Implicit vs Unfolded Graph Neural Networks

Yongyi Yang　　　　　　　　　　　　　　　　　　　　　　YONGYI@UMICH.EDU
University of Michigan
Ann Arbor, United States

Tang Liu　　　　　　　　　　　　　　　　　　　　　　CNLIUTANG@GMAIL.COM
Alibaba
Hangzhou, China

Yangkun Wang　　　　　　　　　　　　　　　　　ESPYLAPIZA@GMAIL.COM
University of California, San Diego
La Jolla, United States

Zengfeng Huang　　　　　　　　　　　　　　　HUANGZF@FUDAN.EDU.CN
Fudan University
Shanghai, China

David Wipf　　　　　　　　　　　　　　　　　　DAVIDWIPF@GMAIL.COM
Amazon Web Services
Shanghai, China

Editor: Pradeep Ravikumar

Abstract

It has been observed that message-passing graph neural networks (GNN) sometimes struggle to maintain a healthy balance between the efficient/scalable modeling of long-range dependencies across nodes while avoiding unintended consequences such oversmoothed node representations, sensitivity to spurious edges, or inadequate model interpretability. To address these and other issues, two separate strategies have recently been proposed, namely *implicit* and *unfolded* GNNs (that we abbreviate to IGNN and UGNN respectively). The former treats node representations as the fixed points of a deep equilibrium model that can efficiently facilitate arbitrary implicit propagation across the graph with a fixed memory footprint. In contrast, the latter involves treating graph propagation as unfolded descent iterations as applied to some graph-regularized energy function. While motivated differently, in this paper we carefully quantify explicit situations where the solutions they produce are equivalent and others where their properties sharply diverge. This includes the analysis of convergence, representational capacity, and interpretability. In support of this analysis, we also provide empirical head-to-head comparisons across multiple synthetic and public real-world node classification benchmarks. These results indicate that while IGNN is substantially more memory-efficient, UGNN models support unique, integrated graph attention mechanisms and propagation rules that can achieve strong node classification accuracy across disparate regimes such as adversarially-perturbed graphs, graphs with heterophily, and graphs involving long-range dependencies.

Keywords: Graph Neural Networks, Graph-Regularized Energy Functions, Algorithm Unfolding, Bilevel Optimization, Implicit Deep Learning.

1. Introduction

Given graph data with node features, graph neural networks (GNNs) represent an effective way of exploiting relationships among these features to predict labeled quantities of interest, e.g., node classification (Wu et al., 2020; Zhou et al., 2018). In particular, each layer of a message-passing GNN (our focus) is constructed by bundling a graph propagation step with an aggregation function such that information can be shared between neighboring nodes to an extent determined by network depth. (Kipf and Welling, 2017; Hamilton et al., 2017b; Kearnes et al., 2016; Velickovic et al., 2018).

For sparsely-labeled graphs, or graphs with entity relations reflecting long-range dependencies, it is often desirable to propagate signals arbitrary distances across nodes, but without expensive memory requirements during training, oversmoothing effects (Oono and Suzuki, 2020; Li et al., 2018), or disproportionate sensitivity to bad/spurious edges (Zhu et al., 2020b,a; Zügner et al., 2019; Zügner and Günnemann, 2019). Moreover, to the extent possible we would like such propagation to follow interpretable patterns that can be adjusted in predictable ways to problem-specific contexts (e.g., graphs possessing heterophily). To address these issues, at least in part, two distinct strategies have recently been proposed that will form the basis of our study.[1] First, the framework of implicit deep learning (Bai et al., 2019; El Ghaoui et al., 2020) has been applied to producing supervised node embeddings that satisfy an equilibrium criteria instantiated through a graph-dependent fixed-point equation. The resulting so-called *implicit* GNN (IGNN) pipeline (Gu et al., 2020), as well as related antecedents (Dai et al., 2018; Gallicchio and Micheli, 2020) and descendants (Baker et al., 2023; Fu et al., 2023; Liu et al., 2021a), mimic the behavior of a GNN model with arbitrary depth for handling long-range dependencies, but is nonetheless trainable via a fixed memory budget and robust against oversmoothing effects. See Figure 1 for a high-level conceptual depiction of IGNNs.

Secondly, *unfolded* GNN (UGNN) architectures are formed via graph propagation layers patterned after the unfolded descent iterations of some graph-regularized energy function (Chen and Eldar, 2021; Di Giovanni et al., 2023; Liu et al., 2021b; Ma et al., 2020; Pan et al., 2021; Xue et al., 2023; Yang et al., 2021; Zhang et al., 2020; Zhu et al., 2021); Figure 1 contains an illustration. In this way, the node embeddings at each UGNN layer can be viewed as increasingly refined approximations to an interpretable energy minimizer, which can be nested within a bilevel optimization framework (Wang et al., 2016) for supervised training akin to IGNN. More broadly, similar unfolding strategies have previously been adopted for designing a variety of new deep architectures or interpreting existing ones (Geng et al., 2021; Gregor and LeCun, 2010; He et al., 2017; Hershey et al., 2014; Sprechmann et al., 2015; Veličković et al., 2020; Yang et al., 2022).

As such, IGNN and UGNN are closely related in the sense that their node embeddings are intentionally aligned with a meta-criterion: either a fixed-point for IGNN or an energy minimizer for UGNN, where in both cases a useful inductive bias is introduced to address similar desiderata. And yet despite these commonalities, there has thus far been no systematic examination of the meaningful similarities and differences. We take key steps in this direction via the following workflow. First, after introducing the basic setup and notation in Section 2, we overview the IGNN framework in Section 3, including its attractive reliance on memory-efficient implicit differentiation and existing convergence results. Next, Section 4 introduces a general-purpose UGNN framework that encompasses a variety of existing unfolded models as special cases, including models with graph attention, broad graph

---

1. There are of course many more than just two approaches for tackling these types of issues, e.g., random-walk GNNs, cooperative GNNs, graph rewiring, graph Transformers (Finkelshtein et al., 2024; Karhadkar et al., 2023; Tönshoff et al., 2023; Topping et al., 2022; Wu et al., 2023); however, a broader treatment of such methodology remains outside of our primary scope herein.
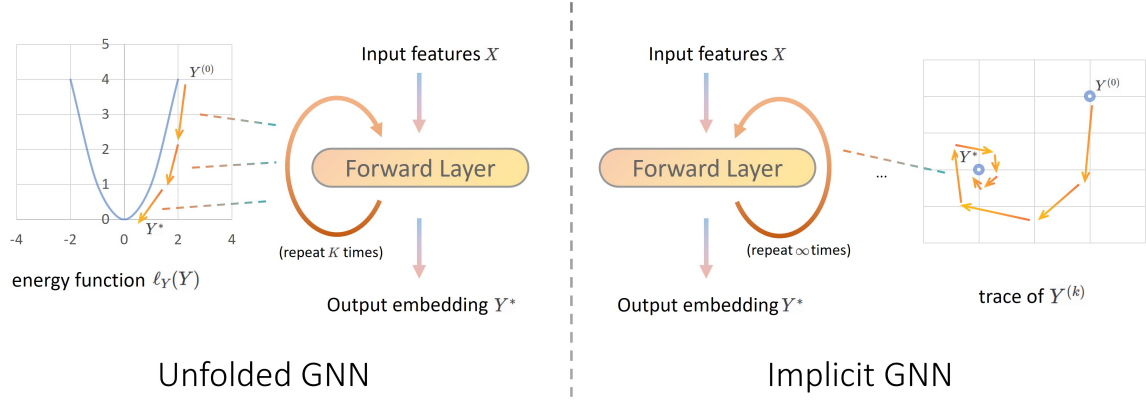
Figure 1: Illustration of UGNNs and IGNNs. The lefthand side displays IGNN forward pass layers, each of which reduces the *distance* between the current node embedding $Y^{(k)}$ and a fixed point Y^*. contrast, the righthand side reveals that, given input features X, each UGNN forward pass layer induces node embeddings Y that form a *descent step* of an energy function ℓ_Y.

propagation operators, and nonlinear activation functions. Sections 5–6 provide comparative analysis of the relative strengths and weaknesses of IGNN and UGNN models with respect to convergence guarantees and model expressiveness. We conclude with empirical comparisons in Section 7 and scalability considerations in Section 8, while proofs and supplementary details are deferred to the Appendix. Overall, our contributions can be summarized as follows:

- Although the original conceptions are unique, we consider a sufficiently broad design space of IGNN and UGNN models such that practically-relevant, interpretable regions of exact overlap can be established, analyzed, and contrasted with key areas of nonconformity.

- Within this framework, we compare the relative ability of IGNNs and UGNNs to converge to optimal (or locally-optimal) solutions per various model-specific design criteria. Among other things, this analysis reveals that IGNNs enjoy an advantage in that (unlike UGNNs) their implicit layers do not unavoidably induce symmetric propagation weights. In contrast, we show that UGNNs are more flexible by accommodating a broader class of robust nonlinear graph propagation operators while still guaranteeing at least local convergence. This capability is relevant to handling graphs with certain types of heterophily (meaning nodes sharing an edge may not share similar labels and/or features) and adversarial perturbations.

- We investigate the consequences of symmetric (layer-tied) UGNN propagation weights (an issue that has been raised but remains unresolved in the literature). In particular, we prove that with linear activations, a UGNN can reproduce any IGNN representation, and define sufficient conditions for equivalency in broader regimes. We also show that UGNN layers with symmetric, layer-tied weights can exactly mimic arbitrary graph convolutional network (GCN) models (Kipf and Welling, 2017) characterized by asymmetric, layer-specific weights without introducing any additional parameters or significant complexity. Collectively, these results suggest that the weight symmetry enforced by UGNN models may not be overly restrictive in practice.

- Empirically, we provide comprehensive, head-to-head comparisons between equivalently-sized IGNN and UGNN models while evaluating computational complexity and memory costs. These results also serve to complement our analytical findings while showcasing node classification regimes whereby IGNN or UGNN can achieve near-SOTA performance. We also demonstrate how graph heterophily can manifest in fundamentally different ways that UGNN models in particular are naturally equipped to handle.

2. Basic Setup

Consider a graph $\mathcal{G} = \{\mathcal{V}, \mathcal{E}\}$, with $n = |\mathcal{V}|$ nodes and $m = |\mathcal{E}|$ edges. We define $L \in \mathbb{R}^{n \times n}$ as 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. We also let $\widetilde{A} = A + I$ (i.e., A with self loops) and denote $\widetilde{D}$ as the corresponding degree matrix.

Both IGNNs and UGNNs incorporate graph structure via optimized embeddings $Y^* \in \mathbb{R}^{n \times d}$ that are a function of some adjustable weights $\mathcal{W}$, i.e., $Y^* \equiv Y^*(\mathcal{W})$ (for simplicity of notation, we will frequently omit including this dependency on $\mathcal{W}$), where by design $\partial Y^*(\mathcal{W})/\partial \mathcal{W}$ is computable, either implicitly (IGNN) or explicitly (UGNN). We may then insert these embeddings within an application-specific meta-loss given by

$$\ell_\theta (\theta, \mathcal{W}) \triangleq \sum_{i=1}^{n'} \mathcal{D}\big(g\left[\boldsymbol{y}_i^*(\mathcal{W}); \theta\right], \boldsymbol{t}_i\big), \tag{1}$$

where $g : \mathbb{R}^d \rightarrow \mathbb{R}^c$ is some differentiable node-wise function with parameters θ and c-dimensional output tasked with predicting ground-truth node-wise targets $\boldsymbol{t}_i \in \mathbb{R}^c$. Additionally, $\boldsymbol{y}_i^*(\mathcal{W})$ is the i-th row of $Y^*(\mathcal{W})$, $n' < n$ is the number of labeled nodes (we assume w.l.o.g. that the first n' nodes are labeled), and $\mathcal{D}$ is a discriminator function, e.g., cross-entropy for classification, squared error for regression, etc. Given that $Y^*(\mathcal{W})$ is differentiable by construction, we can optimize $\ell_\theta (\theta, \mathcal{W})$ via gradient descent to obtain our final predictive model.

At a conceptual level then, the only difference between IGNNs and UGNNs is in how the corresponding optimal embeddings $Y^*(\mathcal{W})$ are motivated and computed. We introduce the specifics of each option in the following two sections.

3. Overview of Implicit GNNs

IGNN models are predicated on the fixed-point update

$$Y^{(k+1)} = \sigma \left[P Y^{(k)} W_p + f(X; W_x) \right], \tag{2}$$

where $Y^{(k)}$ are node embeddings after the k-th iteration, σ is a nonlinear activation function, $\mathcal{W} = \{W_p, W_x\}$ is a set of two weight matrices, and $P \in \mathbb{R}^{n \times n}$ is an arbitrary graph propagation operator, e.g., $P = A$; other variants are also obtainable using, for example, graph rewiring (Gasteiger et al., 2019). Additionally, $f : \mathbb{R}^{n \times d_0} \rightarrow \mathbb{R}^{n \times d}$ is a base predictor function that converts the initial d_0-dimensional node features $X \in \mathbb{R}^{n \times d_0}$ into d-dimensional candidate embeddings, e.g., $f(X; W_x) = X W_x$ or $f(X; W_x) = \text{MLP}[X; W_x]$.

Now assume that σ is a differentiable component-wise non-expansive (CONE) mapping as specified in (Gu et al., 2020). This condition stipulates that σ applies the same function individually

to each element of its input, and that this function satisfies $\|x - y\| \geq \|\sigma(x) - \sigma(y)\|$ for all $\{x, y\} \in \mathbb{R}^2$ (with some abuse of notation, we will overload the definition of σ when the meaning is clear from context). Furthermore, we assume that the weights W_p are such that $\lambda_{\text{pf}}(|P \otimes W_p|) < 1$, where $|\cdot|$ denotes the element-wise absolute value and λ_{pf} refers to the Peron-Frobenius eigenvalue.[2]

Under these conditions, it has been shown in (Gu et al., 2020) that $\lim_{k \to \infty} Y^{(k)} = Y^*$, where Y^* satisfies the fixed-point equation $Y^* = \sigma[PY^*W_p + f(X; W_x)]$. Therefore, as an IGNN forward pass we can iterate (2) K times, where K is sufficiently large, to compute node embeddings $Y^{(K)} \approx Y^*$ for use within the meta-loss from (1). For the IGNN backward pass, implicit differentiation methods (Bai et al., 2019; El Ghaoui et al., 2020), carefully tailored for handling graph-based models (Gu et al., 2020), can but used to compute gradients of Y^* with respect to W_p and W_x. Critically, this implicit technique does *not* require storing each intermediate representation $\{Y^{(k)}\}_{k=1}^K$ from the forward pass, and hence is quite memory efficient even if K is arbitrarily large. As K can be viewed as quantifying the extent of signal propagation across the graph, IGNNs can naturally capture long-range dependencies with a K-*independent* memory budget unlike more traditional techniques.

In follow-up work from (Liu et al., 2021a) a special case of (2) is considered whereby σ is an identity mapping and W_p is constrained to be a symmetric matrix W_p^s. These simplifications facilitate a more stream-lined implementation, referred to as *efficient infinite-depth graph neural networks* (EIGNN) in (Liu et al., 2021a), while maintaining accurate predictions on various node classification tasks. As we will demonstrate in the next section, the EIGNN fixed point minimizes an explicit energy function allowing this approach to be simultaneously interpreted as a special case of UGNN as well. Additionally, while EIGNN is designed to to economize IGNNs by removing the nonlinear activation, more recent work has targeted increasing IGNN expressivity (Baker et al., 2023) by relaxing the requirement that IGNN weights satisfy the Peron-Frobenius eigenvalue constraint from above in certain settings using monotone operator theory; hence the name MIGNN. And in a related vein, it has also been proposed to instantiate IGNNs (and the propagation P in particular) using a parameterized graph Laplacian operator that may be more suitable for handling potential oversmoothing effects (Fu et al., 2023).

4. A General-Purpose Unfolded GNN Framework

In this section we will first introduce a general-purpose energy function that, when reduced with simplifying assumptions, decomposes into various interpretable special cases that have been proposed in the literature, both as the basis for new UGNN models as well as motivation for canonical GNN architectures. Later, we will describe the generic proximal gradient descent iterations designed to optimize this energy. By construction, these iterations unfold in one-to-one correspondence with the UGNN model layers of each respective architecture under consideration.

4.1 Flexible Energy Function Design

Unlike IGNN models, the starting point of UGNNs is an energy function. In order to accommodate a broad variety of existing graph-regularized objectives and ultimately GNN architectures as special

---

2. The Peron-Frobenius (PF) eigenvalue is a real, non-negative eigenvalue that exists for all square, non-negative matrices and produces the largest modulus.

cases, we propose the rather general form

$$\ell_Y(Y; \mathcal{W}, f, \rho, \widetilde{B}, \phi) \triangleq \|Y - f(X; W_x)\|_{W_f}^2 + \sum_{i=1}^{m} \rho\left(\left[\widetilde{B} Y W_p Y^\top \widetilde{B}^\top\right]_{ii}\right) + \sum_{i=1}^{n} \phi(\boldsymbol{y}_i; W_\phi), \quad (3)$$

where $\mathcal{W} = \{W_x, W_f, W_p, W_\phi\}$ is a set of four weight matrices, $\rho : \mathbb{R} \to \mathbb{R}$ is a differentiable function, and $\widetilde{B} \in \mathbb{R}^{m \times n}$ is a function of A that can be viewed as a generalized incidence matrix. Furthermore, $\phi : \mathbb{R}^d \to \mathbb{R}$ is an arbitrary function (possibly non-smooth) of the node embeddings that is bounded from below, and the weighted norm $\|\cdot\|_W$ is defined such that $\|X\|_W^2 = \text{tr}[X^\top W X]$. The above loss is composed of three terms: (i) A quadratic penalty on deviations of the embeddings Y from the node-wise base model $f(X; W_x)$, (ii) a (possibly) non-convex graph-aware penalty that favors smoothness across edges (see illustrative special cases below), and (iii) an arbitrary function of each embedding that is independent of both the graph structure A and the original node features X.

As a representative special case, if $\widetilde{B} = B$, then the second term from (3) satisfies

$$\sum_{i=1}^{m} \rho\left(\left[\widetilde{B} Y W_p Y^\top \widetilde{B}^\top\right]_{ii}\right) = \sum_{\{i,j\} \in \mathcal{E}} \rho\left(\|\boldsymbol{y}_i - \boldsymbol{y}_j\|_{W_p}^2\right), \quad (4)$$

which penalizes deviations between the embeddings of two nodes sharing an edge. As has been noted in our prior work (Yang et al., 2021), if ρ is chosen to be concave and non-decreasing on $[0, \infty)$, the resulting regularization favors robustness to bad edges, and could be useful for handling heterophily graphs or adversarial attacks. Intuitively, this robustness occurs because the penalized differences between the embeddings of two nodes sharing a spurious edge no longer accumulate quadratically, a growth-rate characterized by a well-known sensitivity to outliers (West, 1984).[3]

Simplifying further, if in addition to the assumption from (4), ϕ is set to zero, ρ is a (non-robust) identity mapping, $W_f = I$, and $W_p = \lambda I$ with $\lambda > 0$, then (3) collapses to the more familiar loss

$$\ell_Y(Y; W_x, f) \triangleq \|Y - f(X; W_x)\|_{\mathcal{F}}^2 + \lambda \text{tr}\left[Y^\top L Y\right], \quad (5)$$

as originally proposed in (Zhou et al., 2004) to smooth base predictions from $f(X; W_x)$ using a quadratic, graph-aware regularization factor $\text{tr}\left[Y^\top L Y\right] = \sum_{\{i,j\} \in \mathcal{E}} \|\boldsymbol{y}_i - \boldsymbol{y}_j\|^2$. This construction has also formed the basis of a wide variety of work linking different GNN architectures (Ma et al., 2020; Pan et al., 2021; Xue et al., 2023; Zhang et al., 2020; Zhu et al., 2021); more details to follow in Section 4.2.

Similarly, if we allow for broader choices of $\widetilde{B} \neq B$ with $\widetilde{L} \triangleq \widetilde{B}^\top \widetilde{B} = \pi(A)$ for some function $\pi : \mathbb{S}^n \to \mathbb{S}^n$ over the space of n-dimensional PSD matrices $\mathbb{S}^n$, then (5) can be generalized by swapping L for this $\widetilde{L}$ as proposed in (Ioannidis et al., 2018; Fu et al., 2023). Candidates for $\widetilde{L}$ include normalized graph Laplacians (Von Luxburg, 2007) and various diffusion kernels designed according to application-specific requirements, e.g., graph signal denoising or graph rewiring (Gasteiger et al., 2019; Klicpera et al., 2019b). Moreover, if we reintroduce $W_p \neq I$, then it can be shown that the EIGNN fixed point (assuming suitable hyperparameter settings) minimizes the resulting generalized version of (5); see Appendix for specifics of the derivation as the original EIGNN algorithm (Liu et al., 2021a) is not motivated in this way.

---

3. 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. Meanwhile, concave regularization grows slowly once the embeddings are already sufficiently separated, muting the impact of outliers.

4.2 Descent Iterations That Form UGNN Layers

We now derive descent iterations for the general loss from the previous section (and later more interpretable special cases) that will be mapped to UGNN layers. For this purpose, let $U^{(k)}$ denote a single gradient descent step along (3), excluding the possibly non-differentiable ϕ-dependent term, and evaluated at some candidate point $Y^{(k)}$. We may compute such an abridged gradient step as[4]

$$U^{(k)} = Y^{(k)} - \alpha \left[\widetilde{B}^\top \Gamma^{(k)} \widetilde{B} Y^{(k)} \left(W_p + W_p^\top \right) + Y^{(k)} \left(W_f + W_f^\top \right) - f\left(X; W \right) \right], \quad (6)$$

where α is the step size and $\Gamma^{(k)}$ is a diagonal matrix with i-th diagonal element given by

$$\gamma_e^{(k)} = \left. \frac{\partial \rho\left(z \right)}{\partial z} \right|_{z = \mathrm{diag}\left[\widetilde{B} Y^{(k)} W_p \left(Y^{(k)} \right)^\top \widetilde{B}^\top \right]_e}, \quad (7)$$

and $e \in \{1, \dots, m\}$ is an index for the e-th edge in $\mathcal{E}$. We then have the following:

Lemma 1 *If ρ has Lipschitz continuous gradients, then the proximal gradient update*

$$Y^{(k+1)} = \mathrm{prox}_\phi\left(U^{(k)} \right) \triangleq \arg\min_Y \left[\frac{1}{2\alpha} \| U^{(k)} - Y \|_{\mathcal{F}}^2 + \sum_i \phi\left(\boldsymbol{y}_i; W_\phi \right) \right], \quad (8)$$

is guaranteed to satisfy $\ell_Y(Y^{(k+1)}; \mathcal{W}, f, \rho, \widetilde{B}, \phi) \leq \ell_Y(Y^{(k)}; \mathcal{W}, f, \rho, \widetilde{B}, \phi)$ for any $\alpha \in (0, 1/\mathcal{L}]$, where $\mathcal{L}$ is the Lipschitz constant for gradients of (3) w.r.t. Y, excluding the non-smooth ϕ term.

The function $\mathrm{prox}_\phi : \mathbb{R}^d \to \mathbb{R}^d$ is known as the *proximal operator*[5] associated with ϕ (separably extended across all n nodes in the graph), and the proof follows from basic properties of gradient descent (Bertsekas, 1999) and proximal methods (Combettes and Pesquet, 2011); see Appendix.

Generally speaking, the mapping $Y^{(k)} \mapsto U^{(k)}$ from (6) can be viewed as a (possibly nonlinear) graph filter, while prox_ϕ from (8) serves as an activation function applied to the embedding of each node. Collectively then, $Y^{(k+1)} = \mathrm{prox}_\phi\left(U^{(k)} \right)$ provides a flexible template for UGNN layers that naturally reduces to common GNN architectures per various design choices. And analogous to IGNN, we can execute K UGNN steps to approximate some Y^*, which could be a fixed-point, stationary point, or global optimum; more on this in Sections 4.3 and 5 below.

For example, consider the selection $\sum_i \phi\left(\boldsymbol{y}_i; W_\phi \right) = \sum_{i,j} \mathcal{I}_\infty[y_{ij} < 0]$, where y_{ij} is the (i, j)-th element of Y and $\mathcal{I}_\infty$ is an indicator function that assigns an infinite penalty to any $y_{ij} < 0$. The proximal operator then becomes $\mathrm{prox}_\phi(U) = \mathrm{ReLU}(U)$, i.e., a ReLU function that element-wise sets negative values to zero. If we add this ϕ-dependent term to (5) (as a special case of (3)), the resulting update becomes

$$Y^{(k+1)} = \mathrm{prox}_\phi\left(U^{(k)} \right) = \mathrm{ReLU}\left(Y^{(k)} - \alpha \left[\left(\lambda L + I \right) Y^{(k)} - f\left(X; W \right) \right] \right). \quad (9)$$

And when combined with observations from (Ma et al., 2020; Pan et al., 2021; Zhang et al., 2020; Zhu et al., 2021),[6] if we initialize with $Y^{(0)} = f\left(X; W \right) = XW$ and apply simple reparameterizations, the first iteration reduces to the canonical GCN layer $Y^{(1)} = \mathrm{ReLU}\left[\widetilde{D}^{-1/2} \widetilde{A} \widetilde{D}^{-1/2} XW \right]$

---

4. Note that here w.l.o.g. we have absorbed an extra $W_f + W_f^\top$ factor into $f(X; W)$.

5. If (8) happens to have multiple solutions, then the proximal operator selects one of them so as to remain a proper deterministic function of its argument, which is relevant for forming later connections with IGNN.

6. While these works do not consider the treatment of nonlinear activations, they do nicely introduce how the embedded linear gradient step $U^{(k)}$ relates to GCN filters.

from (Kipf and Welling, 2017). Subsequent iterations are no longer equivalent, although the inherent differences (e.g., the skip connections from the input layer) are useful for avoiding oversmoothing effects that can at times hamper deep GCN models (Oono and Suzuki, 2020; Li et al., 2018; Rong et al., 2020).

Additionally, the widely-used APPNP architecture (Klicpera et al., 2019a) is also a special case of (9) when the ReLU operator is removed (i.e., ϕ is zero), $\alpha = \frac{1}{1+\lambda}$, and L is changed to the symmetric normalized Laplacian (Von Luxburg, 2007). And as a final representative example, if we adopt a nonlinear choice for ρ, then L in (9) will be replaced by $L^{(k)} \triangleq B^\top \Gamma^{(k)} B$, where $\Gamma^{(k)}$ rescales graph edges. As has been discussed in (Yang et al., 2021), this instantiates a form of graph attention, whereby the attention weights $\Gamma^{(k)}$ produced by a concave, non-decreasing ρ add robustness to spurious edges. To see why this happens, note that if ρ is concave and non-decreasing, then each $\gamma_e^{(k)}$ computed via (7) will necessarily be a *non-increasing* function of z. Furthermore, taking $W_p = I$ and $\widetilde{B} = B$ for simplicity of exposition, then $z = \mathrm{diag}\left[\widetilde{B}Y^{(k)}W_p\left(Y^{(k)}\right)^\top \widetilde{B}^\top\right]_e = \|\boldsymbol{y}_i - \boldsymbol{y}_j\|_2^2$ for some $(i,j) \in \mathcal{E}$. Hence if the difference between $\boldsymbol{y}_i$ and $\boldsymbol{y}_j$ becomes *large*, as may be expected across a spurious edge, the corresponding attention weight $\gamma_e^{(k)}$ will necessarily become *small* as desired to avoid dominating the loss.

4.3 UGNN Fixed Points and Connections with IGNN

While UGNN layers are formed from the descent steps of a graph-regularized energy, for a more direct comparison with IGNN, it is illuminating to consider the situation where the update $Y^{(k+1)} = \mathrm{prox}_\phi\left(U^{(k)}\right)$ is iterated with k becoming sufficiently large. In this case, we may ideally reach a fixed point Y^* that satisfies

$$
\begin{aligned}
Y^* &= \mathrm{prox}_\phi\left(Y^* - \alpha\left[\widetilde{B}^\top \Gamma^* \widetilde{B}Y^*\left(W_p + W_p^\top\right) + Y^*\left(W_f + W_f^\top\right) - f\left(X;W\right)\right]\right) \\
&= \mathrm{prox}_\phi\left(Y^*\left[I - \alpha W_f^s - \alpha W_p^s\right] + \alpha\left[I - \widetilde{B}^\top \Gamma^* \widetilde{B}\right]Y^*W_p^s + \alpha f\left(X;W\right)\right), \quad (10)
\end{aligned}
$$

where $W_p^s \triangleq W_p + W_p^\top$ and $W_f^s \triangleq W_f + W_f^\top$ are symmetric weight matrices, and Γ^* is defined analogously to (7). It now follows that if $W_f^s = I - W_p^s, \alpha = 1$ and $\Gamma^* = I$ (i.e., ρ is an identity mapping), and we define the propagation operator as $P = I - \widetilde{B}^\top \widetilde{B} \equiv I - \widetilde{L}$, then (10) reduces to

$$
Y^* = \mathrm{prox}_\phi\left[PY^*W_p^s + f\left(X;W\right)\right]. \quad (11)
$$

This expression is exactly equivalent to the IGNN fixed point from Section 3 when we set σ to the proximal operator of ϕ and we restrict W_p to be symmetric. For direct head-to-head comparisons then, it is instructive to consider what CONE activation functions σ can actually be expressed as the proximal operator of some penalty ϕ. In this regard we have the following:

Lemma 2 *A continuous CONE function $\sigma : \mathbb{R} \to \mathbb{R}$ can be expressed as the proximal operator of some function ϕ iff σ is also non-decreasing.*

The proof follows from the analysis in (Gribonval and Nikolova, 2020). This result implies that, at least with respect to allowances for decreasing activation functions σ, IGNN is more flexible than UGNN.

However, the relative flexibility of IGNN vs UGNN fixed points becomes much more nuanced once we take into account convergence considerations. For example, the proximal gradient iterations

from Section 4.2 do *not* require that prox$_\phi$ is non-expansive or CONE to guarantee descent, although it thus far remains ambiguous how this relates to obtainable fixed points of UGNN. To this end, in Section 5 we will evaluate when such fixed points exist, what properties they have, convergence conditions for reaching them, and importantly for present purposes, how UGNN fixed points relate to their IGNN counterparts.

4.4 Potential for Broader Fixed-Point Equivalency Regimes

Before we tackle formal convergence considerations, a natural question arises regarding the extent to which IGNN and UGNN fixed-points are capable of alignment when we grant greater flexibility to the former. In particular, if for the moment we relax the restriction that IGNN iterations must strictly adhere to (2), we could potentially treat *any* UGNN fixed-point as the fixed-point of a richer class of IGNN models merely by definition. We would therefore obtain a trivial equivalency, and in this sense, IGNNs could be interpreted as uniformly more expressive than UGNNs.

The unresolved issue though, at least for the purposes of the IGNN/UGNN definitions we have adopted for this paper, is that to actually qualify as an IGNN there must exist a stable/scalable procedure for *implicitly* computing the gradients needed for back-propagation and model training. In other words, we require implicit differentiation conditioned on a given fixed point in order to achieve the signature K-independent memory budget of IGNNs as reviewed in Section 3. Without this, we are just swapping names and there is diminished reason to differentiate IGNNs and UGNNs in the first place. However, to our knowledge established implicit differentiation schemes applicable to producing GNN-like architectures equipped with graph propagation are all limited to handling fixed-points induced by (2), with different variants tied to varying choices for σ, P, W_p and f. Generalizing further is non-trivial, necessitating architecture-specific accommodations; see Section D of (Gu et al., 2020) for details of the nuance involved and (Baker et al., 2023) for follow-up effort demonstrating the difficulties of extending further. In fact, even without complex graph structure linking all training instances, achieving stable implicit differentiation is known to be challenging. Hence we adopt (2) for characterizing the design space of IGNN models, and leave broader alternatives, should they arise, to future work.

5. Convergence Comparisons

In this section we first detail situations whereby convergence to a unique fixed point, that also may at times correspond with a unique UGNN global minimum, can be established. In these situations IGNN facilitates stronger guarantees in terms of broader choices for W_p and σ. Later, we examine alternative scenarios whereby UGNN has a distinct advantage with respect to convergence guarantees to potentially local solutions. We remark that prior work examining UGNN iterations has largely been limited to certain special cases of (3), and critically, only established a non-increasing cost function (see for example (Yang et al., 2021)), *not* actual convergence to a meaningful solution set; the latter requires confirmation of additional non-trivial criteria as we will tackle herein. As such, an ancillary benefit of this section is to ground UGNN models to more comprehensive, elucidating convergence properties.

5.1 Convergence to Unique Global Solutions

To facilitate the most direct, head-to-head comparison between IGNN and UGNN, we consider a restricted form of the UGNN energy from (3). In particular, let

$$
\ell_Y(Y; \mathcal{W}, f, \phi) \triangleq \|Y - f(X; W_x)\|_{W_f}^2 + \mathrm{tr}\left[Y^\top \widetilde{L} Y W_p\right] + \sum_{i=1}^{n} \phi(\boldsymbol{y}_i), \tag{12}
$$

which is obtainable from (3) when ρ is an identity mapping and ϕ is assumed to have no trainable parameters. We also define $\Sigma \triangleq I \otimes W_f^s + \widetilde{L} \otimes W_p^s$, with minimum and maximum eigenvalues given by $\lambda_{\min}(\Sigma)$ and $\lambda_{\max}(\Sigma)$ respectively.

Theorem 3 *If* prox_ϕ *is non-expansive and* $\lambda_{min}(\Sigma) > 0$, *then (12) has a unique global minimum. Additionally, starting from any initialization* $Y^{(0)}$, *the iterations* $Y^{(k+1)} = \mathrm{prox}_\phi\left[U^{(k)}\right]$ *applied to (12) with* $\alpha \in (0, 2/\lambda_{max}(\Sigma))$ *are guaranteed to converge to this global minimum as* $k \to \infty$.

Corollary 4 *For any non-decreasing CONE function* σ *and symmetric* $W_p \equiv W_p^s$ *satisfying* $\|W_p^s\|_2 < \|P\|_2^{-1}$, *there will exist a function* ϕ *such that* $\sigma = \mathrm{prox}_\phi$ *and the fixed point of (11) is the unique global minimum of (12) when* W_f *is chosen such that* $W_f^s = I - W_p^s$ *and* $P = I - \widetilde{L}$.

Lemma 5 *For any non-expansive* σ *(not necessarily element-wise) and* W_p *(not necessarily symmetric) satisfying* $\|W_p\|_2 < \|P\|_2^{-1}$, *iterating (2) will converge to a unique fixed point.*[7]

Hence from Theorem 3 and Corollary 4, if we are willing to accept symmetric graph propagation weights W_p^s (and a non-decreasing σ), UGNN enjoys the same convergence guarantee as IGNN, but with the added benefit of an interpretable underlying energy function. In contrast, when W_p is *not* symmetric as in Lemma 5, we can only guarantee a unique IGNN fixed point, but we are no longer able to establish an association with a specific UGNN energy. In fact, it does not seem likely that any familiar/interpretable functional form can even possibly exist to underlie such a fixed point when W_p is not symmetric. This is largely because of the following simple result:

Lemma 6 *There does not exist any second-order smooth function* $h : \mathbb{R}^{n \times d} \to \mathbb{R}$ *such that* $\partial h(Y; W)/\partial Y = YW$ *for all (asymmetric) matrices* $W \in \mathbb{R}^{d \times d}$ *with* $d > 1$ *and* $n \geq 1$.

And by continuity arguments, a similar result will apply to many non-smooth functions. Consequently, with asymmetric weights there is no obvious way to associate fixed points with stationary points as we have done for UGNN. Interestingly, while a wide variety of work on unfolding models (including outside of the graph domain) has led to architectures with symmetric or even more restrictive PSD weight constraints (Amos and Kolter, 2017; Di Giovanni et al., 2023; Frecon et al., 2022; Xie et al., 2023; Yang et al., 2022), the apparent inevitability of these constraints remains underexplored.

5.2 Broader Convergence Regimes

As we move to alternative energy functions with nonlinear dependencies on the graph, e.g., ρ not equal to identity, meaningful (albeit possibly weaker) convergence properties for UGNN can still be

---

7. An analogous result has been shown in (Gu et al., 2020), but restricted to CONE mappings (not arbitrary non-expansive mappings) and with a dependency on the less familiar Peron-Frobenius eigenvalue; see Section 3.

established. In particular, we consider convergence to local minima, or more generally, stationary points of the underlying objective. However, because (3) may be non-convex and non-smooth, we must precisely define an applicable notion of stationarity.

While various possibilities exist, for present purposes we define Y^* as a stationary point of (3) if $\mathbf{0} \in \partial_{\mathcal{F}} \left[\ell_Y(Y^*; \mathcal{W}, f, \rho, \widetilde{B}, \phi) \right]$, where $\partial_{\mathcal{F}}$ denotes the Fréchet subdifferential. The latter generalizes the standard subdifferential as defined for convex functions, to non-convex cases. More formally, the Fréchet subdifferential (Li et al., 2020a) of some function $h(Y)$ is defined as the set

$$\partial_{\mathcal{F}}[h(Y)] = \left\{ S : h(Y) \geq h(Z) + \mathrm{tr}\left[S^\top (Y - Z) \right] + o\left(\|Y - Z\|_{\mathcal{F}} \right) \ \forall Y \right\}, \qquad (13)$$

which is equivalent to the gradient when h is differentiable and the regular subdifferential when h is convex. Loosely speaking, this definition specifies a set of affine functions that approximately lower-bound $h(Y)$ within a restricted region that is sufficiently close to Y as specified by the $o\left(\|Y - Z\|_{\mathcal{F}} \right)$ term; however, as we move away from Y the affine bound need not hold, which allows for a meaningful, non-empty subdifferential in broader non-convex regimes. Based on this definition, we have the following:

Theorem 7 *Assume that W_p and W_f are PSD, ϕ is continuous with a non-empty Fréchet sub-differential for all Y[8] and ρ is a concave, non-decreasing function with Lipschitz-continuous gradients. Additionally, starting from any initialization $Y^{(0)}$, let $\left\{ Y^{(k)} \right\}_{k=0}^{\infty}$ denote a sequence generated by $Y^{(k+1)} = \mathrm{prox}_\phi \left(U^{(k)} \right)$ with step size parameter $\alpha \in (0, 1/\mathcal{L}]$. Then all accumulation points of $\left\{ Y^{(k)} \right\}_{k=0}^{\infty}$ are stationary points of (3). Furthermore, $\lim_{k\to\infty} \ell_Y(Y^{(k)}; \mathcal{W}, f, \rho, \widetilde{B}, \phi) = \ell_Y(Y^*; \mathcal{W}, f, \rho, \widetilde{B}, \phi)$ for some stationary point Y^*.*

The proof is based on Zangwill's global convergence theorem (Luenberger, 1984) and additional results from (Sriperumbudur and Lanckriet, 2009); see Appendix for further details.

Corollary 8 *If in addition to the conditions from Theorem 7, ϕ is non-expansive and ρ is cho-sen such that the composite function $\rho\left[(\cdot)^2\right]$ is convex, then $\lim_{k\to\infty} \ell_Y(Y^{(k)}; \mathcal{W}, f, \rho, \widetilde{B}, \phi) = \ell_Y(Y^*; \mathcal{W}, f, \rho, \widetilde{B}, \phi)$, where Y^* is a global minimizer of (3).*

These results both apply to situations where the k-dependent UGNN graph propagation operator $P^{(k)} \triangleq I - \widetilde{B}^\top \Gamma^{(k)} \widetilde{B}$ is nonlinear by virtue of the $Y^{(k)}$-dependency of $\Gamma^{(k)}$, i.e., $P^{(k)}Y^{(k)}W_p^s \neq PY^{(k)}W_p^s$ for any fixed P. In contrast, it is unknown (and difficult to determine) if general nonlinear alternatives to PYW_p in (2) will converge.

5.3 Recap of Relative IGNN and UGNN Flexibility

In terms of model expressiveness, the advantage of the IGNN framework is two-fold: (i) it allows for asymmetric weights W_p while still providing a strong convergence guarantee, and (ii) it allows for decreasing activation functions. However, the latter is likely much less relevant in practice, as most deep models adopt some form of non-decreasing activation anyway, e.g., ReLU, etc.

In contrast, UGNN models are more flexible than IGNN in their accommodation of: (i) nonlinear graph propagation through the graph attention mechanism that results from a nonlinear ρ as mentioned

---

8. For minor technical reasons, we also assume ϕ is such that $\lim_{\|Y\|\to\infty} \ell_Y(Y; \mathcal{W}, f, \rho, \widetilde{B}, \phi) = \infty$.

in Section 4.2, and (ii) expansive proximal operators. While the latter may seem like a mere technicality, expansive proximal operators actually undergird a variety of popular sparsity shrinkage penalties, which have been proposed for integration with GNN models (Zheng et al., 2021). For example, the ℓ_0 norm and related non-convex regularization factors (Chen et al., 2017b; Fan and Li, 2001) are expansive and can be useful for favoring parsimonious node embeddings. And overall, with only mild assumptions, UGNN at least guarantees cost function descent across a broad class of models per Lemma 1, with even convergence to stationary points possible in relevant situations from Theorem 7 that IGNN cannot emulate.

6. How Limiting Are Symmetric (Layer-Tied) Propagation Weights?

Previously we observed that the primary advantage IGNN has over UGNN, at least in terms of model expressiveness, is that IGNN places no restriction that the propagation weight matrix W_p need be symmetric, although both models ultimately involve an architecture with layer-tied weights unlike typical message-passing models such as GCN. To better understand the implications of this distinction, we will now explore the expressiveness of GNN models with symmetric, layer-tied weights. We remark that resolution of these issues is of growing relevance, as evidenced by a number of recent papers.

For example, (Di Giovanni et al., 2023) explores what can be categorized as UGNN models and the emergence of symmetric weights; however, there is no direct analysis of the inevitability of symmetric weights, the expressiveness relative to GNNs with asymmetric weights, nor consideration of the limitations of layer-tied weights. Additionally, relative to our general UGNN framework emanating from (3), their approach excludes ρ and non-linear activations. Critically, without the latter, our Theorem 10 presented below is not possible. In a related vein, (Luo et al., 2022) addresses the expressiveness of weight-tied GNN layers relative to GCN models, although these results are a special case of ours, and symmetric weights are not considered.[9] More broadly, symmetric weights naturally emerge when deriving unfolding architectures analogous to common deep feed-forward models such as MLPs, ResNets, or Transformers (Amos and Kolter, 2017; Frecon et al., 2022; Xie et al., 2023; Yang et al., 2022). However, even these works do not quantify the expressiveness relative to unconstrained weights as we investigate here.

6.1 Fixed-Point Equivalency with Symmetric and Asymmetric Weights

In this section we examine situations whereby UGNN models are able to reproduce, up to some inconsequential transform, the fixed points of an IGNN, even when the latter includes an asymmetric propagation weight matrix W_p. However, because it is challenging to analyze general situations with arbitrary activation functions, we first consider the case where σ is an identity mapping. We then have the following:

Theorem 9 *For any $W_p \in \mathbb{R}^{d \times d}$, $W_x \in \mathbb{R}^{d_0 \times d}$, $X \in \mathbb{R}^{n \times d_0}$, and P that admit a unique IGNN fixed point $Y^* = PY^*W_p + f(X; W_x)$, there exists a $Y' \in \mathbb{R}^{n \times d'}$, $\tilde{W}_x \in \mathbb{R}^{d \times d'}$, right-invertible*

---

9. We remark that while (Di Giovanni et al., 2023) and (Luo et al., 2022) represent valuable, complementary contributions that have been independently introduced by others, our work was both posted to arxiv and submitted to JMLR well in advance, and hence should not be viewed as derivative of them. [Note: This footnote is to provide context specifically for reviewing purposes and can be removed later.]

transform $T \in \mathbb{R}^{d \times d'}$, and symmetric $W_p^s \in \mathbb{R}^{d' \times d'}$, such that

$$Y'T = PY'TW_p^s + f(X; W_x)\tilde{W}_x, \quad with \ \|Y' - Y^*\|_{\mathcal{F}} < \epsilon, \ \forall \epsilon > 0. \tag{14}$$

This result implies that a UGNN model with $\phi = 0$ can produce node-wise embeddings $Y = Y'T$ capable of matching any IGNN fixed-point with arbitrary precision up to transforms T and $\tilde{W}_x$. And given that T can be absorbed into the meta-loss output layer g from (1) (which is often a linear layer anyway), and $\tilde{W}_x$ can be absorbed into f, the distinction introduced by these transforms is negligible.

Proceeding further, if we allow for nonlinear activation functions σ, we may then consider a more general, parameterized family of penalty functions $\phi(\boldsymbol{y}; W_\phi)$ such that the resulting proximal operator increases our chances of finding a UGNN fixed point that aligns with IGNN. However, some care must be taken to control UGNN model capacity to reduce the possibility of trivial, degenerate alignments. To this end, we consider proximal operators in the set

$$\mathcal{S}_\sigma = \{\text{prox}_\phi : \boldsymbol{y} \mapsto G\sigma(C\boldsymbol{y}) | \text{with } \{G, C\} \text{ chosen such that } \text{prox}_\phi \text{ is proximal operator.}\}, \tag{15}$$

where the matrices G and C can be aggregated into W_ϕ. It is then possible to derive sufficient conditions under which UGNN has optimal solutions equivalent to IGNN fixed points (it remains an open question if a necessary condition can be similarly derived). See Appendix for details.

6.2 UGNN Capacity to Match Canonical GCN Architectures

The previous section considered the alignment of fixed points, which are obtainable after executing potentially infinite graph propagation steps, from models with and without symmetric propagation weights. Somewhat differently, this section investigates analogous issues in the context of the possible embeddings obtainable after k steps of graph propagation. Specifically, we evaluate the expressiveness of a canonical GCN model with arbitrary, layer-independent weights versus a UGNN model structured so as to effectively maintain an equivalent capacity (up to a right-invertible linear transformation as discussed above).

Theorem 10 *Let $Y_{\text{GCN}}^{(k+1)} = \sigma\left(PY_{\text{GCN}}^{(k)}W_p^{(k)} + \beta Y_{\text{GCN}}^{(k)}\right)$ denote the k-th layer of a GCN, where $W_p^{(k)}$ is a layer-dependent weight matrix, $\sigma = \text{prox}_\phi$ for some function ϕ, and $\beta \in \{0, 1\}$ determines whether or not a residual connection is included. Then with $\alpha = 1$, ρ an identity mapping, and $f(X; W_x)$ set to zero, there will always exist a right-invertible T, initialization $Y^{(0)}$, and symmetric weights W_p^s and W_f^s such that the k-th iteration step computed via (8) satisfies $Y^{(k+1)}T = Y_{\text{GCN}}^{(k+1)}$.*

Given the close association between GCNs and certain UGNN models, Theorem 10 suggests that, at least in principle, symmetric layer-tied weights may not be prohibitively restrictive in the finite layer regime. And as can be observed from the proof, this equivalency result is possible using a matching parameter budget instantiated through an expanded hidden dimension but constrained via block-sparse weight matrices W_p^s and W_f^s. Of course in practice when we extrapolate outside of regimes where the results of this section strictly hold, it remains to be seen if symmetric weight constraints have a significant adverse impact. We explore such considerations next.

7. Empirical Accuracy Comparisons

In this section we evaluate the predictions of various comparable IGNN and UGNN models across experimental settings designed to complement our theoretical findings and showcase potential strengths and weaknesses that may arise in practice. To this end, we first conduct a series of quasi-synthetic tests designed to explore the interplay between the type of graph propagation weights (symmetric as with UGNN vs asymmetric as with IGNN), the graph propagation path length (finite as with UGNN vs approximately infinite as with IGNN), and model expressiveness. We then turn to real-world benchmarks to examine relative capabilities handling long-range dependencies, graph heterophily, and adversarial attacks, all of which represent domains within which either IGNNs, UGNNs, or both have been motivated in prior work. Under these different application scenarios, we also evaluate against popular methods specifically designed for the particular task at hand. As for IGNN/UGNN model architectures, we assume the following:

- **IGNN baselines**: We use (2) as the grounding for IGNN baselines with the following notable specializations. First, when σ is an identity, we refer to the resulting model as EIGNN following (Liu et al., 2021a), and when σ is a ReLU activation with relaxed weight constraints on W_p from (Baker et al., 2023), the model is denoted MIGNN (because of its origins from monotone operator theory). We also remark that beyond these candidates, there is as of yet no more expressive IGNN parameterization that we are aware of that comes equipped with an implementation suitable for stable training and implicit differentiation across large graphs.

- **UGNN baselines**: For UGNN we limit our exploration to various special cases of the general framework introduced in Section 4 and instantiated via the composite update described by (6), (7), and (8). To recap the role of these steps, when conditioned on fixed graph attention (6) computes an affine filter that includes a graph propagation operator (as with IGNNs), (7) serves as a form of nonlinear graph attention operator (unique to currently-available UGNNs), and (8) produces the nonlinear activation function for each layer (analogous to σ for IGNNs). While UGNNs also accommodate expansive proximal operators beyond σ, we do not consider this flexibility herein.

Further specification of dataset-specific architectural choices and hyperparameters will be provided in the forthcoming subsections, while full experimental details are deferred to the Appendix. All models and testing were implemented using the Deep Graph Library (DGL) (Wang et al., 2019). We also note that portions of these experiments have appeared previously in our conference paper (Yang et al., 2021), which presents a useful UGNN-related modeling framework and supporting DGL-based code that we further augmented for our purposes herein.

7.1 Exploring the Practical Impact of Symmetric Weights

From our analysis in Section 6 it is reasonable to speculate that UGNN symmetric weights are not a significant limitation relative to IGNN when it comes to model expressiveness, potentially even in regimes that deviate from the stated technical conditions, or when UGNN has limited propagation steps as may occur in practical implementations. To explore this hypothesis empirically, we design the following quasi-synthetic experiment: First, we train separate IGNN and UGNN models on the ogbn-arxiv dataset (Hu et al., 2020), where the architectures are equivalent with the exception of W_p and the number of propagation steps (see Appendix for all network and training details). Additionally, because UGNN requires symmetric propagation weights, a matching parameter count can be achieved by simply expanding the UGNN hidden dimension. Once trained, we then generate

predictions from each model and treat them as new, ground-truth labels. We next train three additional models separately on both sets of synthetic labels:

1. An IGNN with architecture equivalent to the original used for generating labels,

2. An analogous UGNN, but with limited propagation steps consistent with practical use cases,

3. A hybrid model that can equivalently be viewed as either an IGNN with finite propagation steps, or a UGNN where the restriction on symmetric weights has been relaxed.

For all three models above the number of parameters is a small fraction of the number of labels to mitigate overfitting issues.

Gen \ Rec	IGNN	UGNN	Hybrid
IGNN	95.7 ± 0.9	89.8 ± 1.9	91.6 ± 1.2
UGNN	91.0 ± 1.5	94.5 ± 0.8	92.3 ± 0.7

Table 1: Accuracy recovering quasi-synthetic labels.

Prediction accuracy results are presented in Table 1, where the rows indicate the label generating models, and the columns denote the recovery models. We first consider the leftmost two columns labeled IGNN and UGNN. As would be expected, the highest recovery accuracy (95.7 and 94.5) occurs when the generating and recovery models are matched such that perfect recovery is theoretically possible by design. But even in the mismatched cases (91.0 and 89.8) the performance is still quite strong, suggesting that a UGNN with symmetric weights and limited propagation steps can reasonably approximate an IGNN, and vice versa.

To further probe the source of the modest gap that nonetheless does exist, we next examine the rightmost hybrid column of Table 1. Here we observe that the hybrid model performs somewhat worse recovering IGNN data (91.6 vs 95.7), indicating that truncated propagation steps, potentially more so than symmetric weights, reduces performance when the generating model involves true long-range propagation as with IGNN. Not surprisingly, we also notice that when trained on UGNN data, the hybrid model outperforms IGNN (92.3 vs 91.0) since here there is no benefit to modeling long-range propagation beyond the limit of the truncated UGNN propagation steps.

In general though, the fact that the performance of all models varies within a range of a few percentage points, even under these idealized conditions where the data-generating model has maximal advantage, suggests a significant overlap of model expressiveness as might be expected per the analysis in Section 6. Hence UGNN models with symmetric weights and finite propagation steps are likely adequate in many practical scenarios.

7.2 Results Using Long-Dependency Benchmarks

We now turn to experiments involving existing graph benchmarks purported to involve long-range dependencies, a scenario that has been previously used to motivate both UGNN and IGNN models. The modeling goal here is to capture these long-range effects by stably introducing a sufficient number of propagation layers without the risk of oversmoothing, which might otherwise degrade performance. We begin with the synthetic Chains node classification dataset (Gu et al., 2020; Liu

et al., 2021a) explicitly tailored to evaluate GNN models w.r.t. long-range dependencies. Results of IGNN and UGNN models, as well as baselines SGC (Chen et al., 2020), GCN (Kipf and Welling, 2017), GAT (Velickovic et al., 2018), and SSE (Dai et al., 2018) are shown in Figure 2. Baseline results are from (Gu et al., 2020) without error bars; however, in our own testing we observe the results to be quite stable across trials such that error bars are not necessary for conveying the core message. Note that SSE, like IGNN and to some extent UGNNs, is explicitly designed for capturing long-range dependencies. While the Chains dataset has been shown in the past to differentiate IGNN and SSE from traditional GNN architectures (Gu et al., 2020), we observe here that UGNN is equally capable of matching performance, and hence a more challenging benchmark is needed.
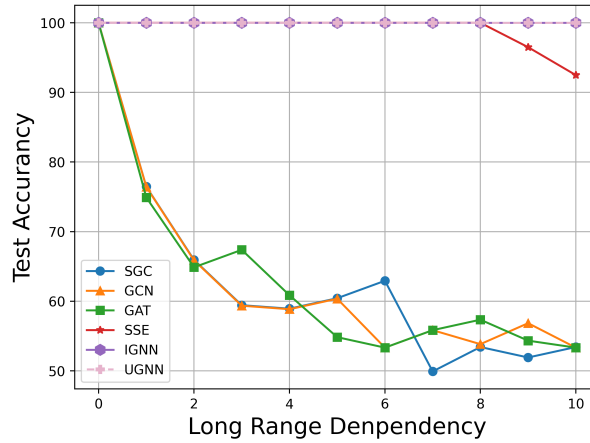

Figure 2: Accuracy comparisons on long-range Chains dataset.

For this purpose we next turn to the Amazon Co-Purchase dataset, a node classification benchmark often used for evaluating long-range dependencies, in part because of sparse labels relative to graph size (Baker et al., 2023; Dai et al., 2018; Gu et al., 2020).[10] We adopt the test setup from (Dai et al., 2018; Gu et al., 2020; Baker et al., 2023), and compare performance using different label ratios. We report results from five baselines, namely SGC, GCN, SSE, IGNN, and MIGNN, where the latter is a more expressive version of IGNN. Regardless, UGNN is still able to outperform all of them by a non-trivial margin as shown in Figures 3 and 4. This likely indicates that while some long-range capability is needed to outperform the canonical GNNs, the unbounded extent of IGNNs provides diminishing marginal returns beyond the UGNN, which has greater flexibility in other relevant aspects. We also remark that, to the degree that MIGNN is more expressive than prior

---

10. An alternative is the Long-Range Graph Benchmark (Dwivedi et al., 2023); unfortunately though, this benchmark is exclusively composed of tiny graphs for solving graph-level prediction problems whereby scalable UGNN and IGNN node-level embeddings are less relevant. Yet another option is PPI (Hamilton et al., 2017a); however, multiple existing methods can already achieve nearly 100% accuracy (Gu et al., 2020; Liu et al., 2021a), so it is less impactful for differentiation.

IGNNs, this does not translate into improved classification performance, possibly because the core architecture is still the same as discussed in Section 3.
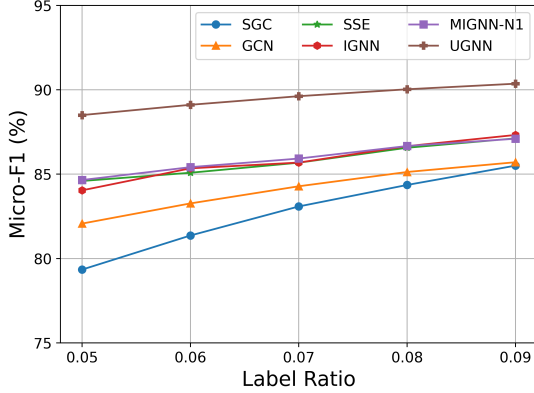

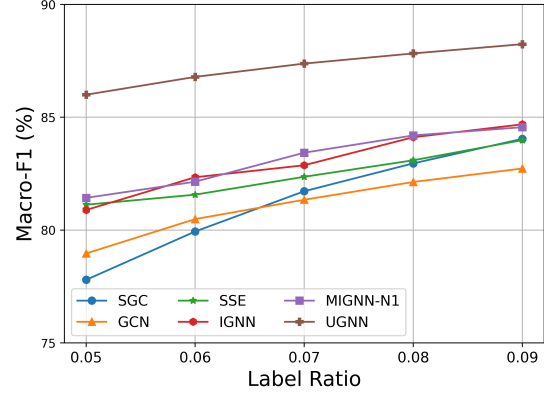Figure 3: Micro-F1 on Amazon Co-Purchase.   Figure 4: Macro-F1 on Amazon Co-Purchase.

7.3 Results on Heterophily Graphs

A *homophily* graph occurs when nearby nodes share similar labels and/or features, and message-passing GNNs are generally well-equipped to handle them. In contrast, *heterophily* graphs exhibit an opposing phenomena, namely, nodes sharing an edge have a reduced likelihood of having the same label. These complementary notions are often quantified via the homophily ratio

$$\mathcal{H} \triangleq \frac{|\{(u,v) \in \mathcal{E} \ : \ \boldsymbol{t}_u = \boldsymbol{t}_v\}|}{|\mathcal{E}|} \tag{16}$$

from (Zhu et al., 2020b), where $\boldsymbol{t}_u$ and $\boldsymbol{t}_v$ are the target labels of nodes u and v respectively. $\mathcal{H}$ quantifies the tendency of nodes to be connected with other nodes from the same class. Graphs with $\mathcal{H} \approx 1$ exhibit strong homophily, while conversely, those with $\mathcal{H} \approx 0$ show strong heterophily, indicating that many edges are connecting nodes with different labels.

With this in mind, it has been suggested that GNNs with more sensitivity to long-range dependencies might be well-suited for handling heterophily graphs (Liu et al., 2021a; Pei et al., 2019), since nodes sharing the same label may now be concentrated many hops away, i.e., a form of long-range association. But we conjecture that the utility of such long-range GNNs will depend on attributes of the heterophilic graph not directly measured by the homophily ratio. More concretely, the implicit assumption is that if we increase the receptive field of a given node, the percentage of nodes with the same label will be higher, improving model performance.

7.3.1 A CLOSER LOOK AT HETEROPHILY AND LONG-RANGE DEPENDENCIES

To kick the tires on the above assumption, we first compute an additional metric across a variety of heterophily graphs. We name this metric the *average same-label distance* (ASD), with formal definition given by

$$\text{ASD} \triangleq \frac{1}{c} \mathbb{E}_{u \in \mathcal{V}} \mathbb{E}_{v \in \mathcal{L}_u} \text{dist}(u,v), \tag{17}$$

where $\mathcal{L}_u$ denotes the set of nodes with the same label as node u, c is the number of classes as introduced previously, and $\text{dist}(u,v)$ counts the shortest path length between nodes u and v. To

reduce computational complexity, we compute the expectation in (17) by randomly sampling 1000 nodes as u and then averaging. The resulting log-scale ASD values are presented in Figure 5 across four classic heterophily datasets examined in (Pei et al., 2019), Actor, Cornell, Texas, and Wisconsin, and the more recent heterophily benchmarks Roman, Amazon, Minesweeper, Tolokers, and Questions from (Platonov et al., 2023).

Interestingly, there is a clear demarcation between the new heterophily datasets and the classical ones: The new datasets unanimously possess a larger $\log(\text{ASD})$ (all $\log(\text{ASD}) > 1$) while the classical ones are smaller (all $\log(\text{ASD}) < 1$). We will now empirically explore the impact of this distinction on performance. To streamline this treatment, we reference the datasets from (Platonov et al., 2023) with $\log(\text{ASD}) > 1$ as ASD_{sup}, while the datasets from (Pei et al., 2019) with $\log(\text{ASD}) < 1$ are referenced as ASD_{inf}.

7.3.2 HOW DIFFERENT MANIFESTATIONS OF HETEROPHILY IMPACT PERFORMANCE

Intuition suggests that incorporating long-range dependencies will be more effective on the ASD_{sup} datasets, since a larger effective receptive field, as occurs with greater numbers of propagation steps over the graph, can potentially lead to aggregation over more nodes with the same label. In contrast, for ASD_{inf} heterophily graphs, same-label nodes may actually be rela-

Figure 5: Average same-label distance (ASD) on heterophily benchmarks.

tively close together, but mixed in with appreciable concentrations of mismatched-labeled nodes. Under these conditions long-range propagation may be less effective and instead, treating edges between mismatched nodes as outliers a preferable alternative. For UGNN models, such treatment can be operationalized by choosing ρ as a concave non-decreasing function as espoused in Section 4.1. And as alluded to in Section 4.2, this leads to an emergent form of graph attention that is capable of selectively pruning bad edges.

To test this hypothesis, we first train both IGNN and UGNN models on the four ASD_{inf} datasets from Section 7.3.1, using the data splits, processed node features, and labels provided by (Pei et al., 2019). Notably, we include UGNN models both *with* and *without* a concave ρ function (by "without" here we just mean that ρ is an inconsequential identity mapping). Node classification accuracy results are presented in Table 2, where for context, we also include results from the additional GNN architectures GCN (Kipf and Welling, 2017), GAT (Velickovic et al., 2018), GraphSAGE (Hamilton et al., 2017b), SOTA heterophily methods GEOM-GCN (Pei et al., 2019) and H$_2$GCN (Zhu et al., 2020b), as well as a baseline MLP as reported in (Zhu et al., 2020b). Two things stand out regarding these results. First, the IGNN, EIGNN, and UGNN (without ρ) models, are comparable or worse than the MLP, subject to some dataset-to-dataset variability, indicating that merely introducing long-range
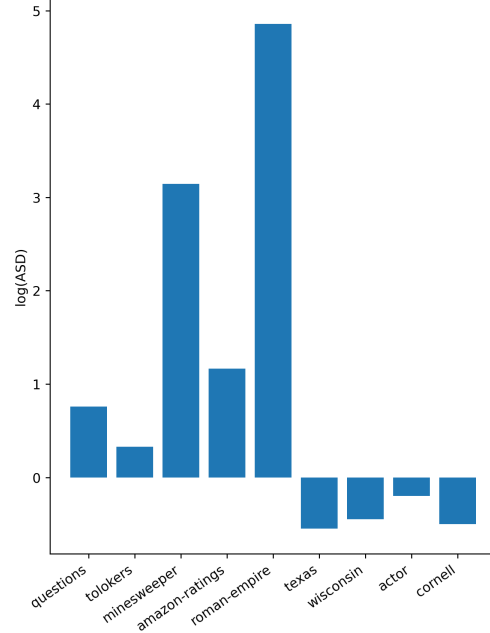
18

DATASET	TEXAS	WISCONSIN	ACTOR	CORNELL	AVG.
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	51.64
GAT	58.38±4.45	55.29±8.71	26.28±1.73	58.92±3.32	49.72
GRAPHSAGE	82.43±6.14	81.18±5.56	34.23±0.99	75.95±5.01	68.45
GEOM-GCN	67.57	64.12	31.63	60.81	56.03
H$_2$GCN	**84.86±6.77**	**86.67±4.69**	**35.86±1.03**	**82.16±4.80**	**72.39**
MLP	81.89±4.78	85.29±3.61	35.76±0.98	81.08±6.37	71.01
IGNN	83.1±5.6	79.2±6.0	35.5±2.9	76.0±6.2	68.45
EIGNN	80.54±5.67	83.72±3.46	34.74±1.19	81.08±5.70	70.02
UGNN(W/O ρ)	81.62±5.51	82.75±7.83	37.10±1.07	83.51±7.30	71.25
UGNN(W/ ρ)	**84.59±3.83**	**86.67±4.19**	**37.43±1.50**	**86.76±5.05**	**73.86**

Table 2: Node classification accuracy on $\mathrm{ASD}_{\mathrm{inf}}$ heterophily graphs. Boldface indicates the best performing model within each block. When equipped with a concave ρ, a simple/generic UGNN model outperforms IGNN models and is competitive with strong baselines explicitly designed for heterophily graphs.

dependencies alone may not necessarily offer a clear-cut advantage over ignoring the graph structure altogether. In contrast, UGNN (with ρ) is uniformly better than the other IGNN/UGNN models, as well as the MLP. This observation is consistent with the notion that heterophily of the $\mathrm{ASD}_{\mathrm{inf}}$ variety can be mitigated using robust/concave penalization that mutes the impact of edges connecting nodes with mismatched labels. This strategy also allows this UGNN model to even outperform strong GNN architectures explicitly designed for heterophily graphs, e.g., H$_2$GCN.

To provide complementary evidence for our hypothesis, we next train IGNN and UGNN models on the five $\mathrm{ASD}_{\mathrm{sup}}$ benchmarks. With these datasets heterophily manifests more as a longer average distance between nodes sharing a label, rather than spurious edges connected to any given node that can be adequately treated as outliers. As such, we might expect that the effectiveness of ρ will be marginal relative to long-range graph propagations that are tantamount to a large receptive field. Results are shown in Table 3, along with a set of recent heterophily GNN models tested in (Platonov et al., 2023) for reference. These models include H$_2$GCN (Zhu et al., 2020c), GPR-GCN (Chien et al., 2020), FSGNN (Maurya et al., 2022), FAGNCN (Bo et al., 2021), GBK-GNN (Du et al., 2022) and JacobiConv (Wang and Zhang, 2022). We also include an MLP as well as GCN and GAT baselines for reference.[11]

From these results, we observe that all UGNN/IGNN-based models significantly outperform the MLP, suggesting that same labels spread across a wider receptive field can in fact be exploited within the $\mathrm{ASD}_{\mathrm{sup}}$ datasets. However, the UGNN with concave ρ no longer offers an advantage, in

---

11. For GCN and GAT we ran the experiments ourselves using the standard implementations from DGL. While (Platonov et al., 2023) also includes results associated with a number of popular baseline GNN architectures (e.g., GCN, GAT, GraphSAGE, GT, and variants thereof etc.), upon inspection of the associated public codebases, we find that the actual implementations involve additional tweaks/enhancements which deviate from the original models. These and other types of modifications may be effective on specific tasks/datasets, and are interesting to consider; however, they can be applied across a wide variety of GNN architectures and are well outside of our analysis-driven priorities here.

that IGNN and UGNN (regardless of ρ) perform similarly. Additionally, both IGNN and UGNN are competitive with the best performing heterophily-motivated GNNs, e.g., FSGNN and GBK-GNN.

DATASET	ROMAN	AMAZON	MINESWEEPER	TOLOKERS	QUESTIONS	AVG.
GCN	53.31 ± 1.55	49.86 ± 0.21	77.49 ± 0.60	76.53 ± 0.11	76.61 ± 0.41	66.76
GAT	71.31 ± 0.55	50.52 ± 0.52	88.86 ± 0.51	75.72 ± 6.19	74.34 ± 0.38	72.15
H$_2$GCN	60.11 ± 0.52	36.47 ± 0.23	89.71 ± 0.31	73.35 ± 1.01	63.59 ± 1.46	64.65
GPR-GNN	64.85 ± 0.27	44.88 ± 0.34	86.24 ± 0.61	72.94 ± 0.97	55.48 ± 0.91	64.88
FSGNN	$\mathbf{79.92 \pm 0.56}$	$\mathbf{52.74 \pm 0.83}$	90.08 ± 0.70	$\mathbf{82.76 \pm 0.61}$	$\mathbf{78.86 \pm 0.92}$	**76.87**
FAGCN	65.22 ± 0.56	44.12 ± 0.30	88.17 ± 0.73	77.75 ± 1.05	77.24 ± 1.26	70.50
GBK-GNN	74.57 ± 0.47	45.98 ± 0.71	$\mathbf{90.85 \pm 0.58}$	81.01 ± 0.67	74.47 ± 0.86	73.38
JACOBICONV	71.14 ± 0.42	43.55 ± 0.48	89.66 ± 0.40	68.66 ± 0.65	73.88 ± 1.16	69.38
MLP	65.88 ± 0.38	45.90 ± 0.52	50.89 ± 1.39	72.95 ± 1.06	70.34 ± 0.76	61.19
IGNN	$\mathbf{84.04 \pm 0.85}$	51.79 ± 0.46	90.73 ± 1.45	82.06 ± 0.72	75.44 ± 0.45	76.81
EIGNN	75.72 ± 0.29	45.27 ± 0.78	88.17 ± 0.66	79.90 ± 0.76	69.13 ± 0.65	71.64
UGNN(w/o ρ)	82.75 ± 1.71	50.99 ± 0.37	$\mathbf{91.41 \pm 0.20}$	$\mathbf{85.62 \pm 0.17}$	$\mathbf{75.83 \pm 0.40}$	**77.32**
UGNN(w/ ρ)	83.54 ± 0.47	$\mathbf{53.38 \pm 0.35}$	87.58 ± 0.08	84.83 ± 0.25	75.21 ± 0.21	76.91

Table 3: Node classification accuracy on $\mathrm{ASD_{sup}}$ heterophily graphs. Boldface indicates the best performing model within each block.

7.4 Adversarial Attack Results

As demonstrated in the previous section, when heterophily implicitly manifests as graphs with spurious edges, UGNN models with a concave ρ provide a unique advantage over IGNN models. To further explore this advantage, we turn to an application scenario whereby spurious edges are explicit, namely, certain forms of adversarial attacks. To this end, we compare the models using graph data corrupted via the the Mettack algorithm (Zügner and 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 and Zitnik, 2020), setting the edge perturbation rate to 20%, adopting the "Meta-Self" training strategy, and a GCN as the surrogate model.

MODEL	ATK-CORA	ATK-CITESEER	AVG.
SURROGATE (GCN)	57.38 ± 1.42	60.42 ± 1.48	58.90
GNNGUARD	$\mathbf{70.46 \pm 1.03}$	65.20 ± 1.84	**67.83**
GNN-JACCARD	64.51 ± 1.35	63.38 ± 1.31	63.95
GNN-SVD	66.45 ± 0.76	$\mathbf{65.34 \pm 1.00}$	65.90
EIGNN	65.16 ± 0.78	68.36 ± 1.01	66.76
IGNN	61.47 ± 2.30	64.70 ± 1.26	63.09
UGNN(w/o ρ)	62.89 ± 1.59	63.83 ± 1.95	63.36
UGNN(w/ ρ)	$\mathbf{70.23 \pm 1.09}$	$\mathbf{70.63 \pm 0.93}$	**70.43**

Table 4: Node classification accuracy under Adversarial attacks. Boldface indicates the best performing model within each block.

As baselines for reference, we use three strong defense models, namely GNNGuard (Zhang and Zitnik, 2020), GNN-Jaccard (Wu et al., 2019) and GNN-SVD (Entezari et al., 2020). Results on Cora and Citeseer data are reported in Table 4. As in Table 2, we again observe that UGNN with ρ outperforms the IGNN models as well as UGNN without ρ. And somewhat surprisingly, UGNN with ρ also performs comparably or better than strong existing models that were meticulously designed to defend against adversarial attacks. These result further demonstrate the UGNN advantage in terms of robustness relative to presently-available IGNN models.

8. Scalability Considerations

In this section we provide complexity analysis and supporting system efficiency empirical comparisons for full-graph training with UGNN and IGNN models. We then conclude with a discussion of sampling-based and lazy training methods that can be applied to further speed up UGNN models.

8.1 Complexity Analysis

Time Complexity: The time complexity of UGNN is $O(mdK + Nnd^2)$, where m is the number of edges, n is the number of nodes, K is the number of propagation steps, N is the total number of MLP layers and d is the largest hidden size. As a useful reference point, the time complexity of a standard K-layer GCN or GraphSAGE model is $O(mdK + Knd^2)$. Moreover, if the UGNN MLP layers are all after propagation (i.e., no parameters before propagation), the time complexity can be reduced to $O(Nnd^2)$ by precomputing the propagation (i.e., the same as an MLP).

For IGNN (as well as MIGNN), the time complexity depends on the number of iterations needed to drive (2) to an approximate fixed point, which relates to certain properties of the weight matrix W_p. However, the official IGNN implementation[12] executes propagation steps until either acceptable error rate or an upper threshold is reached. If the number of iteration steps used to approximate the fixed point is K, then the IGNN time complexity is $O(mdK + Nnd^2)$, which is essentially the same as UGNN (and other message passing-based GNNs), although generally the value of K will be much higher. That being said, the computational complexity of the IGNN backward pass alone is constant with respect to K, although this does not change the overall complexity order.

For EIGNN, based on its linearity, the overall/composite graph propagation matrix can actually be pre-computed in $O(n^3)$ time. In this way, it does not need to be recomputed during training, which reduces the training time complexity by a factor of $O(Nnd^2)$. However, for large-sized graphs often encountered in practice, an $O(n^3)$ computation can be a prohibitive overhead, which would also need to be recomputed any time a propagation-related hyper-parameter is modified.

Memory Complexity: Absent sampling-based training methods to be discussed further in Section 8.3 below, the memory complexity of UGNN is $O(md(K + N))$; a comparable K-layer GCN or GraphSAGE model would require $O(mdK)$. Meanwhile the IGNN memory complexity is only $O(md)$, which is constant with respect to K (a key advantage).

8.2 Empirical Comparisons

In terms of time and memory consumption in practice, we compare UGNN and IGNN on the Amazon Co-Purchase benchmark, which has been advocated in (Gu et al., 2020) as a suitable data source for testing IGNN. Results are shown in Figures 6 and 7 based on executing 100 steps of training and

---

12. https://github.com/SwiftieH/IGNN

evaluation on a single Tesla T4 GPU. Clearly, IGNN maintains a huge advantage in terms of memory consumption, while UGNN has a faster runtime provided the number of propagation steps is not too large. Of course these results are based on full-graph training, so it is worth considering additional scalability measures as described next.
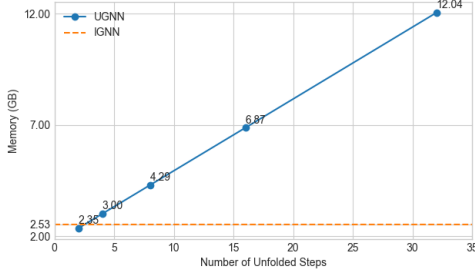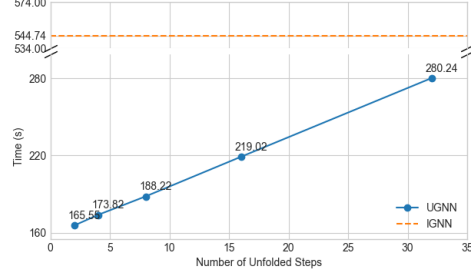

Figure 6: Memory cost

Figure 7: Running time

8.3 Directions for Additional Scalability

We are presently unaware of existing work to further scale IGNN models to handle huge real-world graphs, for example, using sampling-based methods (Chen et al., 2018; Chiang et al., 2019; Hamilton et al., 2017a; Ying et al., 2018; Zou et al., 2019) or historical embedding caches (Chen et al., 2017a; Fey et al., 2021) as have been applied to typical message-passing GNNs. While in principle such methods may be applicable, the stability of implicit differentiation would need to be maintained, and present IGNN work thus far has largely focused on full graph training. In contrast, there exist two recent orthogonal directions explicitly designed for scaling UGNNs.

First, lazy training methods, combined with sampling and a historical embedding cache, have been developed such that UGNN complexity is largely decoupled from model depth and far more efficient than full-graph training (Xue et al., 2023). As a second approach, UGNN models have been combined with offline sampling (Zeng et al., 2021; Gasteiger et al., 2022) such that training can proceed on the very largest publicly-available benchmarks (Jiang et al., 2023). This includes the 377GB dataset MAG240M from (Hu et al., 2021) and the 1.15TB dataset IGB-full from (Khatua et al., 2023), where UGNN models akin to (9), but integrated with offline sampling, achieve SOTA performance. As opposed to full-graph training, the memory requirement for this approach reduces to $O(n_s d(K + N))$, where n_s is the size of the offline sampled subgraphs. A comparable GCN or GraphSAGE model would utilize $O(n_s dK)$.

9. Conclusions

This work has closely examined the relationship between IGNNs and UGNNs, shedding light on their similarities and differences. In this regard, representative contributions and take-home messages are as follows:

- Existing IGNN models broadly coalesce around the propagation update from (2), with notable differences tied to varying choices for σ, P, W_p, and f. As for UGNN, we have introduced a flexible foundation via the general energy from (3) and the attendant descent steps given by (6), (7), and (8). Collectively, these subsume a wide variety of existing UGNN models, while also motivating previously-unused variants with convergence guarantees and broader coverage of IGNN capabilities.

- IGNNs should in principle have an advantage when long-range propagation and full-graph training are dominant factors leading to high accuracy. However, the practical significance of such an advantage is presently limited for two reasons: (i) Many node-classification benchmarks in common use today do not actually require long-range propagation, and relatively shallow GNNs already perform well; (ii) Even in instances where long-range propagation may be helpful (e.g., certain heterophily graphs, the Amazon Co-Purchase benchmark, etc.), the dataset scales are such that UGNNs can easily be applied as well while maintaining greater flexibility for matching or exceeding the accuracy. Of course these observations are subject to change as new application scenarios emerge, and IGNNs remain an elegant paradigm to consider for future use cases.

- We have demonstrated that UGNN symmetric graph propagation weights are essentially unavoidable, while IGNNs have no such restriction. And yet our complementary analytical and empirical results suggest that these symmetric UGNN weights are not a significant hindrance relative to IGNNs, at least in terms of model expressiveness and subsequent node classification accuracy. In contrast, UGNN nonlinear graph propagation operators, as instantiated via a concave ρ and the attention weights from (7), can be advantageous relative to existing IGNNs when unreliable edges are present, e.g., as evidenced by the comparisons in Tables 2 and 4 involving certain types of heterophily graphs and adversarial attacks respectively.

- Regarding heterophily in particular, a by-product of our analysis is the differentiation of two heterophily regimes, characterized by distinct distributions of same-label nodes as quantified by the ASD metric in Figure 5. Within the $\mathrm{ASD_{sup}}$ regime, both UGNN and IGNN models excel by exploiting long-range dependencies; however, in the $\mathrm{ASD_{inf}}$ regime only UGNNs with a concave ρ are competitive.

- IGNN has a major advantage in memory costs for full-graph training, since backpropagation does not require storing intermediate node representations across all propagation steps. However, UGNNs have an advantage in terms of computational complexity if the number of propagation steps can be truncated without compromising accuracy. Furthermore, UGNN memory and computation savings are possible with sampling-based methods or lazy training.

- UGNN model energy functions in particular provide a natural entry point for inducing versatile GNN architectures that yield predictable results. Natural candidates for extending beyond (3) include the handling of heterogeneous graphs using an energy whose minimization mimics knowledge graph embedding computations (Ahn et al., 2022), or weaving negative samples within an energy function targeting link prediction tasks (Wang et al., 2024). Future work could also consider exploiting UGNN models for the sake of explainability, e.g., disentangling the importance of input features versus network effects in arriving at node-level predictions.

Overall, both models benefit from the inductive biases of their respective design criteria, which frequently overlap but retain important areas of distinction that we have elucidated herein.

References

Hongjoon Ahn, Yongyi Yang, Quan Gan, Taesup Moon, and David Wipf. Descent steps of a relation-aware energy produce heterogeneous graph neural networks. *Advances in Neural Information Processing Systems*, 35, 2022.

Brandon Amos and J Zico Kolter. Optnet: Differentiable optimization as a layer in neural networks. In *International Conference on Machine Learning*, pages 136–145. PMLR, 2017.

Shaojie Bai, J Zico Kolter, and Vladlen Koltun. Deep equilibrium models. *arXiv preprint arXiv:1909.01377*, 2019.

Justin Baker, Qingsong Wang, Cory D. Hauck, and Bao Wang. Implicit graph neural networks: A monotone operator viewpoint. In *International Conference on Machine Learning*, 2023.

Dimitri Bertsekas. *Nonlinear Programming*. Athena Scientific, 2nd edition, 1999.

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

Jianfei Chen, Jun Zhu, and Le Song. Stochastic training of graph convolutional networks with variance reduction. *arXiv preprint arXiv:1710.10568*, 2017a.

Jie Chen, Tengfei Ma, and Cao Xiao. FastGCN: fast learning with graph convolutional networks via importance sampling. *arXiv preprint arXiv:1801.10247*, 2018.

Ming Chen, Zhewei Wei, Zengfeng Huang, Bolin Ding, and Yaliang Li. Simple and deep graph convolutional networks. In *Proceedings of the 37th International Conference on Machine Learning, ICML*, volume 119, pages 1725–1735, 2020.

Siheng Chen and Yonina C Eldar. Graph signal denoising via unrolling networks. In *ICASSP 2021-2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 5290–5294, 2021.

Yichen Chen, Dongdong Ge, Mengdi Wang, Zizhuo Wang, Yinyu Ye, and Hao Yin. Strong NP-hardness for sparse optimization with concave penalty functions. In *International Confernece on Machine Learning*, 2017b.

Wei-Lin Chiang, Xuanqing Liu, Si Si, Yang Li, Samy Bengio, and Cho-Jui Hsieh. 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*, pages 257–266, 2019.

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

Patrick L Combettes and Jean-Christophe Pesquet. Proximal splitting methods in signal processing. In *Fixed-point algorithms for inverse problems in science and engineering*, pages 185–212. Springer, 2011.

Hanjun Dai, Zornitsa Kozareva, Bo Dai, Alex Smola, and Le Song. Learning steady-states of iterative algorithms over graphs. In *International Conference on Machine Learning*, pages 1106–1114, 2018.

Francesco Di Giovanni, James Rowbottom, Benjamin Paul Chamberlain, Thomas Markovich, and Michael M Bronstein. Understanding convolution on graphs via energies. *Transactions on Machine Learning Research*, 2023.

Lun Du, Xiaozhou Shi, Qiang Fu, Xiaojun Ma, Hengyu Liu, Shi Han, and Dongmei Zhang. Gbk-gnn: Gated bi-kernel graph neural networks for modeling both homophily and heterophily. In *Proceedings of the ACM Web Conference 2022*, pages 1550–1558, 2022.

Vijay Prakash Dwivedi, Chaitanya K Joshi, Anh Tuan Luu, Thomas Laurent, Yoshua Bengio, and Xavier Bresson. Benchmarking graph neural networks. *Journal of Machine Learning Research*, 24(43):1–48, 2023.

Laurent El Ghaoui, Fangda Gu, Bertrand Travacca, Armin Askari, and Alicia Tsai. Implicit deep learning. *arXiv preprint arXiv:1908.06315*, 2020.

Negin Entezari, Saba A Al-Sayouri, Amirali Darvishzadeh, and Evangelos E Papalexakis. 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*, pages 169–177, 2020.

Jianqing Fan and Runze Li. Variable selection via nonconcave penalized likelihood and its oracle properties. *JASTA*, 96(456):1348–1360, 2001.

Matthias Fey, Jan E Lenssen, Frank Weichert, and Jure Leskovec. GNNAutoScale: Scalable and expressive graph neural networks via historical embeddings. In *International Conference on Machine Learning*, pages 3294–3304. PMLR, 2021.

Ben Finkelshtein, Xingyue Huang, Michael Bronstein, and Ismail Ilkan Ceylan. Cooperative graph neural networks. *International Conference on Machine Learning*, 2024.

Jordan Frecon, Gilles Gasso, Massimiliano Pontil, and Saverio Salzo. Bregman neural networks. In *International Conference on Machine Learning*, pages 6779–6792. PMLR, 2022.

Guoji Fu, Mohammed Haroon Dupty, Yanfei Dong, and Lee Wee Sun. Implicit graph neural diffusion based on constrained dirichlet energy minimization. *arXiv preprint arXiv:2308.03306*, 2023.

Claudio Gallicchio and Alessio Micheli. Fast and deep graph neural networks. In *Proceedings of the AAAI Conference on Artificial Intelligence*, 2020.

Johannes Gasteiger, Stefan Weißenberger, and Stephan Günnemann. Diffusion improves graph learning. *Advances in neural information processing systems*, 32, 2019.

Johannes Gasteiger, Chendi Qian, and Stephan Günnemann. Influence-based mini-batching for graph neural networks. In *Learning on Graphs Conference*, pages 9–1. PMLR, 2022.

Zhengyang Geng, Meng-Hao Guo, Hongxu Chen, Xia Li, Ke Wei, and Zhouchen Lin. Is attention better than matrix decomposition? In *9th International Conference on Learning Representations, ICLR 2021, Virtual Event, Austria, May 3-7, 2021*. OpenReview.net, 2021.

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

Rémi Gribonval and Mila Nikolova. A characterization of proximity operators. *Journal of Mathematical Imaging and Vision*, 62(6):773–789, 2020.

Fangda Gu, Heng Chang, Wenwu Zhu, Somayeh Sojoudi, and Laurent El Ghaoui. Implicit graph neural networks. In *Advances in Neural Information Processing Systems*, 2020.

Will Hamilton, Zhitao Ying, and Jure Leskovec. Inductive representation learning on large graphs. *Advances in Neural Information Processing Systems*, 30, 2017a.

William L Hamilton, Rex Ying, and Jure Leskovec. Inductive representation learning on large graphs. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, pages 1025–1035, 2017b.

Hao He, Bo Xin, Satoshi Ikehata, and David Wipf. From bayesian sparsity to gated recurrent nets. In *Advances in Neural Information Processing Systems*, 2017.

John Hershey, Jonathan Le Roux, and Felix Weninger. Deep unfolding: Model-based inspiration of novel deep architectures. *arXiv preprint arXiv:1409.2574*, 2014.

Weihua Hu, Matthias Fey, Marinka Zitnik, Yuxiao Dong, Hongyu Ren, Bowen Liu, Michele Catasta, and Jure Leskovec. Open graph benchmark: Datasets for machine learning on graphs. *Advances in neural information processing systems*, 33:22118–22133, 2020.

Weihua Hu, Matthias Fey, Hongyu Ren, Maho Nakata, Yuxiao Dong, and Jure Leskovec. OGB-LSC: A large-scale challenge for machine learning on graphs. *arXiv:2103.09430*, 2021.

Vassilis N Ioannidis, Meng Ma, Athanasios N Nikolakopoulos, Georgios B Giannakis, and Daniel Romero. Kernel-based inference of functions on graphs. In D. Comminiello and J. Principe, editors, *Adaptive Learning Methods for Nonlinear System Modeling*. Elsevier, 2018.

Haitian Jiang, Renjie Liu, Xiao Yan, Zhenkun Cai, Minjie Wang, and David Wipf. MuseGNN: Interpretable and convergent graph neural network layers at scale. *arXiv preprint arXiv:2310.12457*, 2023.

Kedar Karhadkar, Pradeep Kr Banerjee, and Guido Montúfar. FoSR: First-order spectral rewiring for addressing oversquashing in gnns. *International Conference on Learning Representations*, 2023.

Steven M. Kearnes, Kevin McCloskey, Marc Berndl, Vijay S. Pande, and Patrick Riley. Molecular graph convolutions: moving beyond fingerprints. *J. Comput. Aided Mol. Des.*, 30(8):595–608, 2016.

Arpandeep Khatua, Vikram Sharma Mailthody, Bhagyashree Taleka, Tengfei Ma, Xiang Song, and Wen-mei Hwu. Igb: Addressing the gaps in labeling, features, heterogeneity, and size of public graph datasets for deep learning research. *arXiv preprint arXiv:2302.13522*, 2023.

D. Kinderlehrer and G. Stampacchia. *An Introduction to Variational Inequalities and Their Applications*. Classics in Applied Mathematics. Society for Industrial and Applied Mathematics (SIAM, 3600 Market Street, Floor 6, Philadelphia, PA 19104), 1980. ISBN 9780898719451.

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

Johannes Klicpera, Aleksandar Bojchevski, and Stephan Günnemann. Predict then propagate: Graph neural networks meet personalized pagerank. In *International Conference on Learning Representations*, 2019a.

Johannes Klicpera, Stefan Weißenberger, and Stephan Günnemann. Diffusion improves graph learning. In *Advances in Neural Information Processing Systems*, 2019b.

Jiajin Li, Anthony Man-Cho So, and Wing-Kin Ma. Understanding notions of stationarity in nonsmooth optimization: A guided tour of various constructions of subdifferential for nonsmooth functions. *IEEE Signal Processing Magazine*, 37(5):18–31, 2020a.

Qimai Li, Zhichao Han, and Xiao-Ming Wu. Deeper insights into graph convolutional networks for semi-supervised learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, 2018.

Yaxin Li, Wei Jin, Han Xu, and Jiliang Tang. Deeprobust: A pytorch library for adversarial attacks and defenses. *arXiv preprint arXiv:2005.06149*, 2020b.

Juncheng Liu, Kenji Kawaguchi, Bryan Hooi, Yiwei Wang, and Xiaokui Xiao. Eignn: Efficient infinite-depth graph neural networks. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, 2021a.

Xiaorui Liu, Wei Jin, Yao Ma, Yaxin Li, Hua Liu, Yiqi Wang, Ming Yan, and Jiliang Tang. Elastic graph neural networks. In *International Conference on Machine Learning*, 2021b.

D.G. Luenberger. *Linear and Nonlinear Programming*. Addison–Wesley, Reading, Massachusetts, second edition, 1984.

Yi Luo, Guiduo Duan, Guangchun Luo, and Aiguo Chen. Unifying label-inputted graph neural networks with deep equilibrium models. *arXiv preprint arXiv:2211.10629v1*, 2022.

Yao Ma, Xiaorui Liu, Tong Zhao, Yozen Liu, Jiliang Tang, and Neil Shah. A unified view on graph neural networks as graph signal denoising. *arXiv preprint arXiv:2010.01777*, 2020.

Sunil Kumar Maurya, Xin Liu, and Tsuyoshi Murata. Simplifying approach to node classification in graph neural networks. *Journal of Computational Science*, 62:101695, 2022.

Yurii Nesterov. *Introductory lectures on convex optimization: A basic course*, volume 87. Springer Science & Business Media, 2003.

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

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

Hongbin Pei, Bingzhe Wei, Kevin Chen-Chuan Chang, Yu Lei, and Bo Yang. Geom-gcn: Geometric graph convolutional networks. In *International Conference on Learning Representations*, 2019.

Oleg Platonov, Denis Kuznedelev, Michael Diskin, Artem Babenko, and Liudmila Prokhorenkova. A critical look at the evaluation of gnns under heterophily: Are we really making progress? In *International Conference on Learning Representations*, 2023.

Yu Rong, Wenbing Huang, Tingyang Xu, and Junzhou Huang. Dropedge: Towards deep graph convolutional networks on node classification. In *8th International Conference on Learning Representations, ICLR*, 2020.

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

Bharath Sriperumbudur and Gert Lanckriet. On the convergence of the concave-convex procedure. In *Advances in Neural Information Processing Systems*, 2009.

Jan Tönshoff, Martin Ritzert, Hinrikus Wolf, and Martin Grohe. Walking out of the Weisfeiler Leman hierarchy: Graph learning beyond message passing. *Transactions on Machine Learning Research*, 2023.

Jake Topping, Francesco Di Giovanni, Benjamin Paul Chamberlain, Xiaowen Dong, and Michael Bronstein. Understanding over-squashing and bottlenecks on graphs via curvature. *International Conference on Learning Representations*, 2022.

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

Petar Veličković, Rex Ying, Matilde Padovano, Raia Hadsell, and Charles Blundell. Neural execution of graph algorithms. In *International Conference on Learning Representations*, 2020.

Ulrike Von Luxburg. A tutorial on spectral clustering. *Statistics and computing*, 17(4):395–416, 2007.

Minjie Wang, Da Zheng, Zihao Ye, Quan Gan, Mufei Li, Xiang Song, Jinjing Zhou, Chao Ma, Lingfan Yu, Yu Gai, Tianjun Xiao, Tong He, George Karypis, Jinyang Li, and Zheng Zhang. Deep graph library: A graph-centric, highly-performant package for graph neural networks. *arXiv preprint arXiv:1909.01315*, 2019.

Xiyuan Wang and Muhan Zhang. How powerful are spectral graph neural networks. In *International Conference on Machine Learning*, pages 23341–23362. PMLR, 2022.

Yuxin Wang, Xiannian Hu, Quan Gan, Xuanjing Huang, Xipeng Qiu, and David Wipf. Efficient link prediction via gnn layers induced by negative sampling. *IEEE Transactions on Knowledge and Data Engineering*, 2024.

Zhangyang Wang, Qing Ling, and Thomas Huang. Learning deep ℓ_0 encoders. In *AAAI Conference on Artificial Intelligence*, 2016.

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

Huijun Wu, Chen Wang, Yuriy Tyshetskiy, Andrew Docherty, Kai Lu, and Liming Zhu. Adversarial examples for graph data: Deep insights into attack and defense. In *Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence, IJCAI*, pages 4816–4823, 2019.

Qitian Wu, Wentao Zhao, Chenxiao Yang, Hengrui Zhang, Fan Nie, Haitian Jiang, Yatao Bian, and Junchi Yan. SGFormer: Simplifying and empowering transformers for large-graph representations. *Advances in Neural Information Processing Systems*, 37, 2023.

Zonghan Wu, Shirui Pan, Fengwen Chen, Guodong Long, Chengqi Zhang, and S Yu Philip. A comprehensive survey on graph neural networks. *IEEE transactions on neural networks and learning systems*, 32(1):4–24, 2020.

Xingyu Xie, Qiuhao Wang, Zenan Ling, Xia Li, Guangcan Liu, and Zhouchen Lin. Optimization induced equilibrium networks: An explicit optimization perspective for understanding equilibrium models. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 45(3):3604–3616, 2023.

Rui Xue, Haoyu Han, MohamadAli Torkamani, Jian Pei, and Xiaorui Liu. LazyGNN: Large-scale graph neural networks via lazy propagation. In *International Conference on Machine Learning*, 2023.

Yongyi Yang, Tang Liu, Yangkun Wang, Jinjing Zhou, Quan Gan, Zhewei Wei, Zheng Zhang, Zengfeng Huang, and David Wipf. Graph neural networks inspired by classical iterative algorithms. In *International Conference on Machine Learning*, 2021.

Yongyi Yang, , Zengfeng Huang, and David Wipf. Transformers from an optimization perspective. In *Advances in Neural Information Processing Systems*, volume 35, 2022.

Rex Ying, Ruining He, Kaifeng Chen, Pong Eksombatchai, William L Hamilton, and Jure Leskovec. Graph convolutional neural networks for web-scale recommender systems. In *Proceedings of the 24th ACM SIGKDD international conference on knowledge discovery & data mining*, pages 974–983, 2018.

Hanqing Zeng, Muhan Zhang, Yinglong Xia, Ajitesh Srivastava, Andrey Malevich, Rajgopal Kannan, Viktor Prasanna, Long Jin, and Ren Chen. Decoupling the depth and scope of graph neural networks. *Advances in Neural Information Processing Systems*, 34, 2021.

Hongwei Zhang, Tijin Yan, Zenjun Xie, Yuanqing Xia, and Yuan Zhang. Revisiting graph convolutional network on semi-supervised node classification from an optimization perspective. *CoRR*, abs/2009.11469, 2020.

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

Xuebin Zheng, Bingxin Zhou, Junbin Gao, Yu Guang Wang, Pietro Lio, Ming Li, and Guido Montúfar. How framelets enhance graph neural networks. In *International Conference on Machine Learning*, 2021.

Dengyong Zhou, Olivier Bousquet, Thomas Navin Lal, Jason Weston, and Bernhard Schölkopf. Learning with local and global consistency. *Advances in Neural Information Processing Systems*, 2004.

Jie Zhou, Ganqu Cui, Zhengyan Zhang, Cheng Yang, Zhiyuan Liu, Lifeng Wang, Changcheng Li, and Maosong Sun. Graph neural networks: A review of methods and applications. *arXiv preprint arXiv:1812.08434*, 2018.

Jiong Zhu, Ryan A Rossi, Anup Rao, Tung Mai, Nedim Lipka, Nesreen K Ahmed, and Danai Koutra. Graph neural networks with heterophily. *arXiv preprint arXiv:2009.13566*, 2020a.

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

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

Meiqi Zhu, Xiao Wang, Chuan Shi, Houye Ji, and Peng Cui. Interpreting and unifying graph neural networks with an optimization framework. *arXiv preprint arXiv:2101.11859*, 2021.

Difan Zou, Ziniu Hu, Yewen Wang, Song Jiang, Yizhou Sun, and Quanquan Gu. Layer-dependent importance sampling for training deep and large graph convolutional networks. *Advances in Neural Information Processing Systems*, 32, 2019.

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

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

Appendix A. Datasets, Experiment Details, and Model Specifications

In this section we introduce additional experiment details as well as give a comprehensive introduction of the UGNN architectures used in the paper.

A.1 Datasets

Practical Impact of Symmetric Weights We choose the popular ogbn-arxiv dataset (Hu et al., 2020) for the experiments in Section 7.1. Original training splits for generating subsequent synthetic labels follow the standard OGB protocol.

Long-Range Dependency/Sparse Label Tests In Section 7.2, we adopt the Amazon Co-Purchase dataset and Chains dataset, both of which have previously been used in (Gu et al., 2020) for evaluating performance involving long-range dependencies. We use the Amazon Co-Purchase dataset provided by the IGNN repo (Gu et al., 2020), including the data-processing and evaluation code, in order to obtain a fair comparison. As for splitting, 10% of nodes are selected as the test set. Because there is no dev set, we directly report the test result of the last epoch. We also vary the fraction of training nodes from 5% to 9%. Additionally, because there are no node features, we learn a 128-dim feature vector for each node. All of these settings from above follow from (Gu et al., 2020). The Chains dataset is a synthetic dataset introduced by (Gu et al., 2020). We use exactly the same setting as in (Gu et al., 2020).

Heterophily Experiments In Section 7.3, we use several heterophily datasets, including Texas, Wisconsin, Actor and Cornell datasets which are introduced by (Pei et al., 2019), and Roman-empire, Amazon-ratings, Minesweeper, Tolokers and Questions which are introduced by (Platonov et al., 2023). We used the data split, processed node features, and labels provided by (Pei et al., 2019) and (Platonov et al., 2023) respectively.

Adversarial Attack Experiments As mentioned in Section 7.4, we tested on Cora and Citerseer using Mettack. We use the DeepRobust library (Li et al., 2020b) and apply the exact same non-targeted attack setting as in (Zhang and Zitnik, 2020). For all the baseline results in Table 4, we run the implementation in the DeepRobust library or the GNNGuard official code. Note the GCN-Jaccard results differ slightly from those reported in (Zhang and Zitnik, 2020), likely because of updates in the DeepRobust library and the fact that (Zhang and Zitnik, 2020) only report results from a single trial (as opposed to averaged results across multiple trails as we report).

Summary Statistics Table 5 summarizes the attributes of each dataset listed above.

A.2 Experiment Details

In all experiments of IGNN and EIGNN, we optionally set W_p^s to be trainable parameters[13] or fix it to I. For IGNN, following the original paper, we optionally set $f(X) = \mathrm{MLP}(PX)$ or $\mathrm{MLP}(X)$, while for EIGNN and UGNN we only set $f(X) = \mathrm{MLP}(X)$.

In label recovering task in Section 7.1 and the evaluation of time and memory in Section 8, for the purpose of comparing finite vs infinite number of propagations and symmetric vs asymmetric weights, we ignore irrelevant options such as different choice of $f(X)$, hidden size, and the attention mechanism in UGNN. We always use $f(X) = XW_x$ and $g\left[\boldsymbol{y}_i^*(\mathcal{W}); \theta\right] = W_g \boldsymbol{y}_i^*(\mathcal{W})$ where $W_g \in \mathbb{R}^{c \times d}$ is a learned matrix that maps the propagated node features to the output space. We set the

---

13. For EIGNN, following the exact formulation in (Liu et al., 2021a), we actually use $W_p^s = \mu W$, where W is trainable and is forced to be symmetric, and μ is a hyperparamter.

Table 5: Dataset statistics. The *FEATURES* column describes the dimensionality of node features. The *Amazon Co-P* row denotes the Amazon Co-Purchase dataset. Note that the Amazon Co-Purchase dataset has no node features.

DATASET	NODES	EDGES	FEATURES	CLASSES
AMAZON CO-P	334,863	2,186,607	-	58
CORA	2,708	5,429	1,433	7
CITESEER	3,327	4,732	3,703	6
OGBN-ARXIV	169,343	1,166,243	128	40
TEXAS	183	309	1,703	5
WISCONSIN	251	499	1,703	5
ACTOR	7,600	33,544	931	5
CORNELL	183	295	1,703	5
ROMAN-EMPIRE	22,662	32,927	300	18
AMAZON-RATINGS	32,927	93,050	300	5
MINESWEEPER	1,000	39,402	7	2
TOLOKERS	11,758	519,000	10	2
QUESTIONS	48,921	153,540	301	2

hidden size to 32 in asymmetric case and 34 in symmetric case to ensure nearly the same number of parameters (we deliberately set the hidden size to this small to ensure the models do not overfit). For generating models, we first train them using the original labels of the dataset by 500 steps. For UGNN, we fix the number of propagation steps is set to 2, and adopt (11) of UGNN.

When projecting W_p (or $\tilde{W}$ in UGNN) to the space that admits unique fixed point, in IGNN's original paper it uses $\|\cdot\|_{\inf}$ (Gu et al., 2020), which would break the symmetry, based on our result in Section 5.1, in our implementation when the model weight is symmetric we project $\|\cdot\|_2$ instead of $\|\cdot\|_{\inf}$. In EIGNN, we use rescaling as in (Liu et al., 2021a) instead of projection.

In the direct comparison between IGNN and UGNN in Section 7.1 and the time $\&$ memory comparison in Section 8, we always use normalized adjacency $P = \hat{A}$ for propagation matrix. However, in other experiments with UGNN, we allow extra flexibility of the propagation matrix.

Finally, for the results in Sections 7.1, 7.3 and 7.4, we run each experiment for 10 trials to obtain the standard deviation.

A.3 Basic UGNN Architecture

The UGNN architecture is composed of the input module $f(X; W)$, followed by the unfolded propagation layers defined by (8), concluding with $g(\boldsymbol{y}; \theta)$.

We allow an optional attention layer in UGNN by specifying the ρ function in (3). See Section A.5 below and (Yang et al., 2021) for details. The aggregate design of UGNN is depicted in in Figure 8. For simplicity, we generally adopt a single attention layer sandwiched between equal numbers of propagation layers; however, for heterophily datasets we apply an extra attention layer before

propagation. Note that the attention only involves reweighting the edge weights of the graph (i.e., it does not alter the node embeddings at each layer).
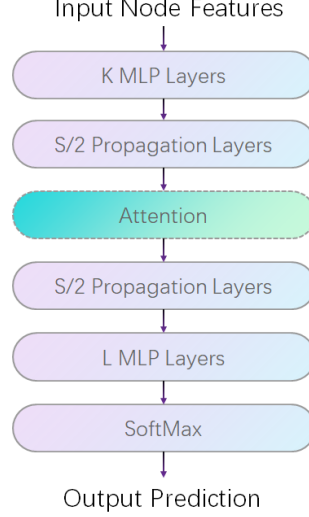

Figure 8: Model Architecture. S is the total number of propagation steps. K and L are number of MLP layers before and after the propagation respectively. While not a requirement, in all of our experiments, either K or L is set to zero, meaning that the MLP exists on only one side of the propagation layers.

A.4 UGNN Variations

Pre-conditioning and Reparameterization   If we unfold the iteration step of (5), define the reparameterized embeddings $Z = \tilde{D}^{1/2}Y$ and left multiply the update rule by $\tilde{D}^{1/2}$, we have

$$
\begin{aligned}
Z^{(k+1)} &= (1-\alpha)Z^{(k)} + \alpha\lambda\tilde{D}^{-1/2}AY^{(k)} + \alpha\tilde{D}^{-1/2}f(X;W) \\
&= (1-\alpha)Z^{(k)} + \alpha\lambda\tilde{D}^{-1/2}A\tilde{D}^{-1/2}Z^{(k)} + \alpha\tilde{D}^{-1}Z^{(0)}.
\end{aligned}
\tag{18}
$$

From here, if we choose $\alpha = \lambda = 1$, for $Z^{(1)}$ we have that

$$
\begin{aligned}
Z^{(1)} &= \left(\tilde{D}^{-1/2}A\tilde{D}^{-1/2} + \tilde{D}^{-1}\right)Z^{(0)} \\
&= \tilde{D}^{-1/2}\tilde{A}\tilde{D}^{-1/2}Z^{(0)},
\end{aligned}
\tag{19}
$$

which gives the exact single-layer GCN formulation in Z-space with $Z^{(0)} = f(X;W)$.

Normalized Laplacian Unfolding   From another perspective, if we replace L in (5) with a normalized graph Laplacian, and then take gradients steps as before, there is no need to do preconditioning and reparameterizing. For example, unfolding (5) with L changed to the symmetrically-normalized version $\tilde{L} = I - \tilde{D}^{-1/2}\tilde{A}\tilde{D}^{-1/2}$, we get

$$
Y^{(k+1)} = (1-\alpha-\alpha\lambda)Y^{(k)} + \alpha\lambda\tilde{D}^{-1/2}\tilde{A}\tilde{D}^{-1/2}Y^{(k)} + \alpha Y^{(0)},
\tag{20}
$$

where we set $\tilde{D} = I + D$. This formula is essentially the same as (18). The main difference is that there is no $\tilde{D}^{-1}$ in front of X, which indicates an emphasis on the initial features. We found this

version to be helpful on ogbn-arxiv and Amazon Co-Purchase data. Note however that all of our theoretical support from the main paper applies equally well to this normalized version, just with a redefinition of the gradient steps to include the normalized Laplacian.

A.5 Specific Attention Formula

We adapt the TWIRLS model from (Yang et al., 2021) as the basis for UGNNs with an optional attention layer that is determined by the ρ function in (3). While the attention mechanism can in principle adopt any concave, non-decreasing function ρ, in this work we restrict ρ to a single functional form that is sufficiently flexible to effectively accommodate all experimental scenarios. Specifically, we adopt

$$\rho(z^2) = \begin{cases} \bar{\tau}^{p-2} z^2 & \text{if } z < \bar{\tau} \\ \frac{2}{p}\bar{T}^p - \rho_0 & \text{if } z > \bar{T} \\ \frac{2}{p} z^p - \rho_0 & \text{otherwise,} \end{cases} \tag{21}$$

where p, $\bar{T}$, and $\bar{\tau}$ are non-negative hyperparameters and $\rho_0 = \frac{2-p}{p}\bar{\tau}^p$ is a constant that ensures ρ is continuous. Additionally, the gradient of ρ produces the attention score function (akin to γ in the main paper) given by

$$s(z^2) \triangleq \frac{\partial p(z^2)}{\partial z^2} = \begin{cases} \bar{\tau}^{p-2} & \text{if } z < \bar{\tau} \\ 0 & \text{if } z > \bar{T} \\ z^{p-2} & \text{otherwise.} \end{cases} \tag{22}$$

And for convenience and visualization, we also adopt the reparameterizations $\tau = \bar{\tau}^{\frac{1}{2-p}}$ and $T = \bar{T}^{\frac{1}{2-p}}$, and plot $\rho(z^2)$ and $s(z^2)$ in Figure 9 using $p = 0.1$, $\tau = 0.2$, $T = 2$.
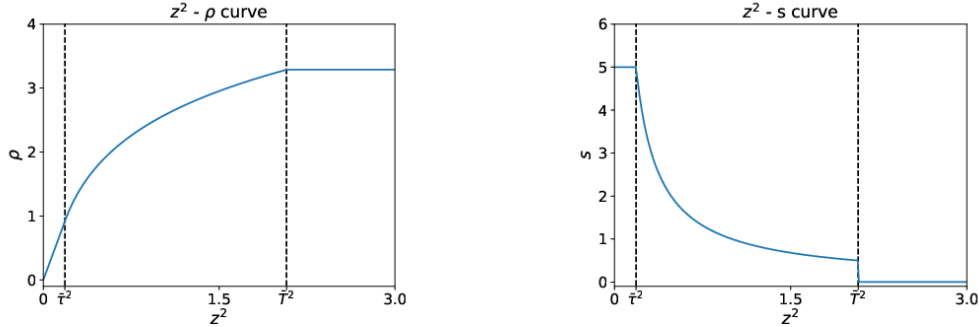

Figure 9: A visualization of attention functions.

Overall, this flexible choice has a natural interpretation in terms of its differing behavior between the intervals $[0, \bar{\tau}]$, $(\bar{\tau}, \bar{T})$, and $(\bar{T}, \infty)$. For example, in the $[0, \bar{\tau}]$ interval a quadratic penalty is applied, which leads to constant attention independent of z. This is exactly like the simple case where there is no attention. In contrast, within the $(\bar{T}, \infty)$ interval ρ is constant and the corresponding attention weight is set to zero (truncation), which is tantamount to edge removal. And finally, the middle interval provides a natural transition between these two extremes, with ρ becoming increasingly flat with larger z values.

34

Additionally, many familiar special cases emerge for certain parameter selections. For example $T = \infty$ corresponds with no explicit truncation, while $p = 2$ can instantiate no attention. And $T = \tau$ means we simply truncate those edges with large distance, effectively keeping the remaining edge attention weights at 1 (note that there is normalization during propagation, so setting edges to a constant is equivalent to setting them to 1).

Appendix B. EIGNN as a Special Case of UGNN

As claimed in the main paper, the EIGNN model can be viewed as a special case of UGNN; we provide the details here. We begin with the general energy from (3), and then choose $\phi(\boldsymbol{y}; W_\phi) = 0$, $\rho(z^2) = z^2$, and adopt the weight parameterization $W_P^s = s^2 F^\top F$, where F is a trainable parameter matrix and s is a rescaling factor given by

$$s = \left(\|F^\top F\|_+ \epsilon_F \right)^{-\frac{1}{2}}. \tag{23}$$

This produces the simplified loss

$$\ell_Y(Y; W_x, f) = \|Y - f(X; W_x)\|_{\mathcal{F}}^2 + \mu \mathrm{tr}\left[F^\top Y^\top LYF \right], \tag{24}$$

where $P = I - L$ is the normalized adjacency matrix. A gradient step on (24) initialized at $Y^{(k)}$ can then be computed as

$$Y^{(k+1)} = \gamma s^2 F^\top FY^{(k)} W_p^s + f(X, W_x), \tag{25}$$

which matches the original EIGNN updates from (Liu et al., 2021a).

Appendix C. Notation Used in Proofs

In following sections, we give proofs of all propositions in the main paper. We first introduce some notations used in proofs here.

For simplicity, in following sections we denote $\ell_Y(Y; \mathcal{W}, f, \rho, \widetilde{B}, \phi)$ in (3) by $\ell_Y(Y)$, denote $\phi(\boldsymbol{y}_i; W_\phi)$ by $\phi(\boldsymbol{y}_i)$ and $\sum_i \phi(\boldsymbol{y}_i)$ by $\phi(Y)$, and denote $f(X; W)$ by $f(X)$. And the smooth part of ℓ_Y is denoted by $\ell_Y^s(Y) = \ell_Y(Y) - \phi(Y)$. And we possibly ignore the parameter of the function (e.g. ℓ_Y).

We denote the vectorization of a matrix M by $\mathrm{vec}(M)$ and the Kronecker product of two matrices M_1 and M_2 denoted by $M_1 \otimes M_2$.

For Fréchet subdifferential respect to Y, we denote it by ∂_Y, ignoring the subscript $_{\mathcal{F}}$ used in main paper.

Appendix D. Proofs Related to UGNN Convergence

In this section, we investigate different conditions to guarantee convergence (to stationary point) of UGNN under different assumptions of ℓ_Y.

To start with, we first prove that as long as proximal gradient descent has a fixed point, the fixed point is a stationary point.

Lemma 11 *For any Z such that*

$$Z \in \arg \min_Y \frac{1}{2\alpha} \|Y - (Z - \alpha \nabla \ell_Y^s(Z))\|_\mathcal{F}^2 + \phi(Y), \tag{26}$$

it must hold that $0 \in \partial_Z \ell_Y(Z)$.

Proof We will use the summation property and local minimality property of Fréchet subdifferential (see (Li et al., 2020a) for detailed descriptions). For any Y' such that

$$Y' \in \arg \min_Y \frac{1}{2\alpha} \|Y - (Z - \alpha \nabla \ell_Y^s(Z))\|_\mathcal{F}^2 + \phi(Y), \tag{27}$$

we have

$$0 \in \partial_Y \left[\frac{1}{2\alpha} \|Y - (Z - \alpha \nabla \ell_Y^s(Z))\|_\mathcal{F}^2 + \phi(Y) \right] \Bigg|_{Y=Y'}. \tag{28}$$

It follows that

$$0 \in \nabla \ell_Y^s(Z) + \partial_{Y'} \phi(Y'). \tag{29}$$

So, if $Y' = Z$, we have

$$0 \in \nabla \ell_Y^s(Z) + \partial_Z \phi(Z) = \partial_Z [\ell_Y^s(Z) + \phi(Z)] = \partial_Z \ell_Y(Z) \tag{30}$$

∎

With Lemma 11, to prove that UGNN converges to a stationary point, we only need to show it has a fixed point.

For the convergence of UGNN, we assume ϕ is continuous and ℓ_Y is Lipschitz smooth (i.e. its gradient is Lipschitz continuous) with Lipschitz constant $\mathcal{L}$. And we further distinguish two cases 1) No other conditions provided 2) ϕ is convex and ℓ_Y is strongly convex. We will use different strategies to prove convergence and show that the conditions on the step-size that guarantee convergence are different in these two cases.

Note that, as mentioned in the main paper, in the first case, we also need to assume $\ell_Y(Y) \to \infty$ when $\|Y\| \to \infty$ to avoid a degenerate case that the process goes to infinite.

The Framework of Proving Convergence

To prove the convergence of the iteration of (8), we will use Zangwill's convergence theorem. For clarity, we state it next.

Proposition 12 (Zangwill's Convergence Theorem (Luenberger, 1984)) *Let $A : \mathcal{X} \to 2^\mathcal{X}$ be a set-valued function, $\mathcal{G} \subset \mathcal{X}$ be a solution set (which can be any set we are interested in). For any sequence $\{x_k\}_{k=1}^\infty$ such that $x_{k+1} \in A(x_k)$, if the following conditions hold jointly:*

1. (boundedness) $\{x_k | 1 \le k \le \infty\}$ is contained in some compact subset of $\mathcal{X}$.

2. (descent function) there exists a continuous function ℓ such that

  (a) if $x \notin \mathcal{G}$, then $\forall y \in A(x), \ell(y) < \ell(x)$.

  (b) if $x \in \mathcal{G}$, then $\forall y \in A(x), \ell(y) \le \ell(x)$.

3. (closedness) the mapping A is closed at all point of $\mathcal{X} \setminus \mathcal{G}$ i.e. $\{(\boldsymbol{x}, \boldsymbol{y}) | \boldsymbol{x} \in \text{clos}\, (\mathcal{X} \setminus \mathcal{G})\, , \boldsymbol{y} \in A(\boldsymbol{x})\}$ is a closed set, where clos means set closure.

then the limit of any convergent subsequence of $\{\boldsymbol{x}_k\}_{k=1}^{\infty}$ is a solution i.e. inside $\mathcal{G}$.

Given Lemma 11, we define the solution set $\mathcal{G}$ as set of fixed points:

$$\mathcal{G} = \left\{ Z \,\middle|\, Z \in \arg\min_Y \frac{1}{2\alpha} \|Y - (Z - \alpha \nabla \ell_Y^s(Z))\|_{\mathcal{F}}^2 + \phi(Y) \right\}. \tag{31}$$

Then the proof is composed of three steps:

1. showing there exists some function that is a descent function;

2. showing the process is closed;

3. showing $\left\{ Y^{(k)} \right\}_{k=1}^{\infty}$ are bounded.

Thereinto, the closedness is a straightforward consequence of Lemma 6 in (Sriperumbudur and Lanckriet, 2009). For boundedness, as long as we find some descent function ℓ, we consider the set $\mathcal{S} = \left\{ Y \mid \ell(Y) \leq \ell\left(Y^{(0)}\right) \right\}$. Apparently $\left\{ Y^{(k)} \right\}_{k=1}^{\infty} \subseteq \mathcal{S}$, so we only need to show that $\mathcal{S}$ is bounded. It suffices to prove $\lim_{\|Y\| \to \infty} \ell(Y) = \infty$. Therefore, step 2 is already finished and step 3 is nearly finished. The most tricky part is proving descent of some function.

D.1 UGNN Convergence in General Case

Here we don't cast convexity assumption on the objective function. In this case we choose ℓ_Y as the candidate descent function. Note that by the assumption that $\lim_{\|Y\| \to \infty} \ell_Y(Y) = \infty$ made in main paper, we have boundedness of $\mathcal{S}$.

To prove descent, we first prove loosely descent (i.e. Lemma 1) and then show when $Y^{(k)} \notin \mathcal{G}$, it is also strict.

Proof of Lemma 1 Here we first state a property of Lipschitz smooth function:

Proposition 13 ((Bertsekas, 1999)) *for any $\ell(\boldsymbol{y})$ whose gradient is $\mathcal{L}$-Lipschitz, it satisfies*

$$\forall \boldsymbol{y}, \boldsymbol{z}, \ell(\boldsymbol{y}) \leq \ell(\boldsymbol{z}) + \nabla \ell(\boldsymbol{z})^\top (\boldsymbol{y} - \boldsymbol{z}) + \frac{\mathcal{L}}{2} \|\boldsymbol{y} - \boldsymbol{z}\|_2^2. \tag{32}$$

From Proposition 13, if function ℓ is $\mathcal{L}$-Lipschitz smooth, then for any $\boldsymbol{z}, \boldsymbol{y}$, we have

$$\ell(\boldsymbol{y}) \leq \ell(\boldsymbol{z}) + \nabla \ell(\boldsymbol{z})^\top (\boldsymbol{y} - \boldsymbol{z}) + \frac{\mathcal{L}}{2} \|\boldsymbol{y} - \boldsymbol{z}\|_2^2 \tag{33}$$

$$= \ell(\boldsymbol{z}) + \nabla \ell(\boldsymbol{z})^\top \boldsymbol{y} - \nabla \ell(\boldsymbol{z})^\top \boldsymbol{z} + \frac{\mathcal{L}}{2} \|\boldsymbol{y}\|_2^2 + \frac{\mathcal{L}}{2} \|\boldsymbol{z}\|_2^2 - \mathcal{L} \boldsymbol{y}^\top \boldsymbol{z} \tag{34}$$

$$= \ell(\boldsymbol{z}) - \frac{1}{2\mathcal{L}} \|\nabla \ell(\boldsymbol{z})\|_2^2$$
$$+ \frac{\mathcal{L}}{2} \left[\frac{1}{\mathcal{L}^2} \|\nabla \ell(\boldsymbol{z})\|_2^2 + \|\boldsymbol{y}\|_2^2 + \|\boldsymbol{z}\|_2^2 + \frac{2}{\mathcal{L}} \nabla \ell(\boldsymbol{z})^\top \boldsymbol{y} - \frac{2}{\mathcal{L}} \nabla \ell(\boldsymbol{z})^\top \boldsymbol{z} - 2 \boldsymbol{y}^\top \boldsymbol{z} \right] \tag{35}$$

$$= \ell(\boldsymbol{z}) - \frac{1}{2\mathcal{L}} \|\nabla \ell(\boldsymbol{z})\|_2^2 + \frac{\mathcal{L}}{2} \left\| \boldsymbol{y} - \left[\boldsymbol{z} - \frac{1}{\mathcal{L}} \nabla \ell(\boldsymbol{z}) \right] \right\|_2^2. \tag{36}$$

37

Taking $\ell(\boldsymbol{y}) = \ell_Y^s(\boldsymbol{y})$, $\boldsymbol{y} = \text{vec}(Y)$ and $\boldsymbol{z} = \text{vec}\left(Y^{(k)}\right)$, and let

$$\beta_Y\left(Y^{(k)}\right) = \ell_Y^s\left(Y^{(k)}\right) - \frac{\alpha}{2}\left\|\nabla\ell_Y^s\left(Y^{(k)}\right)\right\|_{\mathcal{F}}^2, \tag{37}$$

we can define an upper-bound of ℓ_Y:

$$\ell_Y^{(\text{upp})}(Y; Y^{(k)}) = \frac{1}{2\alpha}\left\|Y - \left[Y^{(k)} - \alpha\nabla\ell\left(Y^{(k)}\right)\right]\right\|_{\mathcal{F}}^2 + \sum_i \phi(\boldsymbol{y}_i) + \beta_Y\left(Y^{(k)}\right) \tag{38}$$

$$= \frac{1}{2\alpha}\left\|Y - U^{(k)}\right\|_{\mathcal{F}}^2 + \sum_i \phi(\boldsymbol{y}_i) + \beta_Y\left(Y^{(k)}\right), \tag{39}$$

and adopting (36) we have that:

$$\ell_Y(Y) = \ell_Y^s(Y) + \sum_i \phi(\boldsymbol{y}_i) \leq \ell_Y^{(\text{upp})}. \tag{40}$$

(Notice that since we have assumed $\alpha \leq \frac{1}{\mathcal{L}}$, ℓ_Y is also $\frac{1}{\alpha}$-Lipschitz). It's not difficult to check $\ell_Y^{(\text{upp})}\left(Y^{(k)}; Y^{(k)}\right) = \ell_Y\left(Y^{(k)}\right)$. Also, since adding terms which is not related to Y does not affect the $\arg\min$ result in (8), we have that

$$Y^{(k+1)} \in \arg\min_Y \frac{1}{2\alpha}\|Y - U^{(k)}\|_{\mathcal{F}}^2 + \sum_i \phi(\boldsymbol{y}_i) \tag{41}$$

$$= \arg\min_Y \frac{1}{2\alpha}\|Y - U^{(k)}\|_{\mathcal{F}}^2 + \sum_i \phi(\boldsymbol{y}_i) + \beta_Y\left(Y^{(k)}\right) \tag{42}$$

$$= \arg\min_Y \ell_Y^{(\text{upp})}\left(Y; Y^{(k)}\right). \tag{43}$$

Then we have

$$\ell_Y\left(Y^{(k+1)}\right) \leq \ell_Y^{(\text{upp})}\left(Y^{(k+1)}; Y^{(k)}\right) \leq \ell_Y^{(\text{upp})}\left(Y^{(k)}; Y^{(k)}\right) = \ell_Y\left(Y^{(k)}\right), \tag{44}$$

which proves the the lemma. ∎

Furthermore, when $Y^{(k)} \notin \mathcal{G}$, i.e. $Y^{(k)} \notin \arg\min \ell_Y^{(\text{upp})}\left(Y; Y^{(k)}\right)$, we have that

$$\ell_Y^{(\text{upp})}\left(Y^{(k+1)}; Y^{(k)}\right) < \ell_Y^{(\text{upp})}\left(Y^{(k)}; Y^{(k)}\right), \tag{45}$$

and thus (44) holds strictly, which matches the condition of strictly descent given in Proposition 12.

D.2 UGNN Convergence Under Convexity

When ℓ_Y^s is strongly convex and ϕ is convex, we have already know the existence and uniqueness of stationary point (global optimum) Y^*, so by Lemma 11 the fixed point of iteration (8) must be Y^*, i.e. $\mathcal{G} = \{Y^*\}$. In this case, we choose the distance of current point to the fixed point $\ell(Y) = \|Y - Y^*\|_{\mathcal{F}}^2$ as the descent function. As long as descent is proved, the boundedness of $\ell(Y)$ is obvious.

We prove the descent of $\ell(Y)$ by using the fact that when ϕ and ℓ_Y^s are convex, the proximal operator prox_ϕ is non-expansive and the gradient step is contraction with small enough α.

For proximal operator, Proposition 1 in (Gribonval and Nikolova, 2020) has already shown that as long as ϕ is convex, prox_ϕ is non-expansive, i.e.

$$\forall \boldsymbol{x}, \boldsymbol{y}, \left\|\text{prox}_\phi(\boldsymbol{x}) - \text{prox}_\phi(\boldsymbol{y})\right\|_2 \leq \|\boldsymbol{x} - \boldsymbol{y}\|_2. \tag{46}$$

Thus we only need to prove gradient descent is contraction. To prove this, we first introduce another property of convex Lipschitz smooth function:

Proposition 14 (Theorem 2.1.5 of (Nesterov, 2003)) *For any function ℓ that is convex and whose gradient is $\mathcal{L}$-Lipschitz continuous, we have*

$$\forall \boldsymbol{x}, \boldsymbol{y}, [\nabla\ell(\boldsymbol{x}) - \nabla\ell(\boldsymbol{y})]^\top (\boldsymbol{x} - \boldsymbol{y}) \geq \frac{1}{\mathcal{L}}\|\nabla\ell(\boldsymbol{x}) - \nabla\ell(\boldsymbol{y})\|_2^2. \tag{47}$$

Then we prove the contraction of gradient descent:

Lemma 15 *for any function ℓ that is $\mathcal{L}$-Lipschitz smooth satisfies that*

$$\forall \boldsymbol{x}, \boldsymbol{y}, \|[\boldsymbol{x} - \alpha\nabla\ell(\boldsymbol{x})] - [\boldsymbol{y} - \alpha\nabla\ell(\boldsymbol{y})]\|_2^2 \leq (1 - \alpha^2\mathcal{L}^2 - 2\alpha\mathcal{L})\|\boldsymbol{x} - \boldsymbol{y}\|_2^2. \tag{48}$$

Proof From Proposition 14, we have that:

$$\|[\boldsymbol{x} - \alpha\nabla\ell(\boldsymbol{x})] - [\boldsymbol{y} - \alpha\nabla\ell(\boldsymbol{y})]\|_2^2 \tag{49}$$

$$=\|\boldsymbol{x} - \boldsymbol{y}\|_2^2 + \alpha^2\|\nabla\ell(\boldsymbol{x}) - \nabla\ell(\boldsymbol{y})\|_2^2 - 2\alpha(\boldsymbol{x} - \boldsymbol{y})^\top [\nabla\ell(\boldsymbol{x}) - \nabla\ell(\boldsymbol{y})] \tag{50}$$

$$\leq\|\boldsymbol{x} - \boldsymbol{y}\|_2^2 + \left(\alpha^2 - \frac{2\alpha}{\mathcal{L}}\right)\|\nabla\ell(\boldsymbol{x}) - \nabla\ell(\boldsymbol{y})\|_2^2 \tag{51}$$

$$\leq(1 - \alpha^2\mathcal{L}^2 - 2\alpha\mathcal{L})\|\boldsymbol{x} - \boldsymbol{y}\|_2^2, \tag{52}$$

which finishes the proof. ∎

Combining (46) and Lemma 15, we get following corollary:

Corollary 16 *For $\ell_Y(Y) = \ell_Y^s(Y) + \phi(Y)$, where ℓ_Y^s is strongly convex and $\mathcal{L}$-Lipschitz smooth, ϕ is convex, $\alpha < \frac{2}{\mathcal{L}}$, we have*

$$\left\|Y^{(k+1)} - Y^*\right\|_{\mathcal{F}} < \left\|Y^{(k)} - Y^*\right\|_{\mathcal{F}} \tag{53}$$

if $Y^{(k)} \neq Y^*$.

Proof Since Y^* is the fixed point, we have $Y^* \in \text{prox}_\phi[Y^* - \alpha\nabla\ell_Y^s(Y^*)]$, therefore by (46)

$$\left\|Y^{(k+1)} - Y^*\right\|_{\mathcal{F}} = \left\|\text{prox}_\phi\left[Y^{(k)} - \alpha\nabla\ell_Y^s\left(Y^{(k)}\right)\right] - \text{prox}_\phi[Y^* - \alpha\nabla\ell_Y^s(Y^*)]\right\|_{\mathcal{F}}^2 \tag{54}$$

$$\leq \left\|\left[Y^{(k)} - \alpha\nabla\ell_Y^s\left(Y^{(k)}\right)\right] - [Y^* - \alpha\nabla\ell_Y^s(Y^*)]\right\|_{\mathcal{F}}^2 \tag{55}$$

Since $\alpha < \frac{2}{\mathcal{L}}$, from Lemma 15 we have that when $Y^{(k)} \neq Y^*$,

$$\left\| \left[Y^{(k)} - \alpha \nabla \ell_Y^s \left(Y^{(k)} \right) \right] - \left[Y^* - \alpha \nabla \ell_Y^s (Y^*) \right] \right\|_{\mathcal{F}}^2 < \left\| Y^{(k)} - Y^* \right\|. \tag{56}$$

$\blacksquare$

Therefore, $\ell \left(Y^{(k)} \right) = \left\| Y^{(k)} - Y^* \right\|$ is strictly descent when $Y^{(k)} \neq Y^*$. Combining this and Proposition 12 we conclude that:

Theorem 17 *When $\ell_Y^s(Y)$ is $\mathcal{L}$-Lipschitz smooth and strongly convex, $\phi(Y)$ is continuous and convex, iteration (8) converges to the unique global minimum of ℓ_Y when $\alpha < \frac{2}{\mathcal{L}}$.*

D.3 Remarks

Note that, the upper-bound of α is different in the two cases. Actually, if there is no penalty term ϕ, i.e. no proximal step, in both cases the upper bound is $2/\mathcal{L}$, which is a standard result on gradient descent for smooth functions that can be found in any optimization textbook. To get some intuition, considering the upper bound derived in (39), although $\alpha = 1/\mathcal{L}$ is the optimal step size to minimize $\ell_Y^{(\text{upp})}$, with α relaxed to $\leq 2/\mathcal{L}$, $\ell_Y^{(\text{upp})}$ still goes down:

$$\ell_Y^{(\text{upp})} \left[Y^{(k)} - \alpha \nabla \ell_Y^s \left(Y^{(k)} \right); Y^{(k)} \right] \leq \ell_Y^{(\text{upp})} \left(Y^{(k)}; Y^{(k)} \right) = \ell_Y^s \left(Y^{(k)} \right). \tag{57}$$

However, this is no longer true when the proximal step is added. Thus in the latter case, we either cast a more strict bound on α to still guarantee descent of ℓ_Y, or add extra assumptions that the process is non-expansive to guarantee that the output of each iteration would not be pushed further away from the fixed point despite not ensuring the descent of ℓ_Y.

D.4 Other Related Proofs

Note that Lemma 1 and Theorem 7 has been proved in section D.1 [14]. Here we provide other missing proofs from Section 5.1.

Proof of Theorem 3 First, we will show that ℓ_Y^s is strongly convex under conditions given. We have

$$\text{vec} \left(\frac{\partial \ell_Y^s}{\partial Y} \right) = \text{vec} \left[YW_f^s - f(X; W) + \tilde{L}YW_p^s \right] \tag{58}$$

$$= \left(W_f^s \otimes I + W_p^s \otimes \tilde{L} \right) \text{vec}(Y) + \text{vec} \left[f(X; W) \right] \tag{59}$$

$$= \Sigma \text{vec}(Y) + \text{vec} \left[f(X; W) \right]. \tag{60}$$

Therefore the Hessian of ℓ_Y^s is

$$\frac{\partial^2 \ell_Y^s}{\partial \text{vec}(Y)^2} = \Sigma. \tag{61}$$

We also know that when $\lambda_{\min}(\Sigma) > 0$, ℓ_Y^s is strongly convex respect to Y, thus has a unique global minimum. And from Proposition 1 in (Gribonval and Nikolova, 2020), we know ϕ is convex through the non-expansiveness of σ.

---

14. The last part of Theorem 7 that $\lim_{k \to \infty} \ell_Y \left(Y^{(k)} \right) = \ell_Y \left(Y^* \right)$ follows directly from the continuity of ℓ_Y.

Furthermore, the $\lambda_{\max}(\Sigma)$ is the Lipschitz constant of ℓ_Y^s, and from theorem 17 we know when $\alpha < 2/\lambda_{\max}(\Sigma)$ the algorithm converges to the stationary point, which is the global optimum. ∎

Proof of Corollary 4 Taking $W_f^s = I - W_p^s$ and $P = I - \tilde{L}$, we have that

$$\Sigma = I - P \otimes W_p^s. \tag{62}$$

By the spectral properties of Kronecker product, we have

$$\lambda_{\min}(\Sigma) \geq 1 - \left\|W_p^s\right\|_2 \|P\|_2 > 0 \tag{63}$$

and

$$\lambda_{\max}(\Sigma) \leq 1 + \left\|W_p^s\right\|_2 \|P\|_2 < 2. \tag{64}$$

The discussion in Section 4.3 mentioned that to get (11), we need to set $\alpha = 1 < \frac{2}{\lambda_{\max}(\Sigma)}$, therefore from Theorem 3 we conclude that (11) converges to the global minimum of (12). ∎

Proof of Lemma 5 (Convergence of IGNN) We first convert (2) to the vector form:

$$\text{vec}\left(Y^{(k+1)}\right) = \sigma\left[(W_p \otimes P)\,\text{vec}\left(Y^{(k)}\right) + \text{vec}(f)\right], \tag{65}$$

where $f = f(X; W_x)$, ignoring the parameters for simplicity. Let $M = W_p \otimes P$. From the spectral properties of knronecker product, we have

$$\|M\|_2 = \|W_p\|_2 \|P\|_2 < 1. \tag{66}$$

By definition of matrix norm we have

$$\forall \boldsymbol{x}_1, \boldsymbol{x}_2, \|M(\boldsymbol{x}_1 - \boldsymbol{x}_2)\|_2 \leq \|M\|_2 \|\boldsymbol{x}_1 - \boldsymbol{x}_2\|_2, \tag{67}$$

which means M (as a linear transformation) is a contraction mapping on Euclidean space. We also assumed σ is contraction mapping, and thus from Banach's fixed point theorem(Kinderlehrer and Stampacchia, 1980) we conclude that (65), as well as (2), has a unique fixed point. ∎

Appendix E. Proofs Related to UGNN Expressiveness

In this section, we provide detailed proofs of propositions in Section 6, as well as an analysis under cases where activation functions are non-linear.

E.1 Proof of Lemma 2

This lemma is a straight-forward corollary of Theorem 1 in (Gribonval and Nikolova, 2020). For clarity, we excerpt the theorem here:

Proposition 18 (Theorem 1 in (Gribonval and Nikolova, 2020)) *Let $\mathcal{H}$ be some Hilbert space (e.g. $\mathbb{R}^k$) and $\mathcal{Y} \subset \mathcal{H}$ be non-empty. A function $\sigma : \mathcal{Y} \to \mathcal{H}$ is the proximal operator of a function $\phi : \mathcal{H} \to \mathbb{R} \cup \{+\infty\}$ if and only if there exists a convex l.s.c. (lower semi-continuous) function $\psi : \mathcal{H} \to \mathbb{R} \cup \{+\infty\}$ such that for each $\boldsymbol{y} \in \mathcal{Y}, \sigma(\boldsymbol{y}) \in \partial\psi(\boldsymbol{y})$.*

Since our σ is component-wise, we only need to consider $\mathcal{Y} = \mathcal{H} = \mathbb{R}$. Furthermore, since $\sigma(\boldsymbol{x})$ is continuous, its indefinite integration exists. Let $\psi(\boldsymbol{x}) = \int \sigma(\boldsymbol{x})\mathrm{d}\boldsymbol{x}$, which must be convex since its derivative is non-decreasing. Then according to Proposition 18, there exists a function ϕ whose proximal operator is σ. Conversely, if there exists ϕ such that $\sigma = \mathrm{prox}_\phi$, from Proposition 18 it is the subdifferential of some convex function ψ, which means it is non-decreasing.

E.2 Proof of Lemma 6

We only need a counterexample to prove this. Suppose $n = 1$ and $d = 2$, consider $W = \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix}$, $Y = \begin{bmatrix} a & b \end{bmatrix}$, then $YW = \begin{bmatrix} 0 \\ a \end{bmatrix}$.

If there exists any second order smooth function h such that $\frac{\partial h}{\partial a} = 0$ and $\frac{\partial h}{\partial b} = a$, we have that

$$\frac{\partial^2 h}{\partial a \partial b} = 0 \neq \frac{\partial^2 h}{\partial b \partial a} = 1,$$

which contradicts the second order smoothness assumption of h.

E.3 Proof of Theorem 9

In this subsection we denote $\hat{X} = f(X; W_x)$ for simplicity. First consider a case which allows T and W_p^s to have complex values. Hereinafter we denote the space of all complex-valued matrices with shape $d_1 \times d_2$ by $\mathbb{C}^{d_1 \times d_2}$.

We recall Jordan's decomposition theorem below. Let $J(\lambda)$ be Jordan block matrix

$$J(\lambda) = \begin{bmatrix} \lambda & 1 & & & \\ & \lambda & 1 & & \\ & & \lambda & & \\ & & & \ddots & 1 \\ & & & & \lambda \end{bmatrix}.$$

We have the following well-known fact.

Proposition 19 (Jordan) *For every $W \in \mathbb{R}^{d \times d}$, there exists a complex-valued invertible matrix $P \in \mathbb{C}^{d \times d}$ and complex-valued block-diagonal matrix*

$$\Omega = \begin{bmatrix} J(\lambda_1) & & & \\ & J(\lambda_2) & & \\ & & \ddots & \\ & & & J(\lambda_k) \end{bmatrix}$$

such that $W = P\Omega P^{-1}$, where $\lambda_j \in \mathbb{C}$ is the j-th eigenvalue of W and the size of $J(\lambda_j)$ is the algebraic multiplicity of λ_j.

Corollary 20 *If a square matrix $W \in \mathbb{R}^{d \times d}$ has d distinct eigenvalues, then there exists $P, \Lambda \in \mathbb{C}^{d \times d}$ where P is invertible and Λ is diagonal, such that $W = P\Lambda P^{-1}$.*

Then, we shall show that actually on the complex domain "almost" every matrix can be diagonalized, which means, for each matrix W, either it itself can be diagonalized, or there's a diagonalizable matrix W' that is arbitrarily closed to W.

Corollary 21 *For every square matrix $W \in \mathbb{R}^{d \times d}$ and any $\epsilon > 0$, there exists a diagonalizable square matrix $W' \in \mathbb{R}^{d \times d}$ such that $\|W' - W\| \leq \epsilon$.*

Proof Let $W = R\Omega R^{-1}$ and $E = \text{diag} \begin{bmatrix} \epsilon_1, & \epsilon_2, & \cdots, & \epsilon_n \end{bmatrix}$ such that the diagonal of $\Omega + E$ is distinct and $\epsilon_j^2 < \frac{\epsilon^2}{d\|R\|^2\|R^{-1}\|^2}$.

Since Ω is upper-triangle, its eigenvalues are the elements in its diagonal, which are distinct. Thus according to Corollary 20, $\Omega + E$ is diagonalizable, suppose $\Omega + E = Q\Lambda Q^{-1}$. Let $W' = R(\Omega + E)R^{-1} = RQ\Lambda Q^{-1}R^{-1}$, it is apparent W' also diagonalizable.

Now consider the difference between W' and W. We have $\|W' - W\|^2 = \|PEP^{-1}\|^2 \leq \|E\|^2\|R\|^2\|R^{-1}\|^2 < \epsilon^2$. ∎

Note that the norm in Corollary 21 can be any matrix norm.

Lemma 22 *The solution to the equation $PYW + \hat{X} = Y$ is continuous respect to W as long as a unique solution exists.*

Proof The solution of this equation is $\text{vec}(Y) = \left(I - W^\top \otimes P\right)^{-1}\text{vec}(\hat{X})$, which is continuous. ∎

Theorem 23 *For any $W_p \in \mathbb{R}^{d \times d}$ that admits a unique fixed point for IGNN and $\hat{X} \in \mathbb{R}^{n \times d}$, suppose $Y \in \mathbb{R}^{n \times d}$ is the only solution of $PYW + \hat{X} = Y$, then there exists a $Y' \in \mathbb{C}^{n \times d}, \bar{W}_x \in \mathbb{C}^{d \times d'}$, right-invertible $\bar{T} \in \mathbb{C}^{d \times d'}$ and hermitian $\bar{W}_p^s \in \mathbb{C}^{d' \times d'}$, such that*

$$PY'\bar{T}\bar{W}_p^s + \hat{X}\bar{W}_x = Y'\bar{T}, \text{ with } \|Y' - Y\| < \epsilon, \forall \epsilon > 0, Y' \in \mathbb{C}^{n \times d}. \tag{68}$$

Proof By Corollary 21 and Lemma 22, there exists diagonalizable $W' \in \mathbb{R}^{d \times d}$ such that $PY'W' + \hat{X} = Y'$ for some Y' satisfies $\|Y' - Y\| < \epsilon$.

Suppose now $W' = R\Lambda R^{-1}$ where Λ is diagonal and R is invertible. We have

$$PY'R\Lambda R^{-1} + \hat{X} = Y', \tag{69}$$

and right-multiply by R at each side of this equation we get

$$P(Y'R)\Lambda + \hat{X}R = Y'R. \tag{70}$$

By choosing $\bar{T} = \bar{W}_x = R$ and $\bar{W}_p^s = \Lambda$, we proved the theorem. ∎

Next, we consider restricting this result to the real domain. This is straight-forward since every complex linear transformation can be converted to real linear transformation by consider real and imaginary parts separately. Hereinafter, we denote the real part of an complex-valued matrix M by $M_{(r)}$ and imaginary part by $M_{(i)}$.

Proof of Theorem 9 Let $\bar{Y}, \bar{T}, \bar{W}_p^s, \bar{W}_x$ be the matrices obtained from Theorem 23. Note the last three are complex-valued, $\bar{T}$ is invertible and $\bar{W}_p^s$ is hermitian.

Consider $T = \begin{bmatrix} \bar{T}_{(r)} & \bar{T}_{(i)} \end{bmatrix}$. Since $\bar{T}$ is invertible, let $\bar{L} = \bar{T}^{-1}$, then we have $I = (\bar{T}\bar{L})_{(r)} = \bar{T}_{(r)}\bar{L}_{(r)} - \bar{T}_{(i)}\bar{L}_{(i)}$. Thus $T \begin{bmatrix} \bar{L}_{(r)} \\ -\bar{L}_{(i)} \end{bmatrix} = \bar{T}_{(r)}\bar{L}_{(r)} - \bar{T}_{(i)}\bar{L}_{(i)} = I$. This means T is right-invertible.

Then let $W_s^p = \begin{bmatrix} \bar{W}_{p(r)}^s & \bar{W}_{p(i)}^s \\ -\bar{W}_{p(i)}^s & \bar{W}_{p(r)}^s \end{bmatrix}$. Since $\bar{W}_p^s$ is hermitian, we have that $\bar{W}_{p(r)}^s = \bar{W}_{p(r)}^{s\top}$ and $\bar{W}_{p(i)}^s = -\bar{W}_{p(i)}^{s\top}$, which ensures W_p^s constructed here is symmetric. Finally, let $\tilde{W}_x = \begin{bmatrix} \bar{W}_{x(r)} & \bar{W}_{x(i)} \end{bmatrix}$. Now it's easy to verify that $PY'TW_p^s + \hat{X}\tilde{W}_x = Y'T$. $\blacksquare$

E.4 Proof of Theorem 10

First we will illustrate that by setting the parameters in UGNN to properly, the iteration step from (8) is equivalent to a GCN layer with symmetric shared weights, with or without residual connections. We set ρ to be identity (so that $\Gamma = I$), $\tilde{B}$ satisfies that $\tilde{B}^\top \tilde{B} = I - P$ (e.g. if $P = \hat{A} = \tilde{D}^{-1/2}\tilde{A}\tilde{D}^{-1/2}$ is normalized adjacency, then $\tilde{B}$ is normalized incidence matrix $\tilde{B} = \tilde{D}^{-1/2}B$).

For a GCN model without residual connections, let $W_f^s = I - W_p^s$, then we have $Y^{(k+1)} = \sigma(PY^{(k)}W_p^s)$. Consider $Y^{(0)} = W \begin{bmatrix} Y_{\text{GCN}}^{(0)} & 0 & 0 & \cdots & 0 \end{bmatrix}$ where $Y_{\text{GCN}}^{(0)}$ is the input of GCN, and the ith 0 here denote a block matrix of all zero with the same size as $Y_{\text{GCN}}^{(i)}$. When we let

$$W_p^s = \begin{bmatrix} & W_p^{(1)} & & & & & \\ W_p^{(1)\top} & & W_p^{(2)} & & & & \\ & W_p^{(2)\top} & & W_p^{(3)} & & & \\ & & W_p^{(3)\top} & & \ddots & & \\ & & & & \ddots & & W_p^{(k)} \\ & & & & & W_p^{(k)\top} & \end{bmatrix}, \tag{71}$$

it is not difficult to verify that the kth block of $Y^{(k)}$ is $Y_{\text{GCN}}^{(k)}$. Note the 0s in the (71) are block matrices of all zeros with proper size.

For GCN model with residual connection, let $W_f^s = I - W_p^s - W_r^s$, then $Y^{(k+1)} = \sigma(PY^{(k)}W_p^s + Y^{(k)}W_r^s)$. We take the same W_p^s as before (71), and let

$$W_r^s = \begin{bmatrix} & I & & & & & \\ I & & I & & & & \\ & I & & I & & & \\ & & I & & \ddots & & \\ & & & & \ddots & & I \\ & & & & & I & \end{bmatrix}. \tag{72}$$

Since W_r^s and W_p^s are both symmetric, apparently W_f^s is also symmetric. Also notice that since residual connection exists, the size of all the $W_p^{(i)}$s are the same, so the size of I in the (72) is the same as $W_p^{(i)}$.

E.5 A Sufficient Condition of Equivalence of Symmetric and Asymmetric Weights

In this section, we give a sufficient condition of fixed-point equivalence of symmetric and asymmetric weights in general case as mentioned in Section 6.1. Here we consider general nonlinear functions σ and penalty term ϕ. Hereinafter we denote prox_ϕ by $\tilde{\sigma}$.

Constraint on σ To simplify the discussion, we add a mild assumption on σ and consider a general type of $\tilde{\sigma}$: We assume $\sigma(\boldsymbol{x})$ is a proximal operator of some penalty function which is differentiable with respect to x.

This assumption is consistent with practice. Firstly, commonly used non-linearities like $\tanh$, sigmoid and ReLU are all proximal operators since they are all continuous and component-wise non-decreasing (By Lemma 2). Also, although at some point ReLU and Leaky-ReLU are not differentiable, we can approximate them by

$$\text{ReLU}(x) \approx \frac{1}{r}\log\left(\exp(rx) + 1\right)$$

and

$$\text{Leaky-ReLU}(x, p) \approx \frac{1}{r}\log\left(\exp(rx) + \exp(prx)\right)$$

with big enough r, which are differentiable and do not affect their practical attributes.

$\tilde{\sigma}$ ***Considered*** As discussed in the main paper, it would cause some degenerated cases if we allow a general class of proximal operators. Also it would too difficult to handle in this case. Therefore, we only consider proximal operators in a special while general family

$$\mathcal{S} = \{\tilde{\sigma} : \boldsymbol{x} \mapsto G\sigma(C\boldsymbol{x}) | G, C \text{ are chosen such that } \tilde{\sigma} \text{ is proximal operator}\}$$

Now we consider under what value of G and C we can ensure $\tilde{\sigma}$ is proximal operator. From Proposition 18, we know $\exists\phi, \tilde{\sigma} = \text{prox}_\phi$ iff $\tilde{\sigma}$ is subgradient of some l.s.c convex function. Since we have assumed σ is continuous and differentiable, $\int \sigma(\boldsymbol{x})\mathrm{d}\boldsymbol{x}$ is twice differentiable. We know that a twice-differentiable function ψ is convex iff its Hessian H_ψ is positive semi-definite. Therefore the Hessian of σ must be positive semi-definite, we denote it by $H(\boldsymbol{x}) = \frac{\partial\sigma(\boldsymbol{x})}{\partial\boldsymbol{x}}$, and so do Hessian of $\tilde{\sigma}(\boldsymbol{x})$, which we denote by $\tilde{H}(\boldsymbol{x}) = \frac{\partial\tilde{\sigma}(\boldsymbol{x})}{\partial\boldsymbol{x}}$. From the chain rule we have

$$\tilde{H}(\boldsymbol{x}) = \frac{\partial\tilde{\sigma}(\boldsymbol{x})}{\partial\boldsymbol{x}} = \frac{\partial G\sigma(C\boldsymbol{x})}{\partial\boldsymbol{x}} = G\frac{\partial\sigma(C\boldsymbol{x})}{\partial C\boldsymbol{x}}\frac{\partial C\boldsymbol{x}}{\partial\boldsymbol{x}} = GH(C\boldsymbol{x})C.$$

From the deduction above, we can conclude that:

Lemma 24 $\tilde{\sigma} : \mathbb{R}^d \to \mathbb{R}^d, \boldsymbol{x} \mapsto G\sigma(C\boldsymbol{x})$ *is proximal operator iff* $\forall \boldsymbol{x} \in C\mathbb{R}^d, GH(\boldsymbol{x})C$ *is positive semi-definite where* $H(\boldsymbol{x}) = \frac{\partial\sigma(\boldsymbol{x})}{\partial\boldsymbol{x}}$.

The Condition of General Fixed-Point Alignment

Theorem 25 *For σ satisfying the assumptions above, $W_p \in \mathbb{R}^{d\times d}$ that admits unique fixed point for IGNN, and $W_x \in \mathbb{R}^{d_0\times d}$, suppose Y is the only solution of $Y = \sigma(PYW + XW_x)$, a sufficient condition of*

$$\exists\phi, \text{ right invertible } T \in \mathbb{R}^{d\times d'}, \text{ symmetric } W_p^s \in \mathbb{R}^{d'\times d'} \text{ and } \tilde{W}_x \in \mathbb{R}^{d\times d} \text{ such that}$$
$$\tilde{\sigma}(PYTW_p^s + XW_x) = YT$$

is that

$$\exists \textit{ right-invertible } T \in \mathbb{R}^{d \times d'}, C \in \mathbb{R}^{d' \times d}, W_p^s \in \mathbb{R}^{d' \times d'} \textit{ such that } TW_p^s C = W_p$$
$$\textit{and } \forall \boldsymbol{x} \in C^\top \mathbb{R}, T^\top H(\boldsymbol{x}) C^\top \textit{ is P.S.D,}$$

where $\tilde{\sigma}(\boldsymbol{x}) = \mathrm{prox}_\phi(\boldsymbol{x})$ *and* $H(\boldsymbol{x}) = \frac{\partial \sigma(\boldsymbol{x})}{\partial \boldsymbol{x}}$.

Proof   We have assumed $\tilde{\sigma} \in \mathcal{S}$, thus $\tilde{\sigma}(\boldsymbol{x}) = G\sigma(C\boldsymbol{x})$. We want to prove that

$$YT = \tilde{\sigma}(PYTW_p^s + X\tilde{W}_x) \tag{73}$$
$$= \sigma(PYTW_p^s C + X\tilde{W}_x C)G \tag{74}$$
$$= \sigma(PYTW_p^s C + XW_x)G \qquad (\text{Assume } \tilde{W}_x C = W_x). \tag{75}$$

A sufficient condition for this equation to hold is $T = G$ and $Y = \sigma(AYTW_p^s C + XW_x) = \sigma(AYW_p + XW_x)$. Comparing the last two terms, apparently it holds when $TW_p^s C = W_p$. And to ensure $\tilde{\sigma} \in \mathcal{S}$, from Lemma 24 we know it means $T^\top H(C^\top x)C^\top$ be positive semi-definite for all x. $\blacksquare$

We can verify this Theorem 25 by linear case. If σ is linear, then $H(Cx) = I$, thus simply taking $C = T^{-1}$ (on complex field) we have $TW_p^s T^{-1}$ can generate any real matrix W and $T^\top H(C^\top x)C^\top = T^\top T^{-\top} = I$ is positive semi-definite.