Graph Neural Networks Inspired by Classical Iterative Algorithms

Yongyi Yang [1 †] **Tang Liu** [1 †] **Yangkun Wang** [2 †] **Jinjing Zhou** [3] **Quan Gan** [3] **Zhewei Wei** [4] **Zheng Zhang** [3] **Zengfeng Huang** [1] **David Wipf** [3]

Abstract

Despite the recent success of graph neural networks (GNN), common architectures often exhibit significant limitations, including sensitivity to oversmoothing, long-range dependencies, and spurious edges, e.g., as can occur as a result of graph heterophily or adversarial attacks. To at least partially address these issues within a simple transparent framework, we consider a new family of GNN layers designed to mimic and integrate the update rules of two classical iterative algorithms, namely, proximal gradient descent and iterative reweighted least squares (IRLS). The former defines an extensible base GNN architecture that is immune to oversmoothing while nonetheless capturing long-range dependencies by allowing arbitrary propagation steps. In contrast, the latter produces a novel attention mechanism that is explicitly anchored to an underlying end-to-end energy function, contributing stability with respect to edge uncertainty. When combined we obtain an extremely simple yet robust model that we evaluate across disparate scenarios including standardized benchmarks, adversarially-perturbated graphs, graphs with heterophily, and graphs involving long-range dependencies. In doing so, we compare against SOTA GNN approaches that have been explicitly designed for the respective task, achieving competitive or superior node classification accuracy. Our code is available at this link. And for an extended version of this work, please see (Yang et al., 2021).

[†]Work completed during an internship at the AWS Shanghai AI Lab. [1]Fudan University [2]Shanghai Jiao Tong University [3]Amazon [4]Renmin University of China. Correspondence to: Zengfeng Huang <huangzf@fudan.edu.cn>, David Wipf <davidwipf@gmail.com>.

Proceedings of the 38^{th} International Conference on Machine Learning, PMLR 139, 2021.

1. Introduction

Given data comprised of entities with explicit relationships between them, graph neural networks (GNNs) represent a powerful means of exploiting these relationships within a predictive model (Zhou et al., 2018). Among the most successful candidates in this vein, message-passing GNNs are based on stacking a sequence of propagation layers, whereby neighboring nodes share feature information via an aggregation function (Kipf & Welling, 2017a; Hamilton et al., 2017; Kearnes et al., 2016). This sharing is also at times modulated by various forms of attention to effectively compute propagation weights that reflect the similarity between nodes connected by an edge (Velickovic et al., 2018).

While such approaches display impressive performance on graph benchmarks involving tasks such as semi-supervised node classification conditioned on input features, non-trivial shortcomings remain. For example, as we deviate from more standardized regimes exhibiting graph homophily, meaning nearby nodes share similar labels and features, into the less familiar, contrasting domain of heterophily, many message-passing GNNs (including those with attention) experience a steep degradation in accuracy (Zhu et al., 2020b;a). Similarly, when edge uncertainty is introduced via adversarial attacks or related, performance declines as well (Zügner et al., 2019; Zügner & Günnemann, 2019).

Pushing further, for problems requiring that nodes share information across non-local regions of the graph, message-passing GNNs require a large number of propagation layers. However, this can lead to a well-known oversmoothing phenomena in which node features converge to similar, non-discriminative embeddings (Oono & Suzuki, 2020; Li et al., 2018). To this end, a wide variety of countermeasures have been deployed to address this concern, most notably various types of skip connections designed to preserve local information even while propagating to distant neighbors (Chen et al., 2020; Li et al., 2019; Xu et al., 2018). Even so, most deeper architectures generally display modest improvement, and may require additional tuning or regularization strategies to be effective.

And finally, beyond the above issues there is at times a lingering opaqueness in terms of why and how different GNN architectural choices relate to performance on specific

tasks. This is due, seemingly in part, to the importance of nuanced heuristics in obtaining SOTA performance on graph leaderboards, as opposed to more transparent designs that are likely to consistently transfer to new domains.

Our Contributions As a partial step towards mitigating these concerns, we propose a transparent GNN architecture with an inductive bias inspired by classical iterative algorithms. In doing so, our design benefits from the following:

- All architectural components are in one-to-one correspondence with the unfolded iterations of robust descent algorithms applied to minimizing a principled graph-regularized energy function. Specifically, we adopt propagation layers and nonlinear activations patterned after proximal gradient updates (Section 2), while we integrate these steps with graph attention using iterative reweighted least squares (IRLS) (Section 3). To the best of our knowledge, this is the first end-to-end GNN model, including attention, designed based on such an unfolding perspective (Section 4).
- By anchoring to a principled energy function, we can include an arbitrary number of propagation layers to capture non-local dependencies, but with no risk of oversmoothing. And by unifying each layer via IRLS reweighting functions, we can include a novel family of attention mechanisms that simultaneously contribute robustness to graph uncertainty without introducing additional parameters or design heuristics.
- In head-to-head comparisons with SOTA methods explicitly tailored for disparate testing conditions including standard benchmarks, graphs with heterophily, and adversarial perturbations, our proposed architecture achieves competitive or superior node classification accuracy (Section 5).

2. A Simple GNN Backbone Derived from Proximal Gradient Descent

Our starting point will be a class of affine GNN layers that are immune to oversmoothing and yet are sufficiently flexible to handle nonlinear activations, sampling, and layer-dependent weights within a unified framework. As described next, this is achievable by unfolding the iterations of proximal gradient descent methods applied to a suitably regularized energy function.

2.1. Extensible GNN Layers via Unfolding

Given a graph $\mathcal{G} = \{\mathcal{V}, \mathcal{E}\}$, with $n = |\mathcal{V}|$ nodes and $m = |\mathcal{E}|$ edges, inspired by (Zhou et al., 2004) we define the energy function

$$\ell_Y(Y) \triangleq \|Y - f(X;W)\|_{\mathcal{F}}^2 + \lambda \text{tr}\left[Y^\top L Y\right], \quad (1)$$

where λ is a trade-off parameter and $Y \in \mathbb{R}^{n\times d}$ represents a candidate embedding of d-dimensional features across n nodes. Similarly, $f(X;W)$ denotes a function (parameterized by W) of initial d_0-dimensional node features $X \in \mathbb{R}^{n\times d_0}$. For example, we can have $f(X;W) = XW$ or $f(X;W) = \text{MLP}[X;W]$, i.e., a multi-layer perceptron with weights W. And finally, $L \in \mathbb{R}^{n\times n}$ is the Laplacian of $\mathcal{G}$, meaning $L = D - A = B^\top B$, where D and A are degree and adjacency matrices respectively, while $B \in \mathbb{R}^{m\times n}$ is an incidence matrix.

It is not difficult to show that

$$Y^*(W) \triangleq \arg\min_Y \ell_Y(Y) = (I + \lambda L)^{-1} f(X;W), \quad (2)$$

where obviously the optimal Y^* will be a function of W. This solution represents a loose approximation to $f(X;W)$ that has been smoothed across the graph structure via the trace term in (1), balancing local and global information (Zhou et al., 2004). Hence it is reasonable to treat $Y^*(W)$ as graph-aware features for application to a downstream prediction problem of interest, e.g., node classification.

Given that $Y^*(W)$ is differentiable w.r.t. W, we need only plug into a node classification or regression loss of the form

$$\ell_W(W,\theta) \triangleq \sum_{i=1}^{n'} \mathcal{D}\Big(g\left[\boldsymbol{y}_i^*(W);\theta\right], t_i\Big), \quad (3)$$

where $g : \mathbb{R}^d \to \mathbb{R}$ is a node-wise function with parameters θ, e.g., g could be a linear transformation, an MLP, or even an identity mapping. Additionally, $\boldsymbol{y}_i^*(W)$ is the i-th row of $Y^*(W)$, t_i is a ground-truth target label of node i, and $\mathcal{D}$ is some discriminator function, e.g., cross-entropy for classification, squared error for regression.[1] We may then optimize $\ell_W(W,\theta)$ over both W and θ. In this capacity, each $\boldsymbol{y}_i^*(W)$ serves as a trainable node-level feature that has been explicitly regularized via the graph structure.

While the above strategy may be feasible for small graphs, if n is large then computing the inverse of $I + \lambda L$, a requirement for producing $Y^*(W)$ analytically, can be prohibitively expensive. As a more feasible alternative, we may instead approximate $Y^*(W)$ with an estimate formed by taking gradient steps along (1) with respect to Y, starting from some initial point $Y^{(0)}$, e.g., $Y^{(0)} = f(X;W)$ (Zhou et al., 2004). As long as these gradient steps are themselves differentiable w.r.t. W, we can still plug the resulting approximate estimator $\hat{Y}(W)$ into the meta-objective (3) and proceed as before. In this sense we are essentially *unfolding* a sequence of gradient steps to form a differentiable chain that can be viewed as trainable layers of a deep architecture (Gregor & LeCun, 2010; Hershey et al., 2014; Sprechmann

[1] Note that without loss of generality, (3) implicitly assumes the first $n' < n$ nodes have been labeled for training.

et al., 2015). Analogous strategies have been used in the past to facilitate meta-learning using features originating from a lower-level loss whose optimum itself is not directly differentiable (Wang et al., 2016).

To proceed, we note that

$$\frac{\partial \ell_Y(Y)}{\partial Y} = 2\lambda LY + 2Y - 2f(X;W), \tag{4}$$

and therefore iteration $k+1$ of gradient descent can be computed as

$$Y^{(k+1)} = Y^{(k)} - \alpha\left[QY^{(k)} - f(X;W)\right], \;\; Q \triangleq \lambda L + I, \tag{5}$$

where $\frac{\alpha}{2}$ is the step size. But if Q has a large condition number, the gradient descent convergence rate can be slow (Nocedal & Wright, 2006). Fortunately though, preconditioning techniques exist to reduce the convergence time.

For example, as motivated by the Richardson iterative method for solving linear systems (Axelsson, 1996), Jacobi preconditioning involves simply rescaling each gradient step using $(\text{diag}[Q])^{-1} = (\lambda D + I)^{-1}$. After rearranging terms and defining the diagonal matrix $\tilde{D} \triangleq \lambda D + I$, this leads to the modified update

$$Y^{(k+1)} = (1-\alpha)Y^{(k)} + \alpha\tilde{D}^{-1}\left[\lambda AY^{(k)} + f(X;W)\right]. \tag{6}$$

We will henceforth treat this expression as the k-th differentiable propagation layer of a GNN model; later we will discuss enhancements of this baseline designed to provide greater flexibility and robustness. Note also that, scale factors $(1-\alpha)$ and $\alpha\tilde{D}^{-1}$ notwithstanding, (6) can be interpreted as having skip connections from both the previous layer $Y^{(k)}$ and the input features $f(X;W)$. While prior work has also advocated the use of skip connections in various forms to help mitigate oversmoothing effects (Chen et al., 2020; Li et al., 2019; Xu et al., 2018), within the unfolding framework such connections naturally emerge from an underlying energy function, i.e., they are not introduced as a post hoc patch.

2.2. Relationship with Graph Convolution Layers

Interestingly, under suitable assumptions detailed in the supplementary, if we choose $Y^{(0)} = f(X;W) = \tilde{D}^{1/2}XW$, then the first gradient step $Y^{(1)}$ is equivalent, up to a simple reparameterization by $\tilde{D}^{1/2}$, to the *graph convolutional network* (GCN) layer $\tilde{D}^{-1/2}\tilde{A}\tilde{D}^{-1/2}XW$. See (Kipf & Welling, 2017a) for further details regarding the GCN architecture. We now examine similarities and differences across arbitrary iterations with respect to oversmoothing, nonlinear activations, layer-dependent weights, and sampling.

Oversmoothing Unlike $k=1$, for iterations $k>1$, (6) diverges significantly from GCN layers. This is reflected in critical differences in the asymptotic propagation of (6) versus the analogous GCN-based rule

$$Y^{(k+1)} = \tilde{D}^{-1/2}\tilde{A}\tilde{D}^{-1/2}Y^{(k)}W, \tag{7}$$

where (7) has been explored in (Wu et al., 2019a) under the moniker *simple graph convolutions* (SGC). Specifically, while (6) converges to the solution given by (2), which represents a principled graph diffusion akin to personalized page rank (Klicpera et al., 2019), SGC with $k \to \infty$ converges to an *oversmoothed* solution whereby $\boldsymbol{y}_i^{(\infty)} = \boldsymbol{c}$ for all $i \in \mathcal{V}$, i.e., all node embeddings converge to a constant $\boldsymbol{c}$ (Wu et al., 2019a). As such, by treating (6) as the propagation rule for GNN layers, oversmoothing is not a concern. This is worth noting given the signification consideration given to this topic in the literature (Oono & Suzuki, 2020; Li et al., 2018; Rong et al., 2020).

Nonlinear Activations The original GCN model from (Kipf & Welling, 2017a) followed each convolutional layer with a node-wise nonlinear ReLU activation function. Nonlinear activations can also naturally be introduced within the unfolding framework described in Section 2.1. In fact, there is a direct correspondence between the inclusion of an additional nonlinear regularization term (or constraint) in (1), and the introduction of a nonlinear activation within the update (6). Specifically, let $\eta : \mathbb{R}^d \to \mathbb{R}^d$ denote any function of individual node features. The revised optimization problem

$$\min_Y \; \ell_Y(Y) + \sum_i \eta(\boldsymbol{y}_i) \tag{8}$$

can be efficiently solved using proximal gradient methods (Combettes & Pesquet, 2011), a generalized form of projected gradient descent, whenever the proximal operator

$$\text{prox}_\eta(\boldsymbol{u}) \triangleq \arg\min_{\boldsymbol{y}} \tfrac{1}{2}\|\boldsymbol{u}-\boldsymbol{y}\|_2^2 + \eta(\boldsymbol{y}) \tag{9}$$

is easily computable in closed form when applied to any generic input argument $\boldsymbol{u}$. As an illustrative special case, if $\eta(\boldsymbol{y}) = \sum_j \mathcal{I}_\infty[y_j < 0]$, meaning an indicator function assigning infinite penalty to any $y_j < 0$ (this is effectively equivalent to a constraint on $\boldsymbol{y}$ to the positive orthant), the corresponding proximal operator satisfies $\text{prox}_\eta(\boldsymbol{u}) = \text{ReLU}(\boldsymbol{u}) = \max(\boldsymbol{0}, \boldsymbol{u})$. The proximal descent version of (6) then becomes

$$Y^{(k+1)} = \tag{10}$$
$$\text{ReLU}\left((1-\alpha)Y^{(k)} + \alpha\tilde{D}^{-1}\left[\lambda AY^{(k)} + f(X;W)\right]\right),$$

with guaranteed descent for suitable step sizes (Combettes & Pesquet, 2011). In wider contexts, proximal gradient methods have been applied to designing deep architectures for structured regression tasks (Sprechmann et al., 2015).

Layer-Dependent Weights and Sampling Within the unfolding framework, layer-dependent weights can be effectively introduced by simply altering the specification of

the norms used to define $\ell_Y(Y)$ such that a parameterized/learnable metric is included. Additionally, sampling methods designed to sparsify the graph, which can significantly reduce time and memory costs (Chen et al., 2018), can also naturally be accommodated using an unbiased stochastic estimate of $\ell_Y(Y)$. In both cases, please see the supplementary for more details.

3. Robust Regularization Using IRLS-Based Graph Attention

Now suppose we do *not* believe that

$$\text{tr}\left[Y^\top LY\right] = \|BY\|_{\mathcal{F}}^2 = \sum_{\{i,j\}\in\mathcal{E}} \left\|\boldsymbol{y}_i - \boldsymbol{y}_j\right\|_2^2 \qquad (11)$$

is the most appropriate regularization factor for incorporating graph structure within (1) (here we are assuming that all edge weights are unity, although this assumption can be relaxed). To inject additional flexibility, we may consider upgrading (11), leading to the revised energy $\ell_Y(Y;\rho) \triangleq$

$$\|Y - f(X;W)\|_{\mathcal{F}}^2 + \lambda \sum_{\{i,j\}\in\mathcal{E}} \rho\left(\left\|\boldsymbol{y}_i - \boldsymbol{y}_j\right\|_2^2\right). \qquad (12)$$

Here $\rho : \mathbb{R}^+ \to \mathbb{R}$ is some nonlinear function designed to contribute robustness, particularly w.r.t edge uncertainty.

In this section, we will first motivate how edge uncertainty, which undercuts the effectiveness of (11), can be modeled using Gaussian scale mixtures (GSM) (Andrews & Mallows, 1974). We later apply this perspective to informing suitable choices for ρ in (12). Finally, we map the updated energy to an idealized graph attention mechanism that can be integrated within the unfolding framework from Section 2 via IRLS (Chartrand & Yin, 2008; Daubechies et al., 2010).

3.1. Edge Uncertainty and Gaussian Scale Mixtures

The quadratic penalty (11) is in some sense the optimal regularization factor for Gaussian errors aligned with the graph structure (Jia & Benson, 2020), and indeed, when we apply a $\exp\left[-(\cdot)\right]$ transformation and suitable normalization we obtain a structured Gaussian prior distribution $p(Y)$ with unit variance along each edge. In fact, from this perspective minimizing (1) is equivalent to *maximum a posteriori* (MAP) estimation via $p(Y|X) \propto p(X|Y)p(Y)$, with likelihood $p(X|Y) \propto \exp\left[-\frac{1}{2\lambda}\|Y - f(X;W)\|_{\mathcal{F}}^2\right]$.

But of course it is well known that ℓ_2-norms and Gaussian priors are very sensitive to outliers, since errors accumulate quadratically (West, 1984). In the present context, this would imply that spurious edges (e.g., that were errantly included in the graph) or which link nodes with neither label nor features in common, could dominate the resulting loss leading to poor performance on downstream tasks.

Consequently, given the potential weakness of (11), and by extension (1), it may be preferable to introduce some uncertainty into the allowable variance/penalty along each edge. More specifically, we can replace the fixed, unit variances implicitly assumed above with the more flexible Gaussian scale mixture prior defined as

$$p(Y) = Z^{-1} \prod_{\{i,j\}\in\mathcal{E}} \int \mathcal{N}\left(\boldsymbol{u}_{ij}|0,\gamma_{ij}^{-1}I\right) d\mu\left(\gamma_{ij}\right), \qquad (13)$$

where μ is some positive measure over latent precision parameters γ_{ij}, $\boldsymbol{u}_{ij} \triangleq \boldsymbol{y}_i - \boldsymbol{y}_j \; \forall\{i,j\} \in \mathcal{E}$, and Z is a standard partition function that ensures (13) sums to one. When the measure μ assigns all of its mass to $\gamma_{ij} = 1$ for all edges, then $-\log p(Y)$ so-defined collapses to (11); however, in broader scenarios this model allows us to consider a distribution of precision (or variance) across edges. For example, if μ allocates non-trivial mass to small precision values, it reflects our belief that some edges may be spurious, and the penalty on large values of $\|\boldsymbol{y}_i - \boldsymbol{y}_j\|_2$ is reduced (i.e., it is rescaled by γ_{ij}). As a special case, neighbor sampling (Chen et al., 2018) can be motivated by choosing μ such that all probability mass is partitioned between $\gamma_{ij} = 1$ and $\gamma_{ij} \to 0$ for each edge. The proportions will determine the sampling probability, and sampling each γ_{ij} can be viewed as achieving a *biased* estimator of $-\log p(Y)$.

3.2. From Edge Uncertainty to Graph Attention

Generally speaking, the added flexibility of (13) comes with a significant cost in terms of algorithmic complexity. In particular, sampling-based approximations notwithstanding, the integral over each γ_{ij} may be intractable, and the resulting MAP problem is generally a nonconvex analogue to (1) with no closed-form solution. Interestingly though, closer inspection of (13) allows us to convert the resulting penalty $-\log p(Y)$ into a form directly connected to (12). More concretely, we can apply the following conversion:

Lemma 3.1 *For any $p(Y)$ expressible via (13), we have*

$$-\log p(Y) = \pi(Y;\rho) \triangleq \sum_{\{i,j\}\in\mathcal{E}} \rho\left(\left\|\boldsymbol{y}_i - \boldsymbol{y}_j\right\|_2^2\right) \qquad (14)$$

excluding irrelevant constants, where $\rho : \mathbb{R}^+ \to \mathbb{R}$ is a concave non-decreasing function that depends on μ.

Lemma 3.1 can be derived using the framework from (Palmer et al., 2006); see supplementary for details. Given that concave, non-decreasing functions dampen the impact of outliers by reducing the penalty applied to increasingly large errors (Chen et al., 2017), swapping $\pi(Y;\rho)$ into (1), i.e., as in (12), is a natural candidate for enhancing graph-aware regularization. However unlike the GNN layers derived previously using (1), it remains unclear how (12), and any subsequent propagation layers built on top of it, relate to existing GNN paradigms or reasonable extensions thereof.

To address this shortcoming, we introduce a variational decomposition of $\pi(Y;\rho)$ that allows us to define a family of strict quadratic upper bounds on (12) and ultimately, a useful link to graph attention mechanisms and robust propagation layers. For this purpose, we first define the approximate penalty function

$$\hat{\pi}(Y;\widetilde{\rho},\{\gamma_{ij}\}) \triangleq \sum_{\{i,j\}\in\mathcal{E}} \left[\gamma_{ij}\left\|\boldsymbol{y}_i-\boldsymbol{y}_j\right\|_2^2 - \widetilde{\rho}(\gamma_{ij})\right] \tag{15}$$

where $\{\gamma_{ij}\}_{i,j\in\mathcal{E}}$ denotes a set of variational weights,[2] one for each edge, and $\widetilde{\rho}$ represents the concave conjugate of ρ (Rockafellar, 1970). We then form the alternative quadratic energy

$$\begin{aligned}\hat{\ell}_Y(Y;\Gamma,\widetilde{\rho}) &\triangleq \|Y-f(X;W)\|_{\mathcal{F}}^2 + \lambda\hat{\pi}(Y;\widetilde{\rho},\{\gamma_{ij}\}) \\ &= \|Y-f(X;W)\|_{\mathcal{F}}^2 + \lambda\mathrm{tr}\left[Y^\top\hat{L}Y\right] + f(\Gamma), \end{aligned}\tag{16}$$

where $\Gamma\in\mathbb{R}^{m\times m}$ is a diagonal matrix with the variational parameter γ_{ij} corresponding with each edge arranged along the diagonal, $f(\Gamma)\triangleq-\sum_{\{i,j\}\in\mathcal{E}}\widetilde{\rho}(\gamma_{ij})$, and $\hat{L}\triangleq B^\top\Gamma B$. In this way, for any fixed set of variational parameters, (16) is equivalent to (1), excluding constant terms, only now we have the modified Laplacian $\hat{L}$ formed by weighting or attending to the edges $\mathcal{E}$ of the original graph with $\{\gamma_{ij}\}_{i,j\in\mathcal{E}}$.

The relevance of this construction then becomes apparent per the following relationships:

Lemma 3.2 *For all* $\{\gamma_{ij}\}_{i,j\in\mathcal{E}}$,

$$\hat{\ell}_Y(Y;\Gamma,\widetilde{\rho}) \;\geq\; \ell_Y(Y;\rho), \tag{17}$$

with equality[3] *iff*

$$\begin{aligned}\gamma_{ij} &= \arg\min_{\{\gamma_{ij}>0\}} \widetilde{\pi}(Y;\widetilde{\rho},\{\gamma_{ij}\}) \\ &= \left.\frac{\partial\rho(z^2)}{\partial z^2}\right|_{z=\|\boldsymbol{y}_i-\boldsymbol{y}_j\|_2}.\end{aligned}\tag{18}$$

Corollary 3.2.1 *For any* ρ*, there exists a set of attention weights* $\Gamma^*\equiv\{\gamma_{ij}^*\}_{i,j\in\mathcal{E}}$ *such that*

$$\arg\min_Y \ell_Y(Y;\rho) \;=\; \arg\min_Y \hat{\ell}_Y(Y;\Gamma^*,\widetilde{\rho}). \tag{19}$$

These results suggest that if we somehow knew the optimal attention weights, we could directly minimize $\ell_Y(Y;\rho)$ using the same efficient propagation rules we derived in Section 2.1, only now with $\hat{L}=B^\top\Gamma^*B$ replacing L. But of course Γ^* is generally not known in advance, and cannot be computed using (18) without known the optimal Y. We describe how to circumvent this issue next.

3.3. Graph Attention Layers via IRLS

In prior work involving graph attention, the attention weights are often computed using a (possibly parameterized) module designed to quantify the similarity between nodes sharing an edge, e.g., cosine distance or related (Lee et al., 2019; Thekumparampil et al., 2018). While the similarity metric may be computed in different ways, the resulting attention weights themselves are typically formed as $\gamma_{ij}=h(\boldsymbol{y}_i,\boldsymbol{y}_j)$, where h is a monotonically increasing function of similarity, i.e., more similar nodes receive higher attention weights. But in general, h is chosen heuristically, as opposed to an emergent functional form linked to an explicit energy.

In contrast, the variational perspective described herein provides us with a natural similarity metric and attention weighting scheme per Lemma 3.2. This ultimately allows us to incorporate attention anchored to an integrated energy function, one that can be directly exploited using attention layers and propagation steps combined together using IRLS.

While IRLS can be derived from many different perspectives, the core formalism is based on a majorization-minimization (Hunter & Lange, 2004) approach to optimization. In the present context, our goal is to minimize $\ell_Y(Y;\rho)$ from (12). IRLS operates by alternating between computing an upper bound $\hat{\ell}_Y(Y;\Gamma,\widetilde{\rho})\geq\ell_Y(Y;\rho)$ (the *majorization* step), with equality for at least one value of Y as shown in Lemma 3.2, and then minimizing this bound over Y (the *minimization* step).

To begin IRLS, we first initialize $Y^{(0)}=f(X;W)$. Then until convergence, after the k-th iteration we compute:

1. Update variational parameters using

$$\gamma_{ij}^{(k+1)} \;=\; \left.\frac{\partial\rho(z^2)}{\partial z^2}\right|_{z=\|\boldsymbol{y}_i-\boldsymbol{y}_j\|_2} \;\equiv\; h(\boldsymbol{y}_i,\boldsymbol{y}_j) \tag{20}$$

   for all $i,j\in\mathcal{E}$. This majorization step updates the bound $\hat{\ell}_Y(Y^{(k)};\Gamma^{(k+1)},\widetilde{\rho})$ and can be viewed as quantifying the similarity between node features $\boldsymbol{y}_i$ and $\boldsymbol{y}_j$ across all edges. Moreover, because ρ is concave and non-decreasing, the implicit weighting function h so-defined will necessarily be a *decreasing* function of $\|\boldsymbol{y}_i-\boldsymbol{y}_j\|_2$, and therefore an *increasing* function of similarity as desired.

2. Execute one (or possibly multiple) gradient steps on

[2] With some abuse of notation, we reuse γ_{ij} here as the variational parameters as they are in one-to-one correspondence with edges and function much like a precision variable.

[3] If ρ is not differentiable, then the equality holds for any γ_{ij} which is an element of the subdifferential of $-\rho(z^2)$ evaluated at $z=\|\boldsymbol{y}_i-\boldsymbol{y}_j\|_2$.

$\hat{\ell}_Y(Y^{(k)};\Gamma^{(k+1)},\widetilde{\rho})$ via

$$\begin{aligned} Y^{(k+1)} &= Y^{(k)} - \alpha \left.\frac{\partial \hat{\ell}_Y(Y;\Gamma^{(k+1)},\widetilde{\rho})}{\partial Y}\right|_{\substack{Y=Y^{(k)},\\ \Gamma=\Gamma^{(k+1)}}} \\ &= (1-\alpha)Y^{(k)} + \alpha\left(\tilde{D}^{(k+1)}\right)^{-1} \\ &\quad \times \left[\lambda A^{(k+1)}Y^{(k)} + f\left(X;W\right)\right], \end{aligned} \tag{21}$$

where the second line follows from (6), and $A^{(k+1)}$ denotes the adjacency matrix A with edges weighted by $\Gamma^{(k+1)}$; similarly for $\tilde{D}^{(k+1)}$. And if additional constraints on Y are imposed, then a proximal operator per (10) can also be included here as well.

In aggregate these two steps constitute a single IRLS iteration, with the only caveat being that, while the majorization step (*step 1*) can generally be computed exactly, the minimization step (*step 2*) is only accomplished approximately via movement along the gradient. Even so, we can still guarantee overall cost function descent:

Lemma 3.3 *Provided* $\alpha \leq \frac{1}{2}\left\|\lambda B^\top \Gamma^{(k)} B + I\right\|_2^{-1}$, *the iterations (20) and (21) are such that*

$$\ell_Y(Y^{(k)};\rho) \;\geq\; \ell_Y(Y^{(k+1)};\rho). \tag{22}$$

This result follows directly from Lemma 3.2 and properties of IRLS and gradient descent; see supplementary.

The end result is an iterative algorithm whose individual steps map directly to a rich family of attention-based GNN layers. To summarize, we need only interleave the propagation rule (21) with the attention weight computation from (20). This ultimately allows us to obtain

$$\hat{Y}(W;\rho) \;\approx\; \arg\min_Y \ell_Y(Y;\rho) \tag{23}$$

in such a way that $\partial\hat{Y}(W;\rho)/\partial W$ is actually computable via standard backpropagation. We may therefore optimize a robust analogue to (3) by executing gradient descent on

$$\ell_\rho\left(W,\theta\right) \triangleq \sum_{i=1}^{n'} \mathcal{D}\Big(g\left[\hat{\boldsymbol{y}}_i\left(W;\rho\right);\theta\right], t_i\Big) \tag{24}$$

to learn optimal parameters $\{W^*,\theta^*\}$. We refer to this model as TWIRLS, for *Together With IRLS*. The only unresolved design decision then involves the selection of ρ. In this regard, it is helpful to consider a few special cases to elucidate the link with traditional attention mechanisms.

3.4. Attention Special Cases

The function ρ uniquely determines how the attention weights scale with the inverse similarity metric $z_{ij} = \left\|\boldsymbol{y}_i - \boldsymbol{y}_j\right\|_2$. As a first example, let $\rho\left(z_{ij}^2\right) = \sqrt{z_{ij}^2} = |z_{ij}|$, from which it follows using (18) that $\gamma_{ij} = \frac{1}{2|z_{ij}|}$ is optimal. Consequently, based on (20) we have that

$$\gamma_{ij}^{(k+1)} \;=\; \frac{1}{2}\left(\left\|\boldsymbol{y}_i^{(k)} - \boldsymbol{y}_j^{(k)}\right\|_2\right)^{-1}. \tag{25}$$

This result implies that if the node embeddings $\boldsymbol{y}_i^{(k)}$ and $\boldsymbol{y}_j^{(k)}$ are very close together, the corresponding value of $\gamma_{ij}^{(k+1)}$ will become large, just as typical attention weights are larger for similar nodes. Conversely, if $\boldsymbol{y}_i^{(k)}$ and $\boldsymbol{y}_j^{(k)}$ are quite different, then $\gamma_{ij}^{(k+1)}$ will become small, and the resulting quadratic factor in $\hat{\ell}_Y(Y;\Gamma,\widetilde{\rho})$ will be substantially downweighted, implying that the corresponding edge could be spurious.

Unfortunately though, (25) is unbounded from above, and if at any time during training the embeddings along an edge satisfy $\boldsymbol{y}_i \approx \boldsymbol{y}_j$, we have $\gamma_{ij} \to \infty$. To avoid this situation, we may instead select ρ from more restricted function classes with bounded gradients. Table 1 shows several representative examples along with the corresponding attention weights including upper and lower bounds.

Table 1. Well-behaved robust penalties and their resulting gradients/attention weights. $p \leq 2$, ϵ, and τ are all non-negative constants. *Range* refers to the allowable attention weights.

$\rho\left(z^2\right)$	$\frac{\partial\rho(z^2)}{\partial z^2}$	Range
$\log\left(z^2+\epsilon\right)$	$\frac{1}{z^2+\epsilon}$	$\left(0, \frac{1}{\epsilon}\right]$
$\begin{array}{ll} z^2 & z<\tau \\ \tau & z\geq\tau \end{array}$	$\begin{array}{ll} 1 & z<\tau \\ 0 & z\geq\tau \end{array}$	$\{0,1\}$
$\begin{array}{ll} z^2 & z<\tau \\ \tau^{(2-p)}z^p & z\geq\tau \end{array}$	$\begin{array}{ll} 1 & z<\tau \\ \frac{p}{2}\tau^{(2-p)}z^{(p-2)} & z\geq\tau \end{array}$	$(0,1]$

In general though, there is considerable flexibility in the choice of ρ, with different choices leading to different flavors of attention. And because of the variational link to an explicit energy function associated with different selections, we can directly anticipate how they are likely to behave in various situations of interest. Note also that in certain cases we may first define a plausible attention update and then work backwards to determine the form of ρ that would lead to such an update.

For example, the cosine distance is often used as the basis for computing attention weights (Lee et al., 2019; Zhang & Zitnik, 2020; Thekumparampil et al., 2018). However, if we assume normalized node embeddings $\|\boldsymbol{y}_i\|_2 = 1$ for all i, then $\cos\angle\left(\boldsymbol{y}_i,\boldsymbol{y}_j\right) = \left(1 - \frac{1}{2}\left\|\boldsymbol{y}_i - \boldsymbol{y}_j\right\|_2^2\right)$, which is a decreasing function of $z_{ij} = \left\|\boldsymbol{y}_i - \boldsymbol{y}_j\right\|_2$ just as the candidate weights produced by TWIRLS will be per Lemmas

3.1 and 3.2, and as exemplified by the middle column of Table 1. Hence cosine distance could be implemented within TWIRLS, with the requisite unit-norm constraint handled via the appropriate proximal operator.

And as a final point of comparison, thresholding attention weights between nodes that are sufficiently dissimilar has been proposed as a simple heuristic to increase robustness (Zhang & Zitnik, 2020; Wu et al., 2019b). This can also be incorporated with TWIRLS per the second row of Table 1, and hence be motivated within our integrated framework.

4. Related Work

Message passing GNNs have been previously designed to handle many of the issues described in Section 1, although not generally within a unified framework transparently structured to handle them all. And in most cases, experiments are limited to the more narrow motivating regime that guided the original design process. For example, methods developed to reliably extend network depth are typically tested on standard benchmarks with increasing propagation layers (Chen et al., 2020; Li et al., 2020; Rong et al., 2020; Xu et al., 2018); models that address graph heterophily are tested on datasets with dissimilar labels/features sharing edges (Zhu et al., 2020b;a; Pei et al., 2019); adversarially-robust designs are deployed against targeted edge or feature purturbations (Zhang & Zitnik, 2020; Wu et al., 2019b; Zhu et al., 2019); and efforts to model long-range dependencies apply graph data where node labels may be sparse (Gu et al., 2020; Dai et al., 2018). In contrast, we empirically compare TWIRLS across all of these areas in Section 5. And while some models can be traced back to an underlying energy function (Klicpera et al., 2019; Jia & Benson, 2020), or propagation layers related to gradient descent (Zhou et al., 2004), TWIRLS is the only architecture we are aware of whereby all components, including propagation layers, nonlinear activations, and attention all explicitly follow from iterations that provably descend a principled objective.

Beyond the above, after our original submission of this work we have also recently become aware of four contemporary references (Ma et al., 2020; Pan et al., 2021; Zhang et al., 2020; Zhu et al., 2021) whose foundation, like ours, can be traced back to (Zhou et al., 2004) and the corresponding descent of a quadratic energy as in (1). But although superficially similar in terms of a common starting point, the subsequent analysis in each case is fundamentally different than our derivations and theory from Sections 2 and 3 herein. In brief, (Ma et al., 2020; Pan et al., 2021; Zhang et al., 2020; Zhu et al., 2021) are all focused, at least to some degree, on providing a unified framework for understanding various existing GNN paradigms, whereas we are orthogonally motivated by the robust design of a single new model formed by the merger of proximal gradient descent and IRLS.

Note also that while the inclusion of attention is loosely discussed in (Ma et al., 2020; Pan et al., 2021; Zhu et al., 2021), none of these works actually account for a fully integrated attention mechanism whereby the attention weights themselves are adaptively computed at each layer on the basis of an overarching energy. Instead, attention weights are either included as a heuristic add-on to a standard quadratic energy (Pan et al., 2021), or else fixed to a function of the input features X with no adaptation across layers (Ma et al., 2020). In contrast, the layer-wise adaptation of TWIRLS attention weights from Section 3 explicitly correspond with descent of the global energy from (12). Similarly, none of these prior works incorporate nonlinear activations as proximal operators as we have done. Even so, (Ma et al., 2020; Pan et al., 2021; Zhang et al., 2020; Zhu et al., 2021) all provide a useful perspective that is complementary to our own, and in future work it could be beneficial to combine the prescriptions from each. For example, more refined local neighborhood smoothing, as proposed in (Pan et al., 2021) based on the similarity of initial input features (as opposed to layer-wise attention), could naturally be incorporated within TWIRLS.

And finally, although the context and motivation is different from our work, GNN architectures have also been previously developed to imitate the individual steps of certain types of graph algorithms (Veličković et al., 2020). The latter include breadth-first search, Bellman-Ford, and Prim's algorithm.

5. Experiments

We test the performance of TWIRLS under different application scenarios, in each case comparing against SOTA methods specifically designed for the particular task at hand. Across all experiments, we choose either $f\left(X;W\right)$ or $g\left(\boldsymbol{y};\theta\right)$ as a shallow MLP (or linear layer), while the other is a simple identity mapping with no parameters. Hence *all* trainable parameters are restricted to a single module, and TWIRLS operates as an extremely simple architecture. For the attention, we select ρ as a truncated ℓ_p quasinorm (Wilansky, 2013), and when no attention is included, we refer to the model as $\text{TWIRLS}_{\text{base}}$. For full details regarding the propagation steps, attention formulation, and other hyperparameters, please see the supplementary; similarly for experimental details, additional ablation studies and empirical results, and a discussion of computational complexity. All models and experiments were implemented using the Deep Graph Library (DGL) (Wang et al., 2019).

5.1. Base Model Results with No Attention

We first evaluate $\text{TWIRLS}_{\text{base}}$ on three commonly used citation datasets, namely Cora, Pubmed and Citeseer (Sen et al.,

Table 2. Baseline results on standard benchmarks.

MODEL	CORA	CITESEER	PUBMED	ARXIV
SGC	81.7 ± 0.1	71.3 ± 0.2	78.9 ± 0.1	69.79 ± 0.16
GCN	81.5	71.1	79.0	71.74 ± 0.29
APPNP	83.3	71.8	80.1	71.74 ± 0.29
JKNET	81.1	69.8	78.1	72.19 ± 0.21
GCNII	**85.5 ± 0.5**	73.4 ± 0.6	80.3 ± 0.4	72.74 ± 0.16
DAGNN	84.4 ± 0.5	73.3 ± 0.6	80.5 ± 0.5	72.09 ± 0.25
$\text{TWIRLS}_{\text{BASE}}$	84.1 ± 0.5	**74.2 ± 0.45**	**80.7 ± 0.5**	**72.93 ± 0.19**

2008), plus ogbn-arxiv (Hu et al., 2020), a relatively large dataset. For the citation datasets, we use the semi-supervised setting from (Yang et al., 2016), while for ogbn-arxiv, we use the standard split from the Open Graph Benchmark (OGB) leaderboard (see `https://ogb.stanford.edu`). We compare $\text{TWIRLS}_{\text{base}}$ against the top models from the ogbn-arxiv leaderboard restricted to those with papers and architecture contributions, i.e., we exclude unpublished entries that rely on training heuristics such as adding labels as features, recycling label predictions, non-standard losses/optimization methods, etc. The latter, while practically useful, can be applied to all models to further boost performance, and are beyond the scope of this work. Given these criteria, in Table 2 we report results for JKNet (Xu et al., 2018), GCNII (Chen et al., 2020), and DAGNN (Liu et al., 2020). We also include three common baseline models, GCN (Kipf & Welling, 2017a), SGC (Wu et al., 2019a) and APPNP (Klicpera et al., 2019).

We also conduct experiments showing the behavior of $\text{TWIRLS}_{\text{base}}$ as the number of propagation step becomes arbitrarily large. The results on Cora data are plotted in Figure 1, where we observe that the performance of $\text{TWIRLS}_{\text{base}}$ converges to the analytical solution given by (2) without performance degradation from oversmoothing. In contrast, the accuracy of SGC drops considerably with propagation.

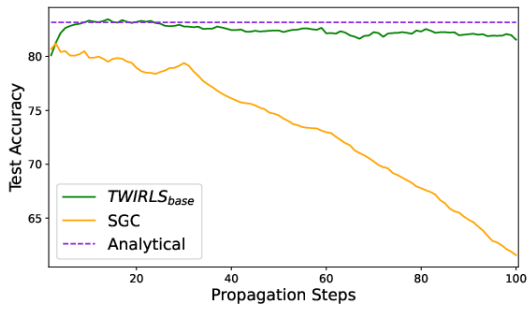

Figure 1. Accuracy versus propagation steps on Cora.

5.2. Adversarial Attack Results

To showcase robustness against edge uncertainty using the proposed IRLS-based attention, we next test the full TWIRLS model using graph data attacked via the the Mettack algorithm (Zügner & Günnemann, 2019). Mettack operates by perturbing graph edges with the aim of maximally reducing the node classification accuracy of a surrogate GNN model that is amenable to adversarial attack optimization. And the design is such that this reduction is generally transferable to other GNN models trained with the same perturbed graph. In terms of experimental design, we follow the exact same non-targeted attack scenario from (Zhang & Zitnik, 2020), setting the edge perturbation rate to 20%, adopting the 'Meta-Self' training strategy, and a GCN as the surrogate model.

For baselines, we use three state-of-the-art defense models, namely GNNGuard (Zhang & Zitnik, 2020), GNN-Jaccard (Wu et al., 2019b) and GNN-SVD (Entezari et al., 2020). Results on Cora and Citeseer data are reported in Table 4, where the inclusion of attention provides a considerable boost in performance over $\text{TWIRLS}_{\text{base}}$. And somewhat surprisingly, TWIRLS generally performs comparably or better than existing SOTA models that were meticulously designed to defend against adversarial attacks.

5.3. Results on Heterophily Graphs

We also consider non-homophily (or heterophily) graphs to further validate the effectiveness of our IRLS-based attention layers in dealing with inconsistent edges. In this regard, the homophily level of a graph can be quantified via the homophily ratio $\mathcal{H} = \frac{|\{(u,v) : (u,v)\in\mathcal{E} \wedge t_u = t_v\}|}{|\mathcal{E}|}$ from (Zhu et al., 2020b), where t_u and t_v are the target labels of nodes u and v. $\mathcal{H}$ quantifies the tendency of nodes to be connected with other nodes from the same class. Graphs with an $\mathcal{H} \approx 1$ exhibit strong homophily, while conversely, those with $\mathcal{H} \approx 0$ show strong heterophily, indicating that many edges are connecting nodes with different labels.

We select four graph datasets with a low homophily ratio, namely, Actor, Cornell, Texas, and Wisconsin, adopting the data split, processed node features, and labels provided by (Pei et al., 2019). We compare the average node classification accuracy of our model with GCN (Kipf & Welling, 2017b), GAT (Velickovic et al., 2018), GraphSAGE (Hamilton et al., 2017), SOTA heterophily methods GEOM-GCN (Pei et al., 2019) and H_2GCN (Zhu et al., 2020b), as well as a baseline MLP as reported in (Zhu et al., 2020b).

Results are presented in Table 3, paired with the homophily ratios of each dataset. Here we observe that the proposed IRLS-based attention helps TWIRLS generally perform better than $\text{TWIRLS}_{\text{base}}$ and existing methods on heterophily graphs, achieving SOTA accuracy on three of the four datasets, and nearly so on the forth.

DATASET	TEXAS	WISCONSIN	ACTOR	CORNELL
HOM. RATIO ($\mathcal{H}$)	0.11	0.21	0.22	0.3
GCN	59.46±5.25	59.80±6.99	30.26±0.79	57.03±4.67
GAT	58.38±4.45	55.29±8.71	26.28±1.73	58.92±3.32
GRAPHSAGE	82.43±6.14	81.18±5.56	34.23±0.99	75.95±5.01
GEOM-GCN	67.57	64.12	31.63	60.81
H_2GCN	**84.86±6.77**	86.67±4.69	35.86±1.03	82.16±4.80
MLP	81.89±4.78	85.29±3.61	35.76±0.98	81.08±6.37
TWIRLS$_{\text{BASE}}$	81.62±5.51	82.75±7.83	37.10±1.07	83.51±7.30
TWIRLS	84.59±3.83	**86.67±4.19**	**37.43±1.50**	**86.76±5.05**

Table 3. Result on heterophily graphs.

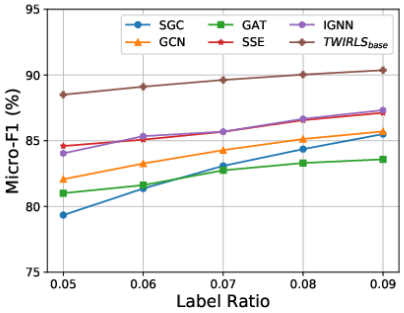

Figure 2. Amazon co-purchase results.

Table 4. Adversarial attack comparison with SOTA models.

MODEL	ATK-CORA	ATK-CITESEER
SURROGATE (GCN)	57.38 ± 1.42	60.42 ± 1.48
GNNGUARD	**70.46 ± 1.03**	65.20 ± 1.84
GNN-JACCARD	64.51 ± 1.35	63.38 ± 1.31
GNN-SVD	66.45 ± 0.76	65.34 ± 1.00
TWIRLS$_{\text{BASE}}$	62.89 ± 1.59	63.83 ± 1.95
TWIRLS	70.23 ± 1.09	**70.63 ± 0.93**

5.4. Results on Long-Dependency Data

Finally we test the ability of our model to capture long-range dependencies in graph data by stably introducing an arbitrary number of propagation layers without the risk of oversmoothing. For this purpose, we apply the Amazon Co-Purchase dataset, a common benchmark used for testing long-range dependencies given the sparse labels relative to graph size (Dai et al., 2018; Gu et al., 2020). We adopt the test setup from (Dai et al., 2018; Gu et al., 2020), and compare performance using different label ratios. We include five baselines, namely SGC (Chen et al., 2020), GCN (Kipf & Welling, 2017a), GAT (Velickovic et al., 2018), SSE (Dai et al., 2018) and IGNN (Gu et al., 2020). Note that SSE and IGNN are explicitly designed for capturing long-range dependencies. Even so, our model is able to outperform both of them by a significant margin.

6. Conclusion

In this work we have derived TWIRLS, a simple integrated framework for combining propagation and attention layers anchored to the iterative descent of an objective function that is robust against edge uncertainty and oversmoothing. And despite its generic underpinnings, TWIRLS can match or exceed the performance of many existing architectures, including domain-specific SOTA models tailored to handle adversarial attacks, heterophily, or long-range dependencies.

Acknowledgements

Zengfeng Huang is supported by National Natural Science Foundation of China No. 61802069, Shanghai Sailing Program No. 18YF1401200, Shanghai Science and Technology Commission No. 17JC1420200. Zhewei Wei is supported by NSFC No. 61972401, No. 61932001, No. 61832017.

References

Andrews, D. and Mallows, C. Scale mixtures of normal distributions. *J. Royal Statistical Society: Series B*, 36(1), 1974.

Axelsson, O. *Iterative Solution Methods*. Cambridge University Press, 1996.

Chartrand, R. and Yin, W. Iteratively reweighted algorithms for compressive sensing. *International Conference on Accoustics, Speech, and Signal Processing*, 2008.

Chen, J., Ma, T., and Xiao, C. FastGCN: Fast learning with graph convolutional networks via importance sampling. *International Conference on Learning Representations*, 2018.

Chen, M., Wei, Z., Huang, Z., Ding, B., and Li, Y. Simple and deep graph convolutional networks. In *Proceedings of the 37th International Conference on Machine Learning, ICML*, volume 119, pp. 1725–1735, 2020.

Chen, Y., Ge, D., Wang, M., Wang, Z., Ye, Y., and Yin, H. Strong NP-hardness for sparse optimization with concave penalty functions. In *International Confernece on Machine Learning*, 2017.

Combettes, P. L. and Pesquet, J.-C. Proximal splitting methods in signal processing. In *Fixed-point algorithms for inverse problems in science and engineering*, pp. 185–212. Springer, 2011.

Dai, H., Kozareva, Z., Dai, B., Smola, A., and Song, L. Learning steady-states of iterative algorithms over graphs.

In *International conference on machine learning*, pp. 1106–1114, 2018.

Daubechies, I., DeVore, R., Fornasier, M., and Güntürk, C. S. Iteratively reweighted least squares minimization for sparse recovery. *Communications on Pure and Applied Mathematics*, 63(1), 2010.

Entezari, N., Al-Sayouri, S. A., Darvishzadeh, A., and Papalexakis, E. E. All you need is low (rank) defending against adversarial attacks on graphs. In *Proceedings of the 13th International Conference on Web Search and Data Mining*, pp. 169–177, 2020.

Gregor, K. and LeCun, Y. Learning fast approximations of sparse coding. In *International Conference on Machine Learning*, 2010.

Gu, F., Chang, H., Zhu, W., Sojoudi, S., and Ghaoui, L. E. Implicit graph neural networks. In *Advances in Neural Information Processing Systems, NeurIPS*, 2020.

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

Hershey, J., Roux, J. L., and Weninger, F. Deep unfolding: Model-based inspiration of novel deep architectures. *arXiv preprint arXiv:1409.2574*, 2014.

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. 2020.

Hunter, D. R. and Lange, K. A tutorial on MM algorithms. *The American Statistician*, 58(1), 2004.

Jia, J. and Benson, A. R. Residual correlation in graph neural network regression. In *International Conference on Knowledge Discovery & Data Mining*, 2020.

Kearnes, S. M., McCloskey, K., Berndl, M., Pande, V. S., and Riley, P. Molecular graph convolutions: moving beyond fingerprints. *J. Comput. Aided Mol. Des.*, 30(8): 595–608, 2016.

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

Kipf, T. N. and Welling, M. Semi-supervised classification with graph convolutional networks. In *5th International Conference on Learning Representations, ICLR*, 2017b.

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

Lee, J. B., Rossi, R., Kim, S., Ahmed, N., and Koh, E. Attention models in graphs: A survey. *ACM Trans. Knowledge Discovery from Data*, 13(6), 2019.

Li, G., Muller, M., Thabet, A., and Ghanem, B. Deepgcns: Can gcns go as deep as cnns? In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 9267–9276, 2019.

Li, G., Xiong, C., Thabet, A., and Ghanem, B. Deepergcn: All you need to train deeper gcns. *arXiv preprint arXiv:2006.07739*, 2020.

Li, Q., Han, Z., and Wu, X.-M. Deeper insights into graph convolutional networks for semi-supervised learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 32, 2018.

Liu, M., Gao, H., and Ji, S. Towards deeper graph neural networks. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pp. 338–348, 2020.

Ma, Y., Liu, X., Zhao, T., Liu, Y., Tang, J., and Shah, N. A unified view on graph neural networks as graph signal denoising. *arXiv preprint arXiv:2010.01777*, 2020.

Nocedal, J. and Wright, S. *Numerical optimization*. Springer Science & Business Media, 2006.

Oono, K. and Suzuki, T. Graph neural networks exponentially lose expressive power for node classification. In *8th International Conference on Learning Representations, ICLR*, 2020.

Palmer, J., Wipf, D., Kreutz-Delgado, K., and Rao, B. Variational EM algorithms for non-Gaussian latent variable models. *Advances in Neural Information Processing Systems*, 2006.

Pan, X., Song, S., and Huang, G. A unified framework for convolution-based graph neural networks, 2021. URL https://openreview.net/forum?id=zUMD--Fb9Bt.

Pei, H., Wei, B., Chang, K. C.-C., Lei, Y., and Yang, B. Geom-gcn: Geometric graph convolutional networks. In *International Conference on Learning Representations*, 2019.

Rockafellar, R. T. *Convex Analysis*. Princeton University Press, 1970.

Rong, Y., Huang, W., Xu, T., and Huang, J. Dropedge: Towards deep graph convolutional networks on node classification. In *8th International Conference on Learning Representations, ICLR*, 2020.

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.

Sprechmann, P., Bronstein, A., and Sapiro, G. Learning efficient sparse and low rank models. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 37(9), 2015.

Thekumparampil, K., Wang, C., Oh, S., and Li, L.-J. Attention-based graph neural network for semi-supervised learning. *arXiv preprint arXiv:1803.03735*, 2018.

Velickovic, P., Cucurull, G., Casanova, A., Romero, A., Liò, P., and Bengio, Y. Graph attention networks. In *6th International Conference on Learning Representations, ICLR*, 2018.

Veličković, P., Ying, R., Padovano, M., Hadsell, R., and Blundell, C. Neural execution of graph algorithms. In *International Conference on Learning Representations*, 2020.

Wang, M., Zheng, D., Ye, Z., Gan, Q., Li, M., Song, X., Zhou, J., Ma, C., Yu, L., Gai, Y., Xiao, T., He, T., Karypis, G., Li, J., and Zhang, Z. Deep graph library: A graph-centric, highly-performant package for graph neural networks. *arXiv preprint arXiv:1909.01315*, 2019.

Wang, Z., Ling, Q., and Huang, T. Learning deep ℓ_0 encoders. In *AAAI Conference on Artificial Intelligence*, volume 30, 2016.

West, M. Outlier models and prior distributions in Bayesian linear regression. *J. Royal Statistical Society: Series B*, 46(3), 1984.

Wilansky, A. *Modern methods in topological vector spaces*. Dover Publications, Inc., 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, 2019a.

Wu, H., Wang, C., Tyshetskiy, Y., Docherty, A., Lu, K., and Zhu, L. Adversarial examples for graph data: Deep insights into attack and defense. In *Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence, IJCAI*, pp. 4816–4823, 2019b.

Xu, K., Li, C., Tian, Y., Sonobe, T., Kawarabayashi, K., and Jegelka, S. Representation learning on graphs with jumping knowledge networks. In *Proceedings of the 35th International Conference on Machine Learning, ICML*, volume 80, pp. 5449–5458, 2018.

Yang, Y., Liu, T., Wang, Y., Zhou, J., Gan, Q., Wei, Z., Zhang, Z., Huang, Z., and Wipf, D. Graph neural networks inspired by classical iterative algorithms. *CoRR*, abs/2103.06064, 2021. URL https://arxiv.org/abs/2103.06064.

Yang, Z., Cohen, W., and Salakhudinov, R. Revisiting semi-supervised learning with graph embeddings. In *International conference on machine learning*, pp. 40–48, 2016.

Zhang, H., Yan, T., Xie, Z., Xia, Y., and Zhang, Y. Revisiting graph convolutional network on semi-supervised node classification from an optimization perspective. *CoRR*, abs/2009.11469, 2020.

Zhang, X. and Zitnik, M. Gnnguard: Defending graph neural networks against adversarial attacks. *Advances in Neural Information Processing Systems*, 33, 2020.

Zhou, D., Bousquet, O., Lal, T. N., Weston, J., and Schölkopf, B. Learning with local and global consistency. *Advances in Neural Information Processing Systems*, 2004.

Zhou, J., Cui, G., Zhang, Z., Yang, C., Liu, Z., Wang, L., Li, C., and Sun, M. Graph neural networks: A review of methods and applications. *arXiv preprint arXiv:1812.08434*, 2018.

Zhu, D., Zhang, Z., Cui, P., and Zhu, W. Robust graph convolutional networks against adversarial attacks. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pp. 1399–1407, 2019.

Zhu, J., Rossi, R. A., Rao, A., Mai, T., Lipka, N., Ahmed, N. K., and Koutra, D. Graph neural networks with heterophily. *arXiv preprint arXiv:2009.13566*, 2020a.

Zhu, J., Yan, Y., Zhao, L., Heimann, M., Akoglu, L., and Koutra, D. Beyond homophily in graph neural networks: Current limitations and effective designs. In *Advances in Neural Information Processing Systems, NeurIPS*, 2020b.

Zhu, M., Wang, X., Shi, C., Ji, H., and Cui, P. Interpreting and unifying graph neural networks with an optimization framework. *arXiv preprint arXiv:2101.11859*, 2021.

Zügner, D. and Günnemann, S. Adversarial attacks on graph neural networks via meta learning. In *7th International Conference on Learning Representations, ICLR*, 2019.

Zügner, D., Akbarnejad, A., and Günnemann, S. Adversarial attacks on neural networks for graph data. In *Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence, IJCAI*, 2019.