Connecting Graph Convolutional Networks and Graph-Regularized PCA

Lingxiao Zhao [1] **Leman Akoglu** [1]

Abstract

Graph convolution operator of the GCN model is originally motivated from a localized first-order approximation of spectral graph convolutions. This work stands on a different view; establishing a *mathematical connection between graph convolution and graph-regularized PCA* (GPCA). Based on this connection, GCN architecture, shaped by stacking graph convolution layers, shares a close relationship with stacking GPCA. We empirically demonstrate that the *unsupervised* embeddings by GPCA paired with a 1- or 2-layer MLP achieves similar or even better performance than GCN on semi-supervised node classification tasks across five datasets including Open Graph Benchmark [1]. This suggests that the prowess of GCN is driven by graph based regularization. In addition, we extend GPCA to the (semi-)supervised setting and show that it is equivalent to GPCA on a graph extended with "ghost" edges between nodes of the same label. Finally, we capitalize on the discovered relationship to design an effective initialization strategy based on stacking GPCA, enabling GCN to converge faster and achieve robust performance at large number of layers. Notably, the proposed initialization is general-purpose and applies to other GNNs.

1. Introduction

Graph neural networks (GNNs) are neural networks designed for the graph domain. Since the breakthrough of GCN (Kipf & Welling, 2017), which notably improved performance on the semi-supervised node classification problem, many GNN variants have been proposed; including GAT (Veličković et al., 2018), GraphSAGE (Hamilton et al., 2017), DGI (Veličković et al., 2019), GIN (Xu et al., 2019), APPNP (Klicpera et al., 2019), to name a few.

Despite the empirical successes of GNNs in both node-level and graph-level tasks, they remain not well understood due to the lack of systematic and theoretical analysis of GNNs. For example, researchers have found that GNNs, unlike their non-graph counterparts, suffer from performance degradation with increasing depth, losing their expressive power exponentially in number of layers (Oono & Suzuki, 2020). Such behavior is only partially explained by the oversmoothing phenomenon (Li et al., 2018; Zhao & Akoglu, 2020). Another surprising observation shows that a Simplified Graph Convolution model, named SGC (Wu et al., 2019), can achieve similar performance to various more complex GNNs on a variety of node classification tasks. Moreover, a simple baseline that does not utilize the graph structure altogether performs similar to state-of-the-art GNNs on graph classification tasks (Errica et al., 2020). These observations call attention to studies for a better understanding of GNNs (NT & Maehara, 2019; Morris et al., 2019; Xu et al., 2019; Oono & Suzuki, 2020; Loukas, 2020a; Srinivasan & Ribeiro, 2020). (See Sec. 2 for more on understanding GNNs.)

Toward a systematic analysis and better understanding of GNNs, we establish a connection between the graph convolution operator of GCN and Graph-regularized PCA (GPCA) (Zhang & Zhao, 2012), and show the similarity between GCN and stacking GPCA. This connection provides a deeper understanding of GCN's power and limitation. Empirically, we also find that GPCA performance matches that of GCN on benchmark semi-supervised node classification tasks. What is more, the unsupervised stacking GPCA can be viewed as "unsupervised GCN" and provides a straightforward, yet systematic way to initialize GCN training. We summarize our contributions as follows:

- **Connection btwn. Graph Convolution and GPCA:** We establish the connection between the graph convolution operator of GCN and the closed-form solution of graph-regularized PCA formulation. We demonstrate that a simple graph-regularized PCA paired with 1- or 2-layer MLP can achieve similar or even better results than GCN over several benchmark datasets. We further extend GPCA to (semi-)supervised setting which can generate embeddings using information of labels, which yields better performance on 3 out of 5 datasets. The outstanding performance of simple GPCA supports that the prowess of GCN on node classification task comes from graph based regularization. This motivates the study and design of other graph regularizations in the future.

[1] https://ogb.stanford.edu/

- GPCANET**: New Stacking GPCA model:** Capitalizing on the connection between GPCA and graph convolution, we design a new GNN model called GPCANET shaped by (1) stacking multiple GPCA layers and nonlinear transformations, and (2) fine-tuning end-to-end via supervised training. GPCANET is a generalized GCN model with adjustable hyperparameters that control the strength of graph regularization of each layer. We show that with stronger regularization, we can train GPCANET with fewer (1–3) layers and achieve comparable performance to much deeper GCNs.
- **First initialization strategy for GNNs:** Capitalizing on the connection between GCN and GPCANET, we design a new strategy to initialize GCN training based on stacking GPCA, outperforming the popular Xaiver initialization (Glorot & Bengio, 2010). We show that the GPCANET-initialization is extremely effective for training deeper GCNs, with greatly improved convergence speed, performance, and robustness. Notably, GPCANET-initialization is general-purpose and also applies to other GNNs. To our knowledge, it is the first initialization method specifically designed for GNNs.

Reproducibility. We open-source all the code developed in this work at `http://bit.ly/GPCANet`. The datasets are already public-domain.

2. Related Work

Understanding GNNs. We discuss related work that provide a more systematic understanding of GNNs in a number of fronts. GCN's graph convolution is originally motivated from the approximation of graph filters in graph signal processing (Kipf & Welling, 2017). NT et al. (2019) show that graph convolution only performs low-pass filtering on original feature vectors, and also states a connection between graph filter and Laplacian regularized least squares. Motivated by the oversmoothing phenomenon of graph convolution, Oono et al. (2020) theoretically prove that GCN can only preserve information of node degrees and connected components when the number of layers goes to infinity, under some conditions of GCN weights. At graph-level, Xu et al. (2019) show that GNNs cannot have better expressive power than the Weisfeiler-Lehman (WL) test of graph isomorphism, and develop the GIN model that is as powerful as the WL test. Morris et al. (2019) extend the work by (Xu et al., 2019) and establish a connection to the higher-order WL algorithm. When given distinguishable node features, Loukas (2020a) has shown that the GNN models can be Turing universal with sufficient depth and width. The relationship between network capacity and network size is also studied in (Loukas, 2020b). In terms of limitation of expressiveness, Chen et al. (2020) studied the attributed graph substructures problem and proved that GNNs cannot count induced subgraphs. Garg et al. (2020) further showed that message-passing based GNNs cannot compute graph properties such as cycle, diameter and clique informations, also providing a Rademacher generalization bound for binary graph classification. Recently, Liao et al. (2021) improved the generalization bound via a PAC-Bayesian approach.

Graph-regularized PCA. PCA and its variants are standard linear dimensionality reduction approaches. Several works extend PCA to graph-structured data, such as Graph-Laplacian PCA (Jiang et al., 2013) and Manifold-regularized Matrix Factorization (Zhang & Zhao, 2012). For other variants, see (Shahid et al., 2016).

Stacking Models and Deep Learning. The connection between CNN and stacking PCA has been explored in PCANet (Chan et al., 2015), which demonstrated that the (unsupervised) simple stacking PCA works as good as supervised CNN over a large variety of vision tasks. The original PCANet is shallow and does not have nonlinear transformations, PCANet+ (Low et al., 2017) overcomes these limitations and pushes the architecture much deeper. The idea of layerwise stacking for feature extraction is not new and was empirically observed to exhibit better representation ability in terms of classification. For a comprehensive review, we refer to (Bengio et al., 2013).

Initialization. Traditionally, neural networks (NNs) were initialized with random weights generated from Gaussian distribution with zero mean and a small standard deviation (Krizhevsky et al., 2012). As training deeper NNs became extremely difficult due to vanishing gradient and activation functions, Glorot et al. (2010) provided a specific weight initialization formula, named Xavier initialization, based on variance analysis without considering activation function. Xavier initialization is widely used for any type of NN even today, and it is the main initialization strategy used for GNNs. Later, He et al. (2015) adapted Xavier initialization to ReLU activation by considering a multiplier. Taking another direction, Saxe et al. (2013) analyzed the dynamics of training deep NNs and proposed random orthonormal initialization. Mishkin and Matas (2015) further improved orthonormal initialization for batch normalization (Ioffe & Szegedy, 2015). Different from these data-independent approaches, others (Wagner et al., 2013; Krähenbühl et al., 2016; Seuret et al., 2017) have employed data-dependent techniques, like PCA, to initialize deep NNs. Although initialization has been widely studied for general NNs, no specific initialization has been proposed for GNNs. In this work, we propose a data-driven initialization technique (based on GPCA), specific to GNNs for the first time.

3. GPCA and Graph Convolution

3.1. Graph Convolution

Consider a node-attributed input graph $G = (V, E, X)$ with $|V| = n$ nodes and $|E| = m$ edges, where $X \in \mathbb{R}^{n\times d}$ denotes the feature matrix with d features. Similar to other neural networks stacked with repeated layers, GCN (Kipf & Welling, 2017) contains multiple graph convolution layers each of which is followed by a nonlinear activation. Let $H^{(l)}$ be the l-th hidden layer representation, then GCN follows:

$$H^{(l+1)} = \sigma(\tilde{A}_{\text{sym}} H^{(l)} W^{(l)}) \tag{1}$$

where $\tilde{A}_{\text{sym}} = \tilde{D}^{-\frac{1}{2}}(A + I)\tilde{D}^{-\frac{1}{2}}$ denotes the $n \times n$ symmetrically normalized adjacency matrix with self-loops, $\tilde{D}$ is the diagonal degree matrix where $\tilde{D}_{ii} = 1 + \sum_{j=1}^{n} A_{ij}$, $W^{(l)}$ is the l-th layer parameter (to be learned), and σ is the nonlinear activation function.

The graph convolution operation is defined as the formulation before activation in Eq. (1). Formally, graph convolution is parameterized with W and maps an input X to a new representation Z as

$$Z = \tilde{A}_{\text{sym}} X W\ . \tag{2}$$

3.2. Graph-regularized PCA (GPCA)

As stated by (Bengio et al., 2013), “Although depth is an important part of the story, many *other priors* are interesting and can be conveniently captured when the problem is cast as one of learning a representation.” GPCA is one such representation learning technique with a graph-based prior.

Standard PCA learns k-dimensional projections $Z \in \mathbb{R}^{n\times k}$ of feature matrix $X \in \mathbb{R}^{n\times d}$, aiming to minimize the reconstruction error

$$\|X - ZW^T\|_F^2\ , \tag{3}$$

subject to $W \in \mathbb{R}^{d\times k}$ being an orthonormal basis. GPCA extends this formalism to graph-structured data by additionally assuming either smoothing bases (Jiang et al., 2013) or smoothing projections (Zhang & Zhao, 2012) over the graph. In this work we consider the latter case where low-dimensional projections are smooth over the input graph G, with its normalized Laplacian matrix $\tilde{L} = I - \tilde{A}_{\text{sym}}$. The objective formulation of GPCA is then given as

$$\min_{Z,W} \quad \|X - ZW^T\|_F^2 + \alpha \operatorname{Tr}(Z^T \tilde{L} Z) \tag{4}$$

$$\text{s.t.} \quad W^T W = I \tag{5}$$

where α is a hyperparameter that balances reconstruction error and the variation of the projections over the graph. Note that the first part of Eq. (4), along with the constraint, corresponds to the objective of the original PCA, while the second part is a graph regularization term that aims to “smooth” the new representations Z over the graph structure. As such, GPCA becomes the standard PCA when $\alpha = 0$.

Similar to PCA, the problem (4-5) is non-convex but has a closed-form solution (Zhang & Zhao, 2012). Surprisingly, as we show, it has a close connection with the graph convolution formulation in Eq. (2). In the following, we give the GPCA solution and then detail its connection to graph convolution in the next subsection.

Theorem 3.1. *GPCA with formulation shown in (4-5) has the optimal solution (Z^*, W^*) following*

$$W^* = (\mathbf{w}_1, \mathbf{w}_2, ..., \mathbf{w}_k)$$
$$Z^* = (I + \alpha\tilde{L})^{-1} X W^*$$

where $\mathbf{w}_1, \mathbf{w}_2, ..., \mathbf{w}_k$ are the eigenvectors corresponding to the largest k eigenvalues of the matrix $X^T(I + \alpha\tilde{L})^{-1}X$.

Proof. We give the proof in two steps.

Step 1: For a fixed W, Solve optimal Z^* as a function of W: When fixing W as constant, the problem becomes quadratic and convex. There is a unique solution, given by first-order optimal condition. Let ℓ denote the objective function as given in (4). Its gradient can be calculated as

$$\frac{\partial \ell}{\partial Z} = 2(I + \alpha\tilde{L})Z - 2XW\ . \tag{6}$$

Setting (6) to 0 leads to the solution $Z^* = (I + \alpha\tilde{L})^{-1}XW$.

Step 2: Replace Z with Z^*, Solve optimal W^*: Substituting Z in objective ℓ with $Z^* = (I + \alpha\tilde{L})^{-1}XW$, we reduce the optimization to

$$\min_{W, W^TW=I} \quad \|X - (I + \alpha\tilde{L})^{-1}XWW^T\|_F^2 + \alpha \operatorname{Tr}\left[W^T X^T (I + \alpha\tilde{L})^{-1}\tilde{L}(I + \alpha\tilde{L})^{-1}XW\right]\ . \tag{7}$$

For this part only, let $M = (I + \alpha\tilde{L})^{-1}$ to simplify the notation. We can show that (7) is equivalent to

$$\min_{W, W^TW=I} \quad \operatorname{Tr}(XX^T + MXWW^TWW^TX^TM) - 2\operatorname{Tr}(MXWW^TX^T) + \alpha \operatorname{Tr}(W^TX^TM\tilde{L}MXW) \tag{8}$$

Using the cyclic property of (Tr)ace (and plugging $(I + \alpha\tilde{L})^{-1}$ for M back), we can write it as (see Supp. A.1 for detailed derivation.)

$$\max_{W, W^TW=I} \quad \operatorname{Tr}\left[W^T X^T (I + \alpha\tilde{L})^{-1} XW\right]\ . \tag{9}$$

Based on the spectral theorem of PSD matrices, the optimal solution W^* of problem (9) is the combination of eigenvectors, associated with the largest c eigenvalues of the graph-revised covariance matrix $X^T(I + \alpha\tilde{L})^{-1}X$. □

3.3. Connection btwn. Graph Convolution and GPCA

Let $\Phi_\alpha := I + \alpha\tilde{L}$. The normalized Laplacian matrix $\tilde{L}$ has absolute eigenvalues bounded by 1, thus, all its positive powers have bounded operator norm. When $\alpha \leq 1$, Φ_α^{-1} can be decomposed into Taylor series as

$$(I + \alpha\tilde{L})^{-1} = I - \alpha\tilde{L} + ... + (-\alpha)^t\tilde{L}^t + ... \tag{10}$$

The first-order truncated form (i.e. approx.) of Eq. (10) is

$$(I + \alpha\tilde{L})^{-1} \approx I - \alpha\tilde{L} = (1-\alpha)I + \alpha\tilde{A}_{\text{sym}} \ . \tag{11}$$

When $\alpha = 1$, the first-order approximation of Z^* follows

$$Z^* \approx \tilde{A}_{\text{sym}}XW^* \ . \tag{12}$$

The (approximate) solution to GPCA in Eq. (12) matches the graph convolution operation in Eq. (2), with W^* plugged in as the eigenvectors of the matrix $X^T\Phi_\alpha^{-1}X$.

We restate the key contribution of this paper: *The graph convolution operation can be viewed as the first-order approximation of GPCA with $\alpha = 1$ with a learnable W. Put differently, the first-order approximation of (unsupervised) GPCA with $\alpha = 1$ can be viewed as a graph convolution with a fixed, data-driven W.* Notably, for $\alpha < 1$, Eq. (11) shows the connection between GPCA and graph convolution equipped with 1-step (scaled) residual connection.

3.4. Supervised GPCA

The standard GPCA problem as given in (4-5) is unsupervised. In this section, we show how to extend it to the supervised setting. Here, besides (1) providing good reconstruction and (2) varying smoothly over the graph structure, we would want to learn embeddings that also (3) highly correlate with the response, or outcome variable(s). For simplicity of presentation, let $\mathbf{z} \in \mathbb{R}^d$ be a 1-d embedding and Y denote the response matrix (considering the general case of multiple responses). We write the additional objective as

$$\max_{\mathbf{z}} \quad \left[\text{corr}(Y, \mathbf{z})\right]^T \left[\text{corr}(Y, \mathbf{z})\right] \ \text{var}(\mathbf{z}) \tag{13}$$

which aims to maximize the correlation between $\mathbf{z}$ and Y.[2] The form of (13) and the variance-maximizing term var($\mathbf{z}$) are for mathematical convenience, that will become explicit in the following. Despite agnostic to labels, including var($\mathbf{z}$) is intuitive since an implicit objective of data projection (embedding) is to ensure that inherent variation (structure) in data is captured as much as possible. (Recall from vanilla PCA that the objective of minimizing reconstruction error in (3) is equivalent to maximizing the variance of data in the projected space.)

[2]Note that for the optimization to be well-posed, constraints on $\mathbf{z}$ would be required which we omit for simplicity of presentation.

We can write (13) in solely matrix form, through a series of transformations (See Supp. A.2), as

$$\begin{aligned} &\max_{\mathbf{z}} \quad \left[\text{corr}(Y, \mathbf{z})\right]^T \left[\text{corr}(Y, \mathbf{z})\right] \text{var}(\mathbf{z}) \\ \equiv &\max_{\mathbf{z}} \quad \mathbf{z}^T YY^T \mathbf{z} \end{aligned} \tag{14}$$

In the general case, we would aim to maximize the trace of Z^TYY^TZ for multi-dimensional embeddings.

Interpretation. For semi-supervised node classification with c classes, let $\mathcal{L} \subset V$ denote the set of labeled nodes. For this task, $Y \in \{0, 1\}^{n \times c}$ would encode the node labels where the v-th row of Y, denoted Y_v, depicts the one-hot encoded label for each $v \in \mathcal{L}$. For $u \in V \backslash \mathcal{L}$ with unknown labels, $Y_u = \mathbf{0}$, set as the c-dimensional all-zero vector.

Then, $(YY^T)_{ij}$ is simply equal to 1 when nodes i and j share the same label, and otherwise 0 (either because they have different labels or labels are unknown, i.e. Y_i and/or Y_j are all 0's). This term simply enforces the representations Z_i and Z_j of two same-labeled nodes to be similar. In a sense, YY^T is adding "ghost" edges between the same-label nodes, further guiding the smoothness of their representations over this extended graph structure. This is particularly beneficial when graph smoothing may not be enough to ensure two nodes of the same label have similar embeddings when they are not directly connected or are far away in the graph.

We remark that earlier work (Gallagher et al., 2008) have heuristically introduced edges between same-label nodes to enhance a given graph for the node classification task. In this work, we have derived the theoretical underpinning for this strategy.

Supervised formulation. We have shown that requiring the embeddings to correlate with known labels can be interpreted as additional smoothing over "ghost" edges between the same-label nodes in the graph. As such, we extend the GPCA problem (4-5) to the (semi-)supervised setting as

$$\min_{Z,W} \quad \|X - ZW^T\|_F^2 + \alpha\,\text{Tr}(Z^T\tilde{L}_{\text{spr}}Z) \tag{15}$$

$$\text{s.t.} \quad W^TW = I \quad ; \tag{16}$$

$$\text{where } \tilde{L}_{\text{spr}} = I - \tilde{A}_{\text{spr}}$$

$$\tilde{A}_{\text{spr}} = (1-\beta)\tilde{A}_{\text{sym}} + \beta D^{-\frac{1}{2}}(YY^T)D^{-\frac{1}{2}} \tag{17}$$

In Eq. (17), β is an additional hyperparameter for trading-off the graph-based regularization (i.e. smoothing) due to the actual input graph edges versus the ones introduced through YY^T between the nodes of the same label, and D is the diagonal matrix with $D_{ii} = \sum_{j=1}^n (YY^T)_{ij}$.

Theorem 3.2. *Supervised GPCA with formulation shown in (15-16) has the optimal solution (Z^*, W^*) following*

$$W^* = (\mathbf{w}_1, \mathbf{w}_2, ..., \mathbf{w}_k) \tag{18}$$

$$Z^* = (I + \alpha\tilde{L}_{\text{spr}})^{-1}XW^* \tag{19}$$

where $\mathbf{w}_1, \ldots, \mathbf{w}_k$ *are the top eigenvectors of matrix* $X^T(I + \alpha\tilde{L}_{\text{spr}})^{-1}X$*, equivalently* $X^T\big((1+\alpha)I - \big[\alpha(1-\beta)\tilde{A}_{sym} + \alpha\beta D^{-\frac{1}{2}}YY^TD^{-\frac{1}{2}}\big]\big)^{-1}X$*, corresponding to the largest* k *eigenvalues.*

Proof. The proof is similar to that of Theorem 3.1. □

The first-order approximation of the matrix inverse in Eq. (19) can be written as $I - \alpha\tilde{L}_{\text{spr}} = (1-\alpha)I + \alpha\tilde{A}_{\text{spr}}$. As such, when $\alpha = 1$,

$$Z^* \approx \big[(1-\beta)\tilde{A}_{\text{sym}} + \beta D^{-\frac{1}{2}}(YY^T)D^{-\frac{1}{2}}\big]XW^* \,. \quad (20)$$

Similar to Eq. (12), the first-order approximation of the supervised GPCA with $\alpha = 1$ can still be viewed as a graph convolution, this time on a graph enhanced with what-we-called “ghost” edges.

3.5. Approximation and Complexity Analysis

According to formulations in Theorems 3.1 and 3.2, obtaining $W^*{\in}\mathbb{R}^{d\times k}$ and $Z^*{\in}\mathbb{R}^{n\times k}$ requires two demanding computations (1) the inverse of $\Phi_\alpha = (I + \alpha\tilde{L}) \in \mathbb{R}^{n\times n}$, or in the supervised case $\Phi_\alpha = (I + \alpha\tilde{L}_{\text{spr}})$; and (2) top k eigenvectors of the matrix $X^T\Phi_\alpha^{-1}X \in \mathbb{R}^{d\times d}$. Eigen-decomposition takes $O(d^3)$ (Pan & Chen, 1999), which is scalable as d is usually small. Computing matrix inverse, on the other hand, can take $O(n^3)$ and require $O(n^2)$ memory, which would be infeasible for very large graphs.

To reduce computation and memory complexity, we instead approximately compute $F := \phi_\alpha^{-1}X$, which is what we really need for both W^* and Z^*. We can equivalently write

$$\begin{aligned}(I + \alpha\tilde{L})F = X &\implies F + \alpha F = \alpha PF + X \\ &\implies F = \frac{\alpha}{1+\alpha}PF + \frac{1}{1+\alpha}X\end{aligned}$$

where $P = \tilde{A}_{\text{sym}}$ in the unsupervised case and $P = (1-\beta)\tilde{A}_{\text{sym}} + \beta D^{-\frac{1}{2}}(YY^T)D^{-\frac{1}{2}}$ when supervised.

Then, we can iteratively (with total T iterations) use the power method (Golub & Van Loan, 1989) to compute F as

$$F^{(t+1)} \leftarrow \frac{\alpha}{1+\alpha}PF^{(t)} + \frac{1}{1+\alpha}X \quad (21)$$

where $t \in \{0, ..., T\}$ depicts the iteration and $F^{(0)} \in \mathbb{R}^{n\times d}$ is initialized as X (or randomly). For the supervised case, $PF^{(t)}$ is computed through a series of (from right to left) matrix-matrix products. This avoids the explicit construction of matrix YY^T in memory. Overall, solving for F takes $O(T(m{+}n)d)$ where m is the number of edges in the graph. The supervised case has an additional term $O(Td|\mathcal{L}|c)$ with c being the number of classes and $|\mathcal{L}| \leq n$ be the number of labeled nodes, which can also be upper-bounded by $O(T(m+n)d)$ when treating c as constant.

Having solved for F, we perform the matrix-matrix product $Z^* = FW^*$ in $O(ndk)$ and then the eigen-decomposition of X^TF in $O(d^3 + nd^2) = O(nd^2)$ $(n \geq d)$. Assuming $O(d) = O(k)$, overall complexity for computing single layer GPCA is given as $O(Tmd{+}Tnd{+}nd^2)$, which is *linear in the number of nodes and edges*. Note that empirically we found $5 \leq T \leq 10$ to be sufficient.

4. GPCANET: A Stacking GPCA Model

4.1. GPCANET

Thus far, we drew a connection between the geometrically motivated manifold-learning based GPCA and the graph convolution operation of deep neural network based GCN. Next we capitalize on this connection to design a new model called GPCANET that takes advantage of the relative strengths of each paradigm; namely, GPCA’s ability to capture data variation and structure (i.e. data manifold), and GCN’s ability to capture multiple levels of abstraction (high-level concepts) through stacked layers and non-linearity.

In a nustshell, GPCANET is a stacking of multiple (unsupervised or supervised) GPCA layers and nonlinear transformations, which shares the same architecture as a multi-layer GCN. It consists of two main stages: (1) *Pre-training*, which sets the layer-wise parameters through closed-form GPCA solutions, and (2) *End-to-end-training*, which refines the parameters through end-to-end gradient-based minimization of a global supervised loss criterion at the output layer.

We remark that GPCANET is *not* the same as GCN, as each convolution layer uses the formulation in Theorems 3.1 and 3.2 (with approximation shown in Sec. 3.5). In fact, when $\alpha = 1$ and $\beta = 0$, GPCANET is the GCN model initialized with GPCANET-initialization, which we discuss more in Sec. 4.2. In other words, GPCANET is a *generalized* GCN model with additional hyperparameters, α and β, controlling the strength of graph regularization based on the existing or “ghost” edges, respectively.

4.1.1. FORWARD PASS AND PRE-TRAINING STAGE

During pre-training, weights of the l-th layer, denoted as $W^{(l)} \in \mathbb{R}^{d_{l-1}\times d_l}$, are pre-set (i.e. initialized) as the leading d_l eigenvectors of the matrix ${H^{(l-1)}}^T\Phi_\alpha^{-1}H^{(l-1)}$,[3] where $H^{(l-1)}$ is the representation as output by the $(l-1)$-th layer (with $H^{(0)} := X$), and Φ_α can be the unsupervised $(I{+}\alpha\tilde{L})$ or the supervised $(I + \alpha\tilde{L}_{\text{spr}})$.

The pre-training stage takes a single forward pass. Alg. 1 shows both the forward pass of GPCANET used during end-to-end-training stage and the procedure of pre-training.

[3]If $d^{(l)}$ is greater than the number of eigenvectors, all eigenvectors are used, with additional vectors generated from random projection of eigenvectors.

Algorithm 1 GPCANet **Forward Pass and Pre-training**

```
1:  **Input:** graph $G = (V, E, X)$, GPCA hyper-param.(s) $\alpha$ (and $\beta$, if supervised), #layers $L$, hidden sizes $\{d_1, \dots, d_L\}$, activation func. $\sigma(\cdot)$, #apprx. steps $T$
2:  **Output:** pre-set layer-wise param.s $\{W^{(1)}, \dots, W^{(L)}\}$
3:  Initialize $H^{(0)} := X$.
4:  for $l = 1$ to $L$ do
5:    Center $H^{(l-1)}$ by subtracting mean of row vectors
6:    $F \leftarrow H^{(l-1)}$
7:    for $t = 1$ to $T$ do
8:      $PF \leftarrow (1-\beta)\tilde{A}_{\text{sym}}F + \beta D^{-\frac{1}{2}}(YY^T)D^{-\frac{1}{2}}F$
9:      $F \leftarrow \frac{\alpha}{1+\alpha}PF + \frac{1}{1+\alpha}H^{(l-1)}$
10:   end for
11:   $W^{(l)} \leftarrow$ top $d_l$ eigenvectors of ${H^{(l-1)}}^T F$
12:   $H^{(l)} \leftarrow \sigma(FW^{(l)})$
13: end for
```

Note that the line marked in blue is an additional step used only for pre-training.

Additional treatment for ReLU: Transformations like ReLU improves model capacity of GPCANet as it can capture nonlinear representations. However at pre-training stage, it causes information loss as all negative values are truncated to 0, which hinders the advantage of using the leading d_l eigenvectors to initialize the weights so as to convey maximum variance (i.e. information) to next layers. To address this issue, we instead use the leading $d_l/2$ eigenvectors $\{e_i\}_{i=1}^{d_l/2}$ and their negatives $\{-e_i\}_{i=1}^{d_l/2}$ to initialize $W^{(l)}$. Empirically we observe this always improves performance when using ReLU activation.

4.1.2. End-to-end-training stage

Pre-training can be seen as an information-preserving initialization, as compared to an uninformative random initialization), after which we refine the layer-wise parameters via gradient-based optimization w.r.t. a supervised loss criterion at the output layer. Specifically for semi-supervised node classification, we perform an end-to-end training w.r.t. the cross-entropy loss on the labeled nodes. All parameters are updated jointly through backpropagation during this stage, with forward computation shown in Alg.1.

4.2. GPCANet-initialization for GCN

When $\alpha = 1, \beta = 0$, and approximating matrix inverse $(I + \alpha\tilde{L})^{-1}$ via first-order truncated Taylor expansion shown in Eq. (11) , GPCANet has the same architecture with GCN. As such, we can use the pre-training stage of GPCANet to initialize GCN with only minor modification. Specifically, we replace lines 6 through 10 in Alg. 1 with a single line:

$$F \leftarrow \tilde{A}_{\text{sym}}H^{(l-1)}$$

Although the modified initialization is for GCN, and is driven by the mathematical connection between GPCANet and GCN that we established, GPCANet-initialization can be used as general-purpose, for other GNNs as well.

5. Experiments

In this section we design extensive experiments to answer the following questions.

- How does the parameter-free, single-layer GPCA compare to layer-wise parameterized, multi-layer GCN?
- To what extent is supervised GPCA beneficial?
- As GPCANet generalizes GCN, can it outperform GCN with the ability to tune regularization via its additional hyperparameters α and β?
- Does GPCANet-initialization help us train better GCN models, in terms of accuracy and robustness, providing better generalization especially with increasing model size (i.e. depth)?

5.1. Experimental Setup

Datasets. We focus on the semi-supervised node classification (SSNC) problem and use 5 benchmark datasets: The first 3 datasets, Cora, CiteSeer, PubMed (Sen et al., 2008), are relatively small (2K to 10K nodes) but widely-used citation networks. For these we use the same data splits as in (Kipf & Welling, 2017). The others, Arxiv and Products, are newest and larger (100K to 2000K) node classification benchmarks from Open Graph Benchmark (Hu et al., 2020), for which we use the official data splits based on real-world scenarios with potential distribution shift. Data statistics can be found in Supp. A.3.

Baseline. For baseline, we only use GCN, as experiments are conducted to verify the established connection between GCN and GPCA instead of achieving the state-of-the-art performance on SSNC. Our analysis also provides insights toward a better understanding of GNNs.

Model configuration. For hyperparameters (HPs), we define a separate pool of values for hidden size, number of layers, weight decay, dropout rate, and regularization trade-off terms α, β for each dataset, where all methods share the same HP pools. Models are trained on every configuration across HP pools and picked based on validation performance. We use the Adam optimizer for all models. Learning rate is first manually tuned for each dataset to achieve stable training, and the same learning rate is fixed for all models—we empirically observed that learning rate is sensitive to datasets but insensitive to models. For GPCA and GPCANet, number of power iterations in Eq. (21) is always set to 5. All experiments use the maximum training epoch as 1000 and repeat 5 times. Detailed configuration of HPs can be found in Supp. A.4. We mainly use a single GTX-1080ti GPU for small datasets Cora, CiteSeer, and PubMed. RTX-3090 GPU is used for Arxiv and Products.

Mini-batch training. As nodes are not independent, GNN is mostly trained in full-batch under semi-supervised setting. We use full-batch training for all datasets except Products, which is too large to fit into GPU memory during training. ClusterGCN (Chiang et al., 2019), a subgraph based mini-batch training algorithm, is used to train GCN and GPCANET. For evaluation, we still use full-batch since a single forward pass can be conducted without memory issues. Initialization is also employed in full-batch.

Fair evaluation. Instead of picking the hyperparameter configurations manually, every value reported in the following sections is based on the *best* configuration selected using validation performance, where all models leverage the *same* hyperparameter pools. Further, each configuration from the pool is conducted 5 times to reduce randomness.

5.2. GPCA vs GCN

5.2.1. UNSUPERVISED GPCA

Having proved the mathematical connection between GPCA and GCN, we hypothesize that unsupervised GPCA can generate a comparable representation to (supervised) GCN. To test this conjecture, we perform GPCA with different α to obtain node representations and then pass those to a 1- or 2-layer MLP. (We use 2-layer MLP for Arxiv and Products as these datasets are large.) The result is shown in Table 1. For reference, the pool for α is $\{1, 5, 10, 20, 50\}$.

Table 1. Comparison btwn. Unsupervised GPCA ($\beta = 0$) and GCN on 5 datasets, w.r.t. mean accuracy and standard deviation (in parentheses) on test set over 5 different seeds with selected configuration, at which model achieves best validation accuracy across all HP configurations in Supp. A.4. GPCA (ALL α) selects best α based on validation, GPCA with specific α uses fixed α.

	CORA	CITESEER	PUBMED	ARXIV	PRODUCTS
GCN	80.62 (0.90)	71.25 (0.05)	78.42 (0.25)	70.64 (0.17)	77.90 (0.33)
GPCA (ALL α)	81.10 (0.00)	71.80 (0.75)	78.78 (0.36)	71.86 (0.18)	79.23 (0.14)
GPCA-α=1	72.57 (0.79)	70.90 (0.58)	76.92 (0.30)	65.47 (0.26)	73.65 (0.07)
GPCA-α=5	80.95 (0.17)	71.80 (0.75)	**79.40** (0.29)	70.69 (0.11)	78.66 (0.09)
GPCA-α=10	**82.23** (0.58)	71.65 (0.53)	78.78 (0.36)	71.37 (0.09)	**79.24** (0.09)
GPCA-α=20	82.05 (0.54)	**72.15** (0.47)	78.15 (0.50)	**71.86** (0.18)	79.23 (0.14)
GPCA-α=50	81.10 (0.00)	71.50 (0.32)	78.00 (0.19)	71.48 (0.15	78.92 (0.10)

Surprisingly, the parameter-free, single-layer GPCA paired with MLP performs consistently better than the end-to-end supervised, multi-layer GCN model across all 5 datasets. By carefully looking at the performance of GPCA with varying α, we find that different datasets have different best α but in general a relatively larger α (comparing with graph convolution of GCN that is equivalent to $\alpha = 1$) is preferable for all datasets. Larger α implies stronger graph-regularization on the representations. The outstanding performance of simple GPCA empirically confirms that the power of GCN on SSNC problem comes from graph regularization, which 1) questions whether GCN or other GNNs can really learn useful representations for SSNC problem by taking advantage of deep neural networks; and 2) points toward a new direction of studying different graph based regularizations to design novel GNNs with new inductive bias.

5.2.2. (SEMI-)SUPERVISED GPCA

The representations generated by unsupervised GPCA does not use any label information from training data. We have extended GPCA to (semi-)supervised setting with an additional HP $\beta \in [0, 1]$ that trades-off graph-based regularization due to the actual input graph edges versus the ones added through YY^T. As such, β=0 corresponds to unsupervised GPCA and larger β uses more information from training set. The latter raises the potential issue of overfitting that can hurt performance when β is too large, or when there is a distribution shift between the training and test sets. For Arxiv and Products, we empirically observe that $\beta > 0$ always hurts performance, possibly because of the distribution difference between training set and test set as described in OGB (Hu et al., 2020). Therefore we only study the effect of β on Cora, CiteSeer and PubMed. For reference, the pool for $\beta > 0$ is $\{0.1, 0.2\}$. Results in Table 2 show that supervised GPCA provides a slight gain over unsupervised GPCA across all 3 datasets.

Table 2. Comparison btwn. Supervised GPCA (β>0), GCN and Unsupervised GPCA on 5 datasets, w.r.t. mean accuracy and standard deviation (in parentheses) on test set over 5 seeds.

	CORA	CITESEER	PUBMED
GCN	80.62 (0.90)	71.25 (0.05)	78.42 (0.25)
UNSUPERVISED GPCA	81.10 (0.00)	71.80 (0.75)	78.78 (0.36)
SUPERVISED GPCA (ALL β>0)	81.17 (0.27)	**73.20 (0.71)**	**79.40 (0.69)**
SUPERVISED GPCA-β=0.1	81.17 (0.27)	72.07 (0.37)	**79.40 (0.69)**
SUPERVISED GPCA-β=0.2	**81.90 (0.00)**	**73.20 (0.71)**	78.73 (0.59)

5.3. GPCANET

Compared to the single-layer GPCA, GPCANET has a deeper architecture like GCN along with nonlinear activation function. Moreover, it employs hyperparameter α at each layer to control the degree of graph regularization. As each graph convolution has fixed level of graph regularization, one may hypothesize that increasing number of layers (L) of GCN corresponds to increasing degree of graph regularization. We empirically test this hypothesis using GPCANET, by varying both L (2 to 10) and α (0.1 to 10) to show their connection (hidden size is fixed as 128). The result is shown in Figure 1. The diagonal pattern empirically suggests that increasing the number of layers has the same effect as increasing graph regularization via α.

The corresponding effect between α and number of layers suggests that we can train a GPCANET with fewer number of layers yet achieve similar regularization by increasing α. Such a shallow model that otherwise behaves like a deep one

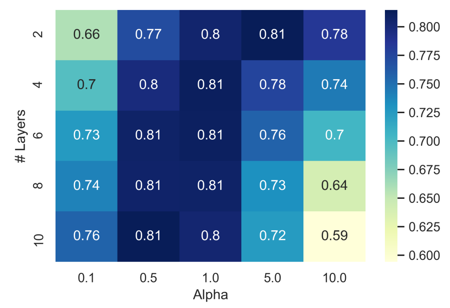

Figure 1. Performance of GPCANet (averaged over 5 different seeds) with varying number of layers and varing α on Cora. Similar results hold for CiteSeer and PubMed.

has the advantage of less memory requirement and faster training due to fewer parameters.

To this end, we train 1–3-layer GPCANet with varying α (see Supp. A.5), and select the best α and number of layers using validation set. We report test set performance in Table 3. We do not observe much improvement by GPCANet over GCN on smaller datasets Cora, CiteSeer, PubMed, but notable gains on the larger Arxiv and Products. As such, GPCANet enables shallow model training via tunable hyperparameter α, achieving comparable or better performance.

Table 3. Comparison btwn. 1–3-layer GPCANet and GCN on 5 datasets, w.r.t. mean accuracy and standard deviation (in parentheses) on test set over 5 different seeds.

	Cora	CiteSeer	PubMed	Arxiv	Products
GCN	80.62 (0.90)	71.25 (0.05)	78.42 (0.25)	70.64 (0.17)	77.90 (0.33)
GPCANet	80.64 (0.33)	71.36 (0.21)	78.52 (0.17)	**72.20** (0.15)	**80.05** (0.29)

5.4. GPCANet-initialization for GCN

Finally, we evaluate the effectiveness of GPCANet-initialization for GCN in terms of performance and robustness under different model sizes, i.e. number of layers L. For comparison, Glorot et al.'s (2010) Xavier initialization is used to initialize GCN.

We report the test set performance (averaged over 5 seeds) of the GCN model with the best configuration based on validation data in Table 4. The results show that GPCANet-initialization tends to outperform the widely-used Xavier initialization, where the improvement grows with increasing number of layers. Notably, GCN with GPCANet-initialization exhibits stable performance across all layers.

Instead of only looking at the average performance, we further study whether GPCANet-initialization improves the training robustness, by reducing performance variation across different seeds. To this end, we first choose the best configuration for each initialization method based on validation performance, and train the GCN model with the chosen configuration using 100 random seeds.

Table 4. Test set performance of GCN with Xaiver initialization versus GPCANet initialization, w.r.t. varying number of layers (L) across all datasets. Each reported value is based on the best configuration selected using validation performance.

Dataset	$2L$	$3L$	$5L$	$10L$	$15L$
Cora Xaiver-init	80.62	**80.62**	79.40	76.37	66.07
Cora GPCANet-init	**81.67**	79.50	**80.90**	**79.82**	**78.00**
CiteSeer Xaiver-init	71.25	**70.15**	**71.10**	61.90	57.40
CiteSeer GPCANet-init	**71.27**	69.27	70.15	**68.67**	**67.87**
PubMed Xaiver-init	**78.42**	**77.90**	77.07	77.00	45.80
PubMed GPCANet-init	78.05	77.25	**78.07**	**77.80**	**78.03**
Arxiv Xaiver-init	69.61	**70.64**	70.33	68.32	61.68
Arxiv GPCANet-init	**69.76**	70.72	**70.52**	**69.77**	**66.28**
Products Xaiver-init	77.90	78.65	78.08	76.27	74.70
Products GPCANet-init	**78.13**	**78.71**	**78.22**	**77.47**	**75.90**

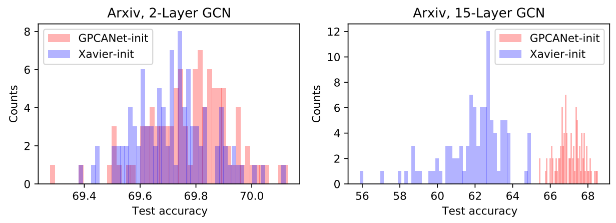

Figure 2. Comparison between Xavier-init and GPCANet-init in terms of test accuracy robustness over 100 seeds on Arxiv.

In Figure 2 we present the histogram of test set accuracy over 100 runs for Arxiv. (For results on other datasets, see Supp. A.6.) For both 2-layer and 15-layer GCN, GPCANet-initialization not only outperforms Xavier-initialization w.r.t. average performance, but also in terms of robustness, achieving much lower performance variation and few bad outliers, especially for deeper GCN. As such, it acts as a good, data-driven prior, facilitating the training of many parameters across layers by identifying a promising region of the parameter space from which supervised fine-tuning is initiated.

6. Conclusion

In this work we have (1) discovered a mathematical connection between GPCA and GCN's graph convolution; (2) extended GPCA to the (semi-)supervised setting; (3) proposed GPCANet, by stacking GPCA and nonlinear transformation, which is a generalized GCN model with additional hyperparameters to control the degree of graph regularization, and (4) based on the established connection, proposed the GPCANet-initialization for GCN. Accordingly, we designed extensive experiments demonstrating that (i) the simple, single-layer GPCA achieves comparable or better performance than GCN, which suggests that GCN's power is driven by graph regularization, (ii) GPCANet allows training of shallow models with competitive performance via increasing the degree of graph regularization at each layer, reducing memory and training time cost, and finally (iii) GPCANet-initialization acts as a good data-driven prior for GCN training, providing robust performance.

References

Bengio, Y., Courville, A., and Vincent, P. Representation learning: A review and new perspectives. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 35 (8):1798–1828, August 2013. ISSN 0162-8828. doi: 10.1109/TPAMI.2013.50. URL `http://ieeexplore.ieee.org/document/6472238/`. Zu bearbeitendes Review.

Chan, T.-H., Jia, K., Gao, S., Lu, J., Zeng, Z., and Ma, Y. PCANet: A simple deep learning baseline for image classification? *IEEE transactions on image processing*, 24(12):5017–5032, 2015.

Chen, Z., Chen, L., Villar, S., and Bruna, J. Can graph neural networks count substructures? *arXiv preprint arXiv:2002.04025*, 2020.

Chiang, W.-L., Liu, X., Si, S., Li, Y., Bengio, S., and Hsieh, C.-J. Cluster-gcn: An efficient algorithm for training deep and large graph convolutional networks. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pp. 257–266, 2019.

Errica, F., Podda, M., Bacciu, D., and Micheli, A. A fair comparison of graph neural networks for graph classification. In *International Conference on Learning Representations*, 2020. URL `https://openreview.net/forum?id=HygDF6NFPB`.

Fey, M. and Lenssen, J. E. Fast graph representation learning with PyTorch Geometric. In *ICLR Workshop on Representation Learning on Graphs and Manifolds*, 2019.

Gallagher, B., Tong, H., Eliassi-Rad, T., and Faloutsos, C. Using ghost edges for classification in sparsely labeled networks. In *KDD*, pp. 256–264. ACM, 2008. URL `http://dblp.uni-trier.de/db/conf/kdd/kdd2008.html#GallagherTEF08`.

Garg, V., Jegelka, S., and Jaakkola, T. Generalization and representational limits of graph neural networks. In *International Conference on Machine Learning*, pp. 3419–3430. PMLR, 2020.

Glorot, X. and Bengio, Y. Understanding the difficulty of training deep feedforward neural networks. In *Proceedings of the thirteenth international conference on artificial intelligence and statistics*, pp. 249–256, 2010.

Golub, G. and Van Loan, C. *Matrix Computations*. Johns Hopkins University Press, 1989.

Hamilton, W., Ying, Z., and Leskovec, J. Inductive representation learning on large graphs. In *Advances in neural information processing systems*, pp. 1024–1034, 2017.

He, K., Zhang, X., Ren, S., and Sun, J. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In *Proceedings of the IEEE international conference on computer vision*, pp. 1026–1034, 2015.

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. *arXiv preprint arXiv:2005.00687*, 2020.

Ioffe, S. and Szegedy, C. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *International conference on machine learning*, pp. 448–456. PMLR, 2015.

Jiang, B., Ding, C., Luo, B., and Tang, J. Graph-laplacian PCA: Closed-form solution and robustness. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 3492–3498, 2013.

Kipf, T. N. and Welling, M. Semi-supervised classification with graph convolutional networks. In *International Conference on Learning Representations (ICLR)*, 2017.

Klicpera, J., Bojchevski, A., and Günnemann, S. Predict then propagate: Graph neural networks meet personalized pagerank. In *International Conference on Learning Representations (ICLR)*, 2019.

Krähenbühl, P., Doersch, C., Donahue, J., and Darrell, T. Data-dependent initializations of convolutional neural networks. In *International Conference on Learning Representations*, 2016.

Krizhevsky, A., Sutskever, I., and Hinton, G. E. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pp. 1097–1105, 2012.

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

Liao, R., Urtasun, R., and Zemel, R. A {pac}-bayesian approach to generalization bounds for graph neural networks. In *International Conference on Learning Representations*, 2021. URL `https://openreview.net/forum?id=TR-Nj6nFx42`.

Loukas, A. What graph neural networks cannot learn: depth vs width. In *International Conference on Learning Representations*, 2020a. URL `https://openreview.net/forum?id=B1l2bp4YwS`.

Loukas, A. How hard is to distinguish graphs with graph neural networks? Technical report, 2020b.

Low, C.-Y., Teoh, A. B.-J., and Toh, K.-A. Stacking PCANet+: An overly simplified convnets baseline for face recognition. *IEEE Signal Processing Letters*, 24(11): 1581–1585, 2017.

Mishkin, D. and Matas, J. All you need is a good init. *arXiv preprint arXiv:1511.06422*, 2015.

Morris, C., Ritzert, M., Fey, M., Hamilton, W. L., Lenssen, J. E., Rattan, G., and Grohe, M. Weisfeiler and leman go neural: Higher-order graph neural networks. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pp. 4602–4609, 2019.

NT, H. and Maehara, T. Revisiting graph neural networks: All we have is low-pass filters. *arXiv preprint arXiv:1905.09550*, 2019.

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

Pan, V. Y. and Chen, Z. Q. The complexity of the matrix eigenproblem. In *Proceedings of the thirty-first annual ACM symposium on Theory of computing*, pp. 507–516, 1999.

Saxe, A. M., McClelland, J. L., and Ganguli, S. Exact solutions to the nonlinear dynamics of learning in deep linear neural networks. *arXiv preprint arXiv:1312.6120*, 2013.

Sen, P., Namata, G., Bilgic, M., Getoor, L., Galligher, B., and Eliassi-Rad, T. Collective classification in network data. *AI magazine*, 29(3):93–93, 2008.

Seuret, M., Alberti, M., Liwicki, M., and Ingold, R. Pca-initialized deep neural networks applied to document image analysis. In *2017 14th IAPR international conference on document analysis and recognition (ICDAR)*, volume 1, pp. 877–882. IEEE, 2017.

Shahid, N., Perraudin, N., Kalofolias, V., Puy, G., and Vandergheynst, P. Fast robust PCA on graphs. *IEEE Journal of Selected Topics in Signal Processing*, 10(4):740–756, 2016.

Srinivasan, B. and Ribeiro, B. On the equivalence between positional node embeddings and structural graph representations. In *International Conference on Learning Representations*, 2020. URL https://openreview.net/forum?id=SJxzFySKwH.

Veličković, P., Cucurull, G., Casanova, A., Romero, A., Liò, P., and Bengio, Y. Graph Attention Networks. In *International Conference on Learning Representations*, 2018.

Veličković, P., Fedus, W., Hamilton, W. L., Liò, P., Bengio, Y., and Hjelm, R. D. Deep graph infomax. In *International Conference on Learning Representations*, 2019. URL https://openreview.net/forum?id=rklz9iAcKQ.

Wagner, R., Thom, M., Schweiger, R., Palm, G., and Rothermel, A. Learning convolutional neural networks from few samples. In *The 2013 International Joint Conference on Neural Networks (IJCNN)*, pp. 1–7. IEEE, 2013.

Wu, F., Souza, A., Zhang, T., Fifty, C., Yu, T., and Weinberger, K. Simplifying graph convolutional networks. In *International Conference on Machine Learning*, pp. 6861–6871, 2019.

Xu, K., Hu, W., Leskovec, J., and Jegelka, S. How powerful are graph neural networks? In *International Conference on Learning Representations*, 2019. URL https://openreview.net/forum?id=ryGs6iA5Km.

Zhang, Z. and Zhao, K. Low-rank matrix approximation with manifold regularization. *IEEE transactions on pattern analysis and machine intelligence*, 35(7):1717–1729, 2012.

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

A. Supplementary Material

A.1. Derivation from Eq. (8) to Eq. (9)

For this part only, let $A = (I + \alpha\tilde{L})^{-1}$ to simplify the notation. We can show that (7) is equivalent to

$$\begin{aligned}
&\min_{W, W^TW=I} \quad \mathrm{Tr}(XX^T) - 2\,\mathrm{Tr}(AXWW^TX^T) \\
&\quad + \mathrm{Tr}(AXWW^TWW^TX^TA) + \alpha\,\mathrm{Tr}(W^TX^TA\tilde{L}AXW) \\
&\equiv \max_{W, W^TW=I} \quad 2\,\mathrm{Tr}(AXWW^TX^T) - \mathrm{Tr}(AXWW^TX^TA) \\
&\qquad - \alpha\,\mathrm{Tr}(W^TX^TA\tilde{L}AXW) \qquad (22)
\end{aligned}$$

Using the cyclic property of (Tr)ace, we can write

$$\begin{aligned}
&\max_{W, W^TW=I} \quad 2\,\mathrm{Tr}(W^TX^TAXW) - \mathrm{Tr}(W^TX^TAAXW) \\
&\qquad - \alpha\,\mathrm{Tr}(W^TX^TA\tilde{L}AXW) \\
&\max_{W, W^TW=I} \quad \mathrm{Tr}\left[W^TX^T(2A - AA - A(\alpha\tilde{L})A)XW\right] \\
&\max_{W, W^TW=I} \quad \mathrm{Tr}\left[W^TX^T\big(A + \{I - A(I+\alpha\tilde{L})\}A\big)XW\right] \\
&\max_{W, W^TW=I} \quad \mathrm{Tr}\left[W^TX^T(I+\alpha\tilde{L})^{-1}XW\right] \qquad (23)
\end{aligned}$$

where the objective simplifies upon replacing A with $(I + \alpha\tilde{L})^{-1}$.

A.2. Derivation from Eq. (13) to Eq. (9)

$$\begin{aligned}
&\max_{\mathbf{z}} \quad \left[\mathrm{corr}(Y,\mathbf{z})\right]^T\left[\mathrm{corr}(Y,\mathbf{z})\right]\mathrm{var}(\mathbf{z}) \\
&\equiv \max_{\mathbf{z}} \quad \mathrm{var}(Y)\left[\mathrm{corr}(Y,\mathbf{z})\right]^T\left[\mathrm{corr}(Y,\mathbf{z})\right]\mathrm{var}(\mathbf{z}) \qquad (24) \\
&\equiv \max_{\mathbf{z}} \quad \left[\mathrm{cov}(Y,\mathbf{z})\right]^T\left[\mathrm{cov}(Y,\mathbf{z})\right] \qquad (25) \\
&\qquad \text{where } \mathrm{cov}(Y,\mathbf{z}) = \sqrt{\mathrm{var}(Y)}\mathrm{corr}(Y,\mathbf{z})\sqrt{\mathrm{var}(\mathbf{z})} \\
&\equiv \max_{\mathbf{z}} \quad \left[Y^T\mathbf{z}\right]^T\left[Y^T\mathbf{z}\right] \qquad (26) \\
&\equiv \max_{\mathbf{z}} \quad \mathbf{z}^TYY^T\mathbf{z} \qquad (27)
\end{aligned}$$

Note that in (24) we added the term $\mathrm{var}(Y)$ without affecting the optimization problem which is w.r.t. $\mathbf{z}$.

A.3. Dataset Statistics

Table 5. Statistics of used datasets.

Dataset	#Nodes	#Edges	#Feat.	#Cls.	Train/Vali./Test
Cora	2,708	5,429	1,433	7	5.2%/18.5%/36.9%
CiteSeer	3,327	4,732	3,703	6	3.6%/15%/30%
PubMed	19,717	44,338	500	3	0.3%/2.5%/5%
Arxiv	169,343	1,166,243	128	40	54%/18%/28%
Products	2,449,029	61,859,140	100	47	8%/2%/90%

Datasets used in the experiments are presented in Table 5. Cora, CiteSeer, and PubMed can be downloaded in Pytorch Geometric Library (Fey & Lenssen, 2019). Arxiv and Products can be accessed in `https://ogb.stanford.edu/`.

A.4. Hyperparameter Configurations

We setup hyperparameters pool for each dataset, presented in Table 6. All methods use the same pool. The only exception is GPCA, as GPCA is just a 1-layer shallow model which can be trained with lager learning rate, we use 0.1 learning rate for it on all datasets.

Table 6. Hyperparameters pool for each dataset, includes learning rate (LR), weight decay (WD), number of layers (#Layers), hidden size, dropout, α, and β. For Arxiv and Products, weight decay is set as 0 because the dataset is large and no overfit happened. Same reason for choosing smaller dropout rate for them.

Dataset	LR	WD	#Layers	Hidden
Cora	0.001	[0.0005, 0.005, 0.05]	[2, 3, 5, 10, 15]	[128, 256]
CiteSeer	0.001	[0.0005, 0.005, 0.05]	[2, 3, 5, 10, 15]	[128, 256]
PubMed	0.001	[0.0005, 0.005, 0.05]	[2, 3, 5, 10, 15]	[128, 256]
Arxiv	0.005	0	[2, 3, 5, 10, 15]	[128, 256]
Products	0.001	0	[2, 3, 5, 10, 15]	[128, 256]

Dataset	Dropout	α	β
Cora	[0, 0.5]	[1, 5, 10, 20, 50]	[0, 0.1, 0.2]
CiteSeer	[0, 0.5]	[1, 5, 10, 20, 50]	[0, 0.1, 0.2]
PubMed	[0, 0.5]	[1, 5, 10, 20, 50]	[0, 0.1, 0.2]
Arxiv	[0, 0.2]	[1, 5, 10, 20, 50]	0
Products	[0, 0.1]	[1, 5, 10, 20, 50]	0

A.5. Configurations for Experiments of $1\sim3$-Layer GPCANet

The goal is train a shallow GPCANet with tunable α (β=0 is used), we setup different α pool for different number of layers, because the effect of increasing α is the same to increasing number of layers (shown in Figure 1). We report the pool for α for each layer in Table 7. For other parameters we use the same setting mentioned in Table 6.

Table 7. Pool of α for $1\sim3$-layer GPCANet, same across all datasets.

# Layers	Pool of α
1-layer	[10, 20, 30]
2-layer	[3, 5, 10]
3-layer	[1, 2, 3, 5]

A.6. GPCANet-Init's Robustness for Additional Datasets

Histogram of test set accuracy over 100 runs for GCN initialized by Xavier-initialization and GPCANet-initialization in Cora (Figure 3), CiteSeer (Figure 4), and PubMed (Figure

5). We have ignored Products as it takes too long to run 100 times, but the result should be similar.

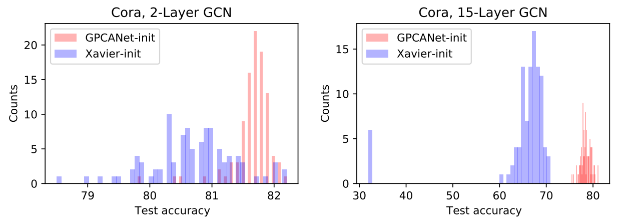

Figure 3. Comparison between Xavier-init and GPCANET-init in terms of test accuracy robustness over 100 seeds on Cora.

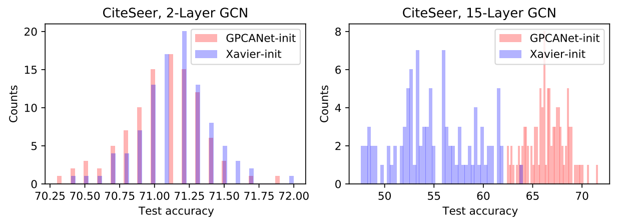

Figure 4. Comparison between Xavier-init and GPCANET-init in terms of test accuracy robustness over 100 seeds on CiteSeer.

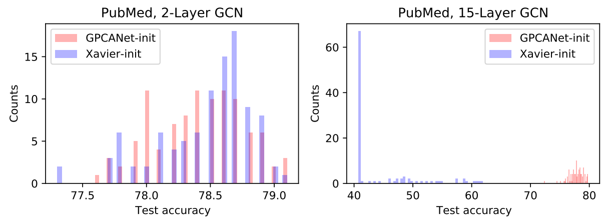

Figure 5. Comparison between Xavier-init and GPCANET-init in terms of test accuracy robustness over 100 seeds on PubMed.