{h}_i^{(L)} - \mu_c \|^2} \, . \tag{20}$$

We manually update μ_c to the next training epoch as:

$$\mu_c' \leftarrow \mu_c - \beta \frac{\partial \mathcal{O}_{CGCN}}{\partial \mu_c} \, , \tag{21}$$

$\beta \in [0, 1]$ denotes the gradient step. This process is not involved in the backpropagation process that optimizes the projection matrix Θ.

4 Experiments and Results

4.1 Datasets

We validate our CauchyGCN on semi-supervised node classification tasks using the following real-world citation networks, Wikipedia-based article networks, and co-authorship networks: 1) **Cora**, **CiteSeer**, and **PubMed** [27] are citation networks widely used in node classification literature, where nodes are bag-of-words representations of documents and edges are citation links. 2) **Wiki-CS** [17] is a Wikipedia-based article network in the field of Computer Science, where nodes are word embeddings of the articles, edges are hyperlinks, and classes are assigned to 10 relative fields. 3) **Coauthor-CS** [21] is a co-authorship network based on the Microsoft Academic Graph, where nodes are paper keywords for each author's papers, edges depict co-authorship, and classes are assigned to the 15 most active fields. We conduct 10 runs on all datasets with the splitting method in [12] for training.

4.2 Settings and Baselines

To ensure fair comparisons, all methods are fixed under the following settings: the same data splitting method; 16 hidden feature size; 0.1 learning rate; 5e-4 weight decay rate; 0.5 dropout rate; 300 training epochs; and all performance is reported as the mean and standard variance of 10 runs in terms of semi-supervised node classification accuracy (%). The propagation depth, which refers to the number of graph convolutional layers, varies depending on the methods used. PPNP [2] and GNN-HF-closed [28] have a propagation depth of one. ChebNet [5], GCN [10], GAT [23], APPNP [7], GNN-HF-iter [28], ElasticGNN [12], and CauchyGCN have a propagation depth of two. All compared methods were implemented using the hyperparameters specified in the respective literature. In the case of CauchyGCN, the layer-wise message-passing scheme has the balancing parameter λ in the range of $[0.2, 0.7]$, the scale γ in Cauchy distribution is set to 1, the scalar of KL divergence in the optimization problem κ is chosen to be less than 0.1, and gradient step of centroids μ is selected from the range $0 < \beta \leq 0.5$.

4.3 Analysis of Classification Performance

The results of semi-supervised node classification are in Table 1. Here are some notable observations:

- *Importance of recognizing the closely embedded neighbors*: GAT demonstrates superior performance over GCN and ChebNet in Cora, CiteSeer, and Wiki-CS, suggesting the crucial importance of paying more attention to information aggregated and propagated from closely embedded neighbors. PPNP, APPNP, GNN-HF-closed, and GNN-HF-iter outperform GCN across most datasets, indicating that filtering out less frequently related neighboring information can effectively denoise the aggregated information at the center

node. These findings emphasize the significance of utilizing the underlying topology to assess the importance of neighbors, a concept incorporated into CauchyGCN through the modeling of the weight factor w_{ij}.
– *Importance of letting the data represent itself*: In contrast, ElasticGNN, the closest contender, also introduces an additional smoothing strategy to preserve local smoothness. However, CauchyGCN consistently outperforms ElasticGNN, particularly on Wiki-CS, a dataset characterized by high-frequency edges between nodes. While ElasticGNN relies on the l_1 norm and a soft-thresholding operator to preserve closely embedded neighbors, CauchyGCN distinguishes itself by allowing the data itself to determine the preservation of local smoothness through a Cauchy distribution.

It is worth noting that GNN-HF-closed outperforms CauchyGCN on Cora and CiteSeer by 0.3% and 0.1%, respectively. As discussed earlier, GNN-HF improves upon GCN from the spectral aspect by filtering information via a refined graph filter. On the other hand, CauchyGCN improves GCN by introducing a novel smoothing strategy from the spatial perspective. These two perspectives are independent and can be examined for potential integration to further improve the performance of GCNs.

Method	Cora	Citeseer	Pubmed	Wiki-CS	Coauthor-CS
ChebNet [5]	79.6 ± 1.7	69.1 ± 2.3	76.9 ± 1.9	68.2 ± 4.6	91.3 ± 0.4
GCN [10]	80.3 ± 1.6	69.7 ± 1.5	77.3 ± 1.8	74.6 ± 2.6	90.9 ± 0.6
PPNP [7]	81.5 ± 1.1	70.6 ± 1.5	78.6 ± 1.8	75.2 ± 2.2	88.9 ± 1.4
APPNP [7]	80.4 ± 1.8	70.0 ± 1.5	77.7 ± 2.1	75.2 ± 2.7	91.6 ± 0.5
GNN-HF-closed [28]	82.0 ± 1.2	**71.6 ± 1.2**	**79.3 ± 2.2**	70.7 ± 3.2	91.0 ± 0.5
GNN-HF-iter [28]	80.4 ± 1.5	70.7 ± 1.8	77.7 ± 2.3	72.4 ± 3.1	91.9 ± 0.5
GAT [23]	80.7 ± 1.7	70.6 ± 1.5	oom	74.7 ± 2.6	oom
ElasticGNN [12]	81.3 ± 1.5	70.8 ± 1.5	78.4 ±2.1	43.9 ± 13.1	91.5 ± 0.4
CauchyGCN	**82.2 ± 0.8**	71.3 ± 1.5	79.2 ± 1.7	**75.7 ± 2.7**	**92.0 ± 0.5**

Table 1. Semi-supervised node classification results (%). Bold is used to show the best results. *oom means out of memory.

4.4 Ablation Study

This section analyzes the performance of CauchyGCN with different configurations, this includes parameters: λ (the balancing factor in Eq. (9)) and ϵ (in Eq. (13)) in the message-passing scheme, as well as β (in Eq. (21)) and κ (in Eq. (18)) in clustering analysis. We use a heat map in Fig. 1 to illustrate the average classification accuracy of 10 runs across all possible configurations of λ_1 and λ_2, representing the balancing factor in the first and second graph convolutional layers, respectively. These results are conducted under a fixed value of ϵ, β, and κ, which are tuned for the best performance of CauchyGCN shown

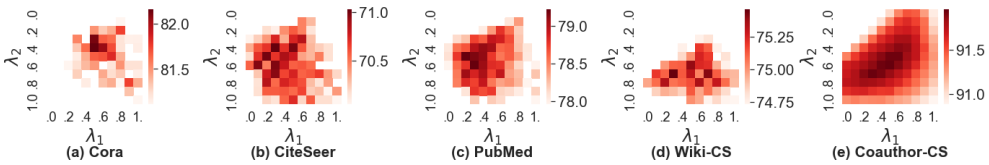

Fig. 1. CauchyGCN ablation study on balancing parameters λ in Eq. (9). Specifically, λ_1 and λ_2 are the balancing factors in the first and second graph convolutional layers, respectively. All colored cubes denote performance equal to or higher than that of the fully Laplacian smoothing (GCN) method, where $[\lambda_1, \lambda_2] = [1, 1]$, with darker colors indicating superior performance.

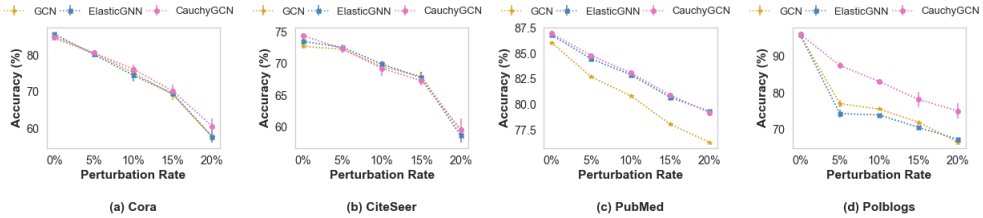

Fig. 2. Sensitivity analysis under adversarial graph attack.

in Table 1. Observing Fig. 1: 1) The value of $(1 - \lambda)$ represents the proportion of Cauchy smoothing employed in CauchyGCN, whereas λ denotes the percentage of Laplacian smoothing. Generally, the presence of more colored cubes compared to uncolored ones suggests that CauchyGCN is readily adjustable for superior performance over GCN. Notably, the best performance of CauchyGCN emerges when $[\lambda_1, \lambda_2]$ being $[0.5, 0.4]$ on Cora, $[0.3, 0.4]$ on CiteSeer, $[0.3, 0.6]$ on PubMed, $[0.3, 0.7]$ on Wiki-CS, and $[0.3, 0.7]$ on Coauthor-CS, illustrating a significant reliance and need for Cauchy smoothing in preserving local smoothness. 2) All experiments have a Cauchy-based clustering analysis since the tuned value of β and κ are greater than 0. GCN with clustering analysis, the special case wherein $[\lambda_1, \lambda_2] = [1, 1]$, underperformed CauchyGCN but outperformed or equal to GCN without clustering analysis. This observation underscores the effectiveness of both the Cauchy-based message-passing scheme and the clustering analysis in CauchyGCN.

4.5 Robustness under Graph Attack

We assess the robustness of CauchyGCN under adversarial attacks [29] on graph structure with varying perturbation ratios. The experimental results, as illustrated in Fig. 2, encompass different methods evaluated under perturbation rates of 0%/5%/10%/15%/20%. Note that the outcomes for the 0% perturbation rate are not directly comparable to those in Table 1 due to differing data splitting techniques. For this study, nodes are randomly split as 10%/10%/80% for

training, validation, and testing [12]. The findings from Fig. 2 indicate the superior performance of CauchyGCN over GCN and ElasticGNN, attributed to the Cauchy smoothing. It efficiently identifies proximal neighbors and preserves local smoothness, even when confronted with noisy and irregular neighboring information.

5 Conclusion

This paper introduces CauchyGCN with a Cauchy-smoothing message-passing scheme and Cauchy-based unsupervised clustering analysis. The scheme leverages the underlying topology to explain and preserve smoothness within closely embedded and mutually connected neighborhoods, alleviating strict penalization of distant neighbors seen in Laplacian smoothing. The Cauchy-based unsupervised clustering analysis enhances intra-class smoothness in the output layer, thereby simultaneously improving the classifier's ability to learn both local and global smoothness. Extensive experiments on node classifications showcase the effectiveness of CauchyGCN, highlighting the importance of preserving local smoothness.

6 Acknowledgments

The authors would like to thank anonymous reviewers for their insightful comments. The work performed at the Temple University is partially supported by U.S. National Science Foundation under Grant No. OAC-2311758 and CCF-2134203.

References

1. Belkin, M., Niyogi, P.: Laplacian eigenmaps for dimensionality reduction and data representation. Neural Computation **15**(6), 1373–1396 (2003)
2. Cao, Y., Long, M., Liu, B., Wang, J.: Deep cauchy hashing for hamming space retrieval. In: IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 1229–1237 (2018)
3. Chen, D., Lin, Y., Li, W., Li, P., Zhou, J., Sun, X.: Measuring and relieving the over-smoothing problem for graph neural networks from the topological view. Proceedings of the Conference on Artificial Intelligence **34**(04), 3438–3445 (2020)
4. Chung, F.R.K.: Spectral graph theory. Providence, RI: American Mathematical Society (1997)
5. Defferrard, M., Bresson, X., Vandergheynst, P.: Convolutional neural networks on graphs with fast localized spectral filtering. In: Proceedings of the 30th International Conference on Neural Information Processing Systems. p. 3844–3852 (2016)
6. Gao, C., Zheng, Y., Li, N., Li, Y., Qin, Y., Piao, J., Quan, Y., Chang, J., Jin, D., He, X., Li, Y.: A survey of graph neural networks for recommender systems: Challenges, methods, and directions. ACM Trans. Recomm. Syst. **1**(1) (2023)

7. Gasteiger, J., Bojchevski, A., Günnemann, S.: Predict then propagate: Graph neural networks meet personalized pagerank. In: International Conference on Learning Representations (2019)
8. Giles, C.L., Bollacker, K.D., Lawrence, S.: Citeseer: An automatic citation indexing system. In: Proceedings of the Third ACM Conference on Digital Libraries. p. 89–98. DL '98 (1998). https://doi.org/10.1145/276675.276685
9. Hamilton, W., Ying, Z., Leskovec, J.: Inductive representation learning on large graphs. In: Advances in Neural Information Processing Systems. vol. 30 (2017)
10. Kipf, T.N., Welling, M.: Semi-supervised classification with graph convolutional networks. In: International Conference on Learning Representations (2017)
11. Li, Q., Han, Z., Wu, X.M.: Deeper insights into graph convolutional networks for semi-supervised learning. In: Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence. AAAI Press (2018)
12. Liu, X., Jin, W., Ma, Y., Li, Y., Hua, L., Wang, Y., Yan, M., Tang, J.: Elastic graph neural networks. In: Proceedings of the 38th International Conference on Machine Learning. PMLR (2021)
13. Luo, D., Ding, C., Nie, F., Huang, H.: Cauchy graph embedding. In: Proceedings of the 28th International Conference on International Conference on Machine Learning. p. 553–560. ICML'11, Omnipress, Madison, WI, USA (2011)
14. Ma, Y., Liu, X., Shah, N., Tang, J.: Is homophily a necessity for graph neural networks? In: International Conference on Learning Representations (2022)
15. Ma, Y., Liu, X., Zhao, T., Liu, Y., Tang, J., Shah, N.: A unified view on graph neural networks as graph signal denoising. In: Proceedings of the 30th ACM International Conference on Information & Knowledge Management (2021)
16. McCallum, A.K., Nigam, K., Rennie, J., Seymore, K.: Automating the construction of internet portals with machine learning. Information Retrieval **3**, 127–163 (2000)
17. Mernyei, P., Cangea, C.: Wiki-cs: A wikipedia-based benchmark for graph neural networks. CoRR (2020), https://arxiv.org/abs/2007.02901
18. Page, L., Brin, S., Motwani, R., Winograd, T.: The pagerank citation ranking : Bringing order to the web. In: The Web Conference (1999)
19. Parikh, N., Boyd, S.: Proximal algorithms. Foundations and Trends in Optimization **1**(3), 127–239 (2014). https://doi.org/10.1561/2400000003
20. Scarselli, F., Gori, M., Tsoi, A.C., Hagenbuchner, M., Monfardini, G.: The graph neural network model. IEEE Transactions on Neural Networks **20**(1), 61–80 (2009)
21. Shchur, O., Mumme, M., Bojchevski, A., Günnemann, S.: Pitfalls of graph neural network evaluation. ArXiv (2019), https://arxiv.org/abs/1811.05868
22. Sun, M., Zhang, X., Zheng, J., Ma, G.: Ddgcn: Dual dynamic graph convolutional networks for rumor detection on social media. Proceedings of the AAAI Conference on Artificial Intelligence **36**(4), 4611–4619 (2022)
23. Veličković, P., Cucurull, G., Casanova, A., Romero, A., Liò, P., Bengio, Y.: Graph attention networks. In: International Conference on Learning Representations (2018)
24. Xie, J., Girshick, R., Farhadi, A.: Unsupervised deep embedding for clustering analysis. In: Proceedings of The 33rd International Conference on Machine Learning. vol. 48, pp. 478–487 (2016)
25. Xu, K., Li, C., Tian, Y., Sonobe, T., Kawarabayashi, K.i., Jegelka, S.: Representation learning on graphs with jumping knowledge networks. In: Proceedings of the 35th International Conference on Machine Learning. vol. 80, pp. 5453–5462 (2018)
26. Yang, R., Dai, W., Li, C., Zou, J., Xiong, H.: Ncgnn: Node-level capsule graph neural network for semisupervised classification. IEEE Transactions on Neural Networks and Learning Systems pp. 1–15 (2022)

27. Yang, Z., Cohen, W.W., Salakhutdinov, R.: Revisiting semi-supervised learning with graph embeddings. In: Proceedings of the 33rd International Conference on International Conference on Machine Learning - Volume 48. p. 40–48 (2016)
28. Zhu, M., Wang, X., Shi, C., Ji, H., Cui, P.: Interpreting and unifying graph neural networks with an optimization framework. In: Proceedings of the Web Conference 2021. p. 1215–1226 (2021)
29. Zügner, D., Günnemann, S.: Adversarial attacks on graph neural networks via meta learning. In: International Conference on Learning Representations (2019)