Abstract

State-of-the-art Graph Neural Networks (GNNs) have limited scalability with respect to the graph and model sizes. On large graphs, increasing the model depth often means exponential expansion of the scope (i.e., receptive field). Beyond just a few layers, two fundamental challenges emerge: 1. degraded expressivity due to oversmoothing, and 2. expensive computation due to neighborhood explosion. We propose a design principle to decouple the depth and scope of GNNs – to generate representation of a target entity (i.e., a node or an edge), we first extract a localized subgraph as the bounded-size scope, and then apply a GNN of arbitrary depth on top of the subgraph. A properly extracted subgraph consists of a small number of critical neighbors, while excluding irrelevant ones. The GNN, no matter how deep it is, smooths the local neighborhood into informative representation rather than oversmoothing the global graph into “white noise”. Theoretically, decoupling improves the GNN expressive power from the perspectives of graph signal processing (GCN), function approximation (GraphSAGE) and topological learning (GIN). Empirically, on seven graphs (with up to 110M nodes) and six backbone GNN architectures, our design achieves significant accuracy improvement with orders of magnitude reduction in computation and hardware cost.

1 Introduction

Graph Neural Networks (GNNs) have now become the state-of-the-art models for graph mining [48, 13, 58], facilitating applications such as social recommendation [35, 52, 37], knowledge understanding [40, 38, 59] and drug discovery [43, 32]. With the numerous architectures proposed [22, 12, 44, 49], it still remains an open question how to effectively scale up GNNs with respect to both the model size and graph size. There are two fundamental obstacles when we increase the number of GNN layers:

• Expressivity challenge (i.e., oversmoothing [30, 36, 39, 17]): iterative mixing of neighbor features collapses embedding vectors of different nodes into a fixed, low-dimensional subspace.
• Scalability challenge (i.e., neighbor explosion [7, 8, 9, 55]): recursive expansion of multi-hop neighborhood results in exponentially growing receptive field size (and thus computation cost).

To address the expressivity challenge, most remedies focus on neural architecture exploration: [44, 12, 49, 29] propose more expressive aggregation functions when propagating neighbor features. [50, 28, 18, 34, 1, 33, 31] use residue-style design components to construct flexible and dynamic receptive fields. Among them, [50, 28, 18] use skip-connection across multiple GNN layers, and [34, 1, 33, 31] encourage multi-hop message passing within each single layer. As for the scalability challenge, sampling methods have been explored to improve the training speed and efficiency. Importance based layer-wise sampling [8, 7, 61] and subgraph-based sampling [54, 9, 55] alleviate neighbor explosion, while preserving training accuracy. Unfortunately, such sampling methods cannot be naturally generalized to inference without accuracy loss (see also Section 4).

The above lines of research have only guided us to partial solutions. Yet what is the root cause of both the expressivity and scalability challenges? Setting aside the design of GNN architectures or sampling schemes, we provide an alternative perspective by interpreting the data in a different way.

Two views on the graph. Given an input graph \( G \) with node set \( V \), the most straightforward way to understand \( G \) is by viewing it as a single global graph. So any two nodes \( u \) and \( v \) belong to the same \( G \), and if \( u \) and \( v \) lie in the same connected component, they will ultimately see each other in their own neighborhood no matter how far away \( u \) and \( v \) are. Alternative to the above global view, we can take a local view on \( G \). For each node \( v \), there is a latent \( G_{[v]} \) surrounding it which captures the characteristics of just \( v \) itself. The full \( G \) is observed (by the data collection process) as the union of all such \( G_{[v]} \). Consequently, \( V_{[v]} \) rather than \( V \) defines \( v \)'s neighborhood: if \( u \notin V_{[v]} \), \( v \) will never consider \( u \) as a neighbor. Our “decoupling” design is based on the local view.

Scope of GNNs. Both the expressivity and scalability challenges are closely related to the enlargement of the GNN’s scope (i.e., receptive field). More importantly, how we define the scope is determined by how we view \( G \). With the global view above, an \( L \)-layer GNN has the scope of the full \( L \)-hop neighborhood. With the local view, the GNN scope is simply \( V_{[v]} \) regardless of the GNN depth. The two existing lines of research, one on architectural exploration and the other on sampling, both take the global view. Consequently, the depth (i.e., number of layers) and scope of such GNNs are tightly coupled. Such coupling significantly limits the design space exploration of GNNs with various depths [53]. Consider the example of ogbn-products, a medium-scale graph in Open Graph Benchmark [16]. The average number of 4-hop neighbors is around 0.6M, corresponding to 25% of the full graph size. To generate representation of a single target node, a 4-layer coupled GNN needs to propagate features from the 0.6M neighbors. Such propagation can be inefficient or even harmful since most nodes in the huge neighborhood would be barely relevant to the target node.

Decoupling the GNN depth and scope. Taking the local view on \( G \), we propose a general design principle to decouple the GNN depth and scope. To generate the representation of the target node \( v \), we first extract from \( G \) a small subgraph \( G_{[v]} \) surrounding \( v \). On top of \( G_{[v]} \), we apply a GNN whose number of layers and message passing functions can be flexibly chosen. “Decoupling” means we treat the scope extraction function and GNN depth as two independently tuned parameters – effectively we introduce a new dimension in the GNN design space. We intuitively illustrate the benefits of decoupling by an example GNN construction, where the scope is the \( L \)-hop neighborhood and depth is \( L' \) (\( L' > L \)). When we use more layers (\( L' \)) than hops (\( L \)), each pair of subgraph nodes may exchange messages multiple times. The extra message passing helps the GNN better absorb and embed the information within scope, and thus leads to higher expressivity. We further justify the above intuition with multifaceted theoretical analysis. From the graph signal processing perspective, we prove that decoupled-GCN performs local-smoothing rather than oversmoothing, as long as the scopes of different target nodes are different. From the function approximation perspective, we construct a linear target function on neighbor features and show that decoupling the GraphSAGE model reduces the function approximation error. From the topological learning perspective, we apply deep GIN-style message passing to differentiate non-regular subgraphs of a regular graph. As a result, our model is more powerful than the 1-dimensional Weisfeiler-Lehman test [41].

Practical implementation: \textsc{Shadow-GNN}. The decoupling principle leads to a practical implementation, \textsc{Shadow-GNN}: Decoupled GNN on a shallow subgraph. In \textsc{Shadow-GNN}, the scope is a shallow yet informative subgraph, only containing a fraction of the 2- or 3-hop neighbors of \( G \) (see Section 5). On the other hand, the model of \textsc{Shadow-GNN} is deeper (e.g., \( L' = 5 \)). To efficiently construct the shallow scope on commodity hardware, we propose various subgraph
“Decoupling” is a design principle to improve the expressivity and scalability of GNNs. It essentially means that we decouple the depth and scope of the GNN. By doing so, we can improve accuracy and scalability. On seven benchmarks, including the largest ogbn-papers100M graph with 111M nodes, SHADOW-GNNs achieve significant accuracy gains compared to the original models. Meanwhile, the computation and hardware costs are reduced by orders of magnitude. Our code is available at https://github.com/facebookresearch/shadow_gnn.

2 Preliminaries

Let \( G(V, E, X) \) be an undirected graph, with node set \( V \), edge set \( E \subseteq V \times V \) and node feature matrix \( X \in \mathbb{R}^{V \times d} \). Let \( N_u \) denote the set of \( u \)'s direct neighbors in \( G \). The \( u \)-th row of \( X \) corresponds to a length-\( d \) feature of node \( u \). Let \( A \) be the adjacency matrix of \( G \) where \( A_{u, v} = 1 \) if edge \((u, v) \in E\) and \( A_{u, v} = 0 \) otherwise. Let \( D \) be the diagonal degree matrix of \( A \). Denote \( \hat{A} = D^{-\frac{1}{2}} A D^{-\frac{1}{2}} \) as the adjacency matrix after symmetric normalization ("*" means augmented with self-edges), and \( \hat{A} = D^{-1} A \) (or \( D^{-1} A \)) as the one after random walk normalization. Let subscript "[u]" mark the quantities corresponding to a subgraph surrounding node \( u \). For example, the subgraph itself is \( G_{[u]} \). Subgraph matrices \( X_{[u]} \) and \( A_{[u]} \) have the same dimension as the original \( X \) and \( A \). Yet, row vector \( [X_{[u]}]_v = 0 \) for \( u \notin V_{[u]} \). Element \( [A_{[u]}]_{u, v} = 0 \) if either \( u \notin V_{[u]} \) or \( w \notin V_{[u]} \).

For an \( L \)-layer GNN, let superscript "(\( \ell \))" denote the layer-\( \ell \) quantities. Let \( d^{(\ell)} \) be the number of channels for layer \( \ell \); \( H^{(\ell)} \in \mathbb{R}^{V \times d^{(\ell-1)}} \) and \( H^{(\ell)} \in \mathbb{R}^{V \times d^{(\ell)}} \) be the input and output features. So \( H^{(0)} = X \) and \( d^{(0)} = d \). Further, let \( \sigma \) be the activation and \( W^{(\ell)} \) be the learnable weight. For example, a GCN layer performs \( H^{(\ell)} = \sigma \left( \hat{A} H^{(\ell-1)} W^{(\ell)} \right) \). A GraphSAGE layer performs \( H^{(\ell)} = \sigma \left( H^{(\ell-1)} W_1^{(\ell)} + \hat{A} H^{(\ell-1)} W_2^{(\ell)} \right) \).

Our analysis in Section 3 mostly focuses on the node classification task. Yet our design can be generalized to the link prediction task, as demonstrated by our experiments in Section 5.

**Definition 2.1. (Depth of subgraph)** Assume the subgraph \( G_{[v]} \) is connected. The depth of \( G_{[v]} \) is defined as \( \max_{u \in V_{[v]}} d(u, v) \), where \( d(u, v) \) denotes the shortest path distance from \( u \) to \( v \).

The above definition enables us to make comparison such as "the GNN is deeper than the subgraph". For "decoupling the depth and scope", we refer to the model depth rather than the subgraph depth.

3 Decoupling the Depth and Scope of GNNs

"Decoupling the depth and scope of GNNs" is a design principle to improve the expressivity and scalability of GNNs without modifying the layer architecture. We name a GNN after decoupling a SHADOW-GNN (see Section 3.6 for explanation of the name). Compared with a normal GNN, SHADOW-GNN contains an additional component: the subgraph extractor EXTRACT. To generate embedding of a target node \( v \), SHADOW-GNN proceeds as follows: 1. We use EXTRACT \((v, G)\) to return a connected \( G_{[v]} \), where \( G_{[v]} \) is a subgraph containing \( v \), and the depth of \( G_{[v]} \) is \( L \). 2. We build an \( L' \)-layer GNN on \( G_{[v]} \) by treating \( G_{[v]} \) as the new full graph and by ignoring all nodes / edges not in \( G_{[v]} \). So \( G_{[v]} \) is the scope of SHADOW-GNN. The key point reflecting “decoupling” is that \( L' > L \).

A normal GNN is closely related to a SHADOW-GNN. Under the normal setup, an \( L \)-layer GNN operates on the full \( G \) and propagates the influence from all the neighbors up to \( L \) hops away from \( v \). Such a GNN is equivalent to a model where EXTRACT returns the full \( L \)-hop subgraph and \( L' = L \).

We theoretical demonstrate how SHADOW-GNN improves expressivity from three different angles. On SHADOW-GCN (Section 3.1), we come from the graph signal processing perspective. The GCN propagation can be interpreted as applying filtering on the node signals \([47]\). Deep models correspond to high-pass filters. Filtering the local graph \( G_{[v]} \) preserves richer information than the global \( G \). On SHADOW-SAGE (Section 3.2), we view the GNN as a function approximator. We construct a target function and study how decoupling reduces the approximation error. On SHADOW-GIN (Section 3.3), we focus on learning topological information. We show that decoupling helps capture local graph structure which the 1D Weisfeiler-Lehman test fails to capture.
3.1 Expressivity Analysis on SHADOW-GCN: Graph Signal Processing Perspective

GCNs [22] suffer from “oversmoothing” [30] – Each GCN layer smooths the features of the direct (i.e., 1-hop) neighbors, and many GCN layers smooths the features of the full graph. Eventually, such repeated smoothing process propagates to any target node just the averaged feature of all \( V \). “Oversmoothing” thus incurs significant information loss by wiping out all local information.

Formally, suppose the original features \( X \) reside in a high-dimensional space \( \mathbb{R}^{|V| \times d} \). Oversmoothing pushes \( X \) towards a low-dimensional subspace \( \mathbb{R}^{|V| \times d'} \), where \( d' < d \). Corresponding analysis comes from two perspectives: oversmoothing by a deep GCN, and oversmoothing by repeated GCN-style propagation. The former considers the full neural network with non-linear activation, weight and bias. The later characterizes the aggregation matrix \( M = \lim_{L \to \infty} A^L X \). It is shown that even with the vanilla architecture, a deep GCN with bias parameters does not oversmooth [17]. In addition, various tricks [60, 39, 34] can prevent oversmoothing from the neural network perspective. However, a deep GCN still suffers from accuracy drop, indicating that the GCN-style propagation (rather than other GCN components like activation and bias) may be the fundamental reason causing difficulty in learning. Therefore, we study the asymptotic behavior of the aggregation matrix \( M \) under the normal and SHADOW design. In other words, here in Section 3.1 we ignore the non-linear activation and bias parameters. Such setup is consistent with many existing literature such as [30, 33, 34, 60].

**Proposition 3.1.** \( \infty \) number of feature propagation by SHADOW-GCN leads to

\[
m_{[v]} = \left[ e_{[v]} \right]_v \cdot \left( e_{[v]}^T X_{[v]} \right)
\]

where \( e_{[v]} \) is defined by \( e_{[v]} = \frac{\delta_{[v]}(u)}{\sum_{u \in V_{[v]}} \delta_{[v]}(u)} \); \( \delta_{[v]}(u) \) returns the degree of \( u \) in \( G_{[v]} \) plus 1.

**Oversmoothing by normal GCN propagation.** With a large enough \( L \), the full \( L \)-hop neighborhood becomes \( \mathcal{V} \) (assuming connected \( G \)). So \( \forall u, v \), we have \( G_{[u]} = G_{[v]} = \mathcal{G} \), implying \( e_{[u]} = e_{[v]} \) and \( X_{[u]} = X_{[v]} = X \). From Proposition 3.1, the aggregation converges to a point where no feature and little structural information of the target is preserved. The only information in \( m_{[v]} \) is \( v \)’s degree.

**Local-smoothing by SHADOW-GCN propagation.** With a fixed subgraph, no matter how many times we aggregate using \( A_{[u]} \), the layers will not include the faraway irrelevant nodes. From Proposition 3.1, \( m_{[v]} \) is a linear combination of the neighbor features \( X_{[v]} \). Increasing the number of layers only pushes the coefficients of each neighbor features to the stationary values. The domain \( X_{[v]} \) of such linear transformation is solely determined by \( \text{EXTRACT} \) and is independent of the model depth. Intuitively, if \( \text{EXTRACT} \) picks non-identical subgraphs for two nodes \( u \) and \( v \), the aggregations should be different due to the different domains of the linear transformation. Therefore, SHADOW-GCN preserves local feature information whereas normal GCN preserves none. For structural information in \( m_{[v]} \), note that \( e_{[v]} \) is a normalized degree distribution of the subgraph around \( v \), and \( \left[ e_{[v]} \right]_v \) indicates the role of the target node in the subgraph. By simply letting \( \text{EXTRACT} \) return the 1-hop subgraph, \( \left[ e_{[v]} \right]_v \), alone already contains all the information preserved by a normal GCN, which is \( v \)'s degree in \( G \). For the general \( \text{EXTRACT} \), \( e_{[u]} \) additionally reflects \( v \)'s ego-net structure. Thus, a deep SHADOW-GCN preserves more structural information than a deep GCN.

**Theorem 3.2.** Let \( \overline{m}_{[v]} = \phi_G(v) \cdot m_{[v]} \) where \( \phi_G \) is any non-zero function only depending on the structural property of \( v \). Let \( \mathcal{M} = \{ \overline{m}_{[v]} \mid v \in \mathcal{V} \} \). Given \( G \), \( \text{EXTRACT} \) and some continuous probability distribution in \( \mathbb{R}^{|V| \times d} \) to generate \( X \), then \( \overline{m}_{[v]} \neq \overline{m}_{[u]} \) if \( V_{[u]} \neq V_{[v]} \), almost surely.

**Corollary 3.2.1.** Consider \( \text{EXTRACT}_1 \), where \( \forall v \in \mathcal{V}, |V_{[v]}| \leq n \). Then \( |\mathcal{M}| \geq \left( \frac{|\mathcal{V}|}{n} \right)^{|\mathcal{V}|} \) a.s.

**Corollary 3.2.2.** Consider \( \text{EXTRACT}_2 \), where \( \forall u, v \in \mathcal{V}, V_{[u]} \neq V_{[v]} \). Then \( |\mathcal{M}| = |\mathcal{V}| \) a.s.

Theorem 3.2 proves SHADOW-GCN does not oversmooth: 1. A normal GCN pushes the aggregation of same-degree nodes to the same point, while SHADOW-GCN with \( \text{EXTRACT}_2 \) ensures any two nodes (even with the same degree) have different aggregation. 2. A normal GCN wipes out all information in \( X \) after many times of aggregation, while SHADOW-GCN always preserves feature information. Particularly, with \( \phi_G(v) = \left( \delta_{[v]}(v) \right)^{-1/2} \), a normal GCN generates only one unique value of \( \overline{m} \) for all \( v \). By contrast, SHADOW-GNN generates \( |\mathcal{V}| \) different values for any \( \phi_G \) function.
3.2 Expressivity Analysis on SHADOW-SAGE: Function Approximation Perspective

We compare the expressivity by showing 1. SHADOW-SAGE can express all functions GraphSAGE can, and 2. SHADOW-SAGE can express some function GraphSAGE cannot. Recall, a GraphSAGE layer performs the following: \( h_v^{(t)} = \sigma \left( \left( W_1^{(t)} \right)^T h_v^{(t-1)} + \left( W_2^{(t)} \right)^T \left( \frac{1}{|N_v|} \sum_{u \in N_v} h_u^{(t-1)} \right) \right) \).

We can prove Point 1 by making an \( L' \)-layer SHADOW-SAGE identical to an \( L \)-layer GraphSAGE with the following steps: 1. let EXTRACT return the full \( L \)-hop neighborhood, and 2. set \( W_1^{(t)} = I \), \( W_2^{(t)} = 0 \) for \( L + 1 \leq t \leq L' \). For point 2, we consider a target function: \( \tau (X, G[v]) = C \cdot \sum_{u \in V[v]} \delta_v (u) \cdot x_u \) for some neighborhood \( G[v] \), scaling constant \( C \) and \( \delta_v (u) \) as defined in Proposition 3.1. An expressive model should be able to learn well this simple linear function \( \tau \).

GraphSAGE cannot learn \( \tau \) accurately, while SHADOW-SAGE can. We first show the GraphSAGE case. Let the depth of \( G[v] \) be \( L \). Firstly, we need GraphSAGE to perform message passing for exactly \( L \) times (where such a model can be implemented, e.g., \( L \) layers or \( L' \) layers with \( W_2 = 0 \) for \( L' - L \) layers). Otherwise, the extra \( L' - L \) message passings will propagate influence from nodes \( v' \notin V[v] \), violating the condition that \( \tau \) is independent of \( v' \). Next, suppose GraphSAGE can learn a function \( \zeta \) such that on some \( G'[v] \), we have \( \zeta (G'[v]) = \tau (G'[v]) \). We construct another \( G''[v] \) by adding an extra edge \( e \) connecting two depth-\( L \) nodes in \( G''[v] \). Edge \( e \) changes the degree distribution \( \delta_v (\cdot) \), and thus \( \zeta (G''[v]) \neq \zeta (G''[v]) \). On the other hand, there is no way for GraphSAGE to propagate the influence of edge \( e \) to the target \( v \), unless the model performs at least \( L + 1 \) message passings. So \( \zeta (G''[v]) \neq \zeta (G''[v]) \) regardless of the activation function and weight parameters. Therefore, \( \zeta \neq \tau \).

For SHADOW-SAGE, let EXTRACT return \( G[v] \). Then the model can output \( \zeta' = \left[ \hat{A}^{L'}_v X \right]_{v,v} \), after we 1. set \( W_1^{(t)} = 0 \) and \( W_2^{(t)} = I \) for all layers, and 2. either remove the non-linear activation or bypass \( \text{ReLU} \) by shifting \( X \) with bias. With known results in Markov chain convergence theorem [27], we derive the following theorem by analyzing the convergence of \( \hat{A}^{L'}_v \) when \( L' \to \infty \).

**Theorem 3.3.** SHADOW-SAGE can approximate \( \tau \) with error decaying exponentially with depth.

We have the following conclusions from above: 1. SHADOW-SAGE is more expressive than GraphSAGE. 2. appropriate EXTRACT function improves SHADOW-GNN expressivity. 3. There exists cases where it may be desirable to set the SHADOW-GNN depth much larger than the subgraph depth.

3.3 Expressivity Analysis on SHADOW-GNN: Topological Learning Perspective

While GCN and GraphSAGE are popular architectures in practice, they are not the theoretically most discriminative ones. The work in [49] establishes the relation in discriminativeness between GNNs and 1-dimensional Weisfeiler-Lehman test (i.e., 1-WL). And GIN [49] is an example architecture achieving the same discriminativeness as 1-WL. We show that applying the decoupling principle can further improve the discriminativeness of such GNNs, making them more powerful than 1-WL.

1-WL is a graph isomorphism test aiming at distinguishing graphs of different structures. A GNN as expressive as 1-WL thus well captures the topological property of the target node. While 1-WL is already very powerful, it may still fail in some cases. e.g., it cannot distinguish certain non-isomorphic, regular graphs. To understand why SHADOW-GNN works, we first need to understand why 1-WL fails. In a regular graph, all nodes have the same degree, and thus the “regular” property describes a global topological symmetry among nodes. Unfortunately, 1-WL (and the corresponding normal GNN) also operates globally on \( G \). Intuitively, on two different regular graphs, there is no way for 1-WL (and the normal GNN) to assign different labels by breaking such symmetry.

On the other hand, SHADOW-GNN can break such symmetry by applying decoupling. In Section 1, we have discussed how SHADOW-GNN is built from the local perspective on the full graph. The
key property benefiting SHADOW-GNN is that a subgraph of a regular graph may not be regular. Thus, SHADOW-GNN can distinguish nodes in a regular graph with the non-regular subgraphs as the scope. We illustrate the intuition with the example in Figure [1]. The graph $G$ is 3-regular and we assume all nodes have identical features. Our goal is to discriminate nodes $u$ and $v$ since their neighborhood structures are different. No matter how many iterations 1-WL runs, or how many layers the normal GNN has, they cannot distinguish $u$ and $v$. On the other hand, a SHADOW-GNN with 1-hop EXTRACT and at least 2 layers can discriminate $u$ and $v$.

**Theorem 3.4.** Consider GNNs whose layer function is defined by

$$h_v^{(ℓ)} = f_1^{(ℓ)} \left( h_v^{(ℓ-1)}, \sum_{u \in N_v} f_2^{(ℓ)} \left( h_u^{(ℓ-1)}, h_v^{(ℓ-1)} \right) \right),$$

where $f_1^{(ℓ)}$ and $f_2^{(ℓ)}$ are the update and message functions of layer-$ℓ$, implemented as MLPs. Then, such SHADOW-GNN is more discriminative than the 1-dimensional Weisfeiler-Lehman test.

The theorem also implies that SHADOW-GIN is more discriminative than a normal GIN due to the correspondence between GIN and 1-WL. See Appendix A for the proof of all theorems in Section 3.

### 3.4 Subgraph Extraction Algorithms

Our decoupling principle does not rely on specific subgraph extraction algorithms. Appropriate EXTRACT can be customized given the characteristic of $G$, and different EXTRACT leads to different implementation of our decoupling principle. In general, we summarize three approaches to design EXTRACT: 1. **heuristic based**, where we pick graph metrics that reflect neighbor importance and then design EXTRACT by such metrics; 2. **model based**, where we assume a generation process on $G$ and set EXTRACT as the reverse process, and 3. **learning based**, where we integrate the design of EXTRACT as part of the GNN training. In the following, we present several examples on heuristic based EXTRACT, which we also empirically evaluate in Section 5. We leave detailed evaluation on the model based and learning based EXTRACT as future work. See also Appendix C for details.

**Example heuristic based EXTRACT.** The algorithm is derived from the selected graph metrics. For example, with the metric being shortest path distance, we design a $L$-hop extractor. i.e., EXTRACT returns the full set or randomly selected subset of the target node’s $L$-hop neighbors in $G$. Picking the random walk landing probability as the metric, we can design a PPR-based extractor. i.e., we first run the Personalized PageRank (PPR) algorithm on $G$ to derive the PPR score of other nodes relative to the target node. Then EXTRACT define $V_{[ℓ]}$ by picking the top-$K$ nodes with the highest PPR scores. The subgraph $G_{[v]}$ is the node-induced subgraph $[v]$ of $G$ from $V_{[v]}$. One can easily extend this approach by using other metrics such as Katz index [20], SimRank [19] and feature similarity.

### 3.5 Architecture

**Subgraph pooling.** For a normal GNN performing node classification, the multi-layer message passing follows a “tree structure”. The nodes at level $L$ of the tree correspond to the $L$-hop neighborhood. And the tree root outputs the final embedding of the target node. Thus, there is no way to apply subgraph pooling or READOUT on the final layer output, since the “pool” only contains a single vector. For a SHADOW-GNN, since we decouple the $L$th layer from the $L$-hop neighborhood, it is natural to let each layer (including the final layer) output embeddings for all subgraph nodes. This leads to the design to READOUT all the subgraph node embeddings as the target node embedding.

We can understand the pooling for SHADOW-GNN from another perspective. In a normal GNN, the target node at the final layer receives messages from all neighbors, but two neighbor nodes may not have a chance to exchange any message to each other (e.g., two nodes $L$-hop away from the target may be 2$L$-hop away from each other). In our design, a SHADOW-GNN can pass messages between any pair of neighbors when the model depth is large enough. Therefore, all the subgraph node embeddings at the final layer capture meaningful information of the neighborhood.

In summary, the power of the decoupling principle lies in that it establishes the connection between the node- / link-level task and the graph-level task, e.g., to classify a node is seen as to classify

---

1 Unlike other PPR-based models [23, 6] which rewrite the graph by treating top PPR nodes as direct neighbors, our PPR neighborhood preserves the original multi-hop topology by returning node-induced subgraph.
the subgraph surrounding the node. From the neural architecture perspective, we can apply any subgraph pooling / READOUT operation originally designed for graph classification (e.g., AS-GCN [18], DeepGCN [28], JK-net [50], MixHop [1], Snowball [33], DAGNN [31] and GCNI [34] all include some variants of residue connection, either

Shallow-GNN, we can implement a trivial READOUT as “discarding all neighbor embeddings”, corresponding to performing center pooling. See Appendix D for algorithm and experiments.

Subgraph ensemble. It may be challenging in practice to design a single EXTRACT capturing all meaningful characteristics of the neighborhood. We can use multiple EXTRACT to jointly define the receptive field, and then ensemble multiple SHADOW-GNN at the subgraph level. Consider $R$ candidates \{EXTRACT$_i$\}, each returning $G'_i[v]$. To generate $v$’s embedding, we first use $R$ branches of $L'$-layer GNN to obtain intermediate embeddings for each $G'_i$, and then aggregate the $R$ embeddings by some learnable function $g$. In practice, we design $g$ as an attention based aggregation function (see Appendix D.2). Subgraph ensemble is useful both when \{EXTRACT$_i$\} consists of different algorithms and when each EXTRACT$_i$ performs the same algorithm under different parameters.

CASE STUDY Consider PPR-based EXTRACT$_i$ with different threshold $\theta_i$ on the neighbor PPR score. A SHADOW-GNN-ensemble can approximate PPRGo [6]. PPRGo generates embedding as: $\xi_v = \sum_u \pi_u h_v$, where $\pi_u$ is $u$’s PPR score and $h_v = MLP(x_v)$. We can partition $V$ into $1 \leq i \leq R$ s.t. nodes in $V_i$ have similar PPR scores denoted by $\pi_i$ and $\pi_i \leq \pi_{i+1}$. So $\xi_v = \sum_{i=1}^{R} \rho_i \left( \sum_{u \in V_i} h_v \right)$, where $\rho_i = \pi_i - \sum_{j<i} \pi_j$ and $V_i = \bigcup_{k=i}^{R} V_k$. Now for each branch of SHADOW-GNN-ensemble, let parameter $\theta_i = \pi_i$, so that EXTRACT$_i$ returns $V_i$. The GNN on $V_i$ can then learn $\sum_{u \in V_i} h_u$ (e.g., by a simple “mean” READOUT). Finally, set the ensemble weight as $\rho_i$. SHADOW-GNN-ensemble learns $\xi_v$. As EXTRACT also preserves graph topology, our model can be more expressive than PPRGo.

3.6 Practical Design: SHADOW-GNN

We now discuss the practical implementation of decoupled GNN – SHADOW-GNN. As the name suggests, in SHADOW-GNN, the scope is a shallow subgraph (i.e., with depth often set to 2 or 3).

In many realistic scenarios (e.g., citation networks, social networks, product recommendation graphs), a shallow neighborhood is both necessary and sufficient for the GNN to learn well. On “sufficiency”, we consider the social network example: the friend of a friend of a friend may share little commonality with you, and close friends may be at most 2 hops away. Formally, by the $\gamma$-decaying theorem [56], a shallow neighborhood is sufficient to accurately estimate various graph metrics. On “necessity”, since the neighborhood size may grow exponentially with hops, a deep neighborhood would be dominated by nodes irrelevant to the target. The corresponding GNN would first need to differentiate the many useless nodes from the very few useful ones, before it can extract meaningful features from the useful nodes. Finally, a shallow subgraph ensures scalability by avoiding “neighborhood explosion”.

Remark on decoupling. So far we have defined a decoupled model as having the model depth $L'$ larger than the subgraph depth $L$. Strictly speaking, a decoupled model also admits $L' = L$. For example, suppose in the full $L$-hop neighborhood, there are 70% nodes $L$ hops away. Applying decoupling, the EXTRACT excludes most of the $L$-hop neighbors, and the resulting subgraph $G'_v$ contains only 20% nodes $L$ hops away. Then it is reasonable to consider an $L$-layer model on such a depth-$L$ subgraph as also a decouple model. Compared with an $L$-layer model on the full $L$-hop neighborhood, an $L$-layer model on such a depth-$L$ $G'_v$ propagates much less information from nodes $L$ hops away. So the $L$ message passings are indeed decoupled from the full $L$-hop neighborhood.

Remark on neighborhood. The “sufficiency” and “necessity” in shallow neighborhood are not universal. In many other applications, long-range dependencies can be critical, as studied in [2]. In such cases, our practical implementation of SHADOW-GNN would incur accuracy loss. However, our decoupling principle in general may still be beneficial – “shallow subgraph” is a practical guideline rather than a theoretical requirement. We leave the work on such applications as future work.

4 Related Work

Deep GNNs. To improve the GNN performance while increasing the model depth, various layer architectures have been proposed. AS-GCN [18], DeepGCN [28], JK-net [50], MixHop [1], Snowball [33], DAGNN [31] and GCNI [34] all include some variants of residue connection, either
across multiple layers or within a single layer. In principle, such architectures can also benefit the feature propagation of a deep SHADOW-GNN, since their design does not rely on a specific neighborhood (e.g., $L$-hop). In addition to architectures, DropEdge [39] and Bayesian-GDC [14] propose regularization techniques by adapting dropout [42] to graphs. Such techniques are only applied during training, and inference may still suffer from issues such as oversmoothing.

**Sampling based methods.** Neighbor or subgraph sampling techniques have been proposed to improve training efficiency. FastGCN [8], VR-GCN [7], AS-GCN [13], LADIES [61] and MVS-GNN [10] sample neighbor nodes per GNN layer. Cluster-GCN [9] and GraphSAINT [55] sample a subgraph as the training minibatch. While sampling also changes the receptive field, all the above methods are fundamentally different from ours. The training samplers aim at estimating the quantities related to the full graph (e.g., the aggregation of the full $L$-hop neighborhood), and so the inference model still operates on the full neighborhood to avoid accuracy loss. For SHADOW-GNN, since the decoupling principle is derived from a local view on $G$, our EXTRACT does not estimate any full neighborhood quantities. Consequently, the sampling based methods only improve the training efficiency, while SHADOW-GNN addresses the computation challenge for both training and inference.

**Re-defining the neighborhood.** Various works reconstruct the original graph and apply the GNN on the re-defined neighborhood. GDC [23] views the reconstructed adjacency matrix as the diffusion matrix. SIGN [11] applies reconstruction for customized graph operators. AM-GCN [46] utilizes the reconstruction to separate feature and structure information. The above work re-define the neighborhood. However, they still have tightly coupled depth and scope. SHADOW-GNN can also work with the reconstructed graph $G'$ by simply applying EXTRACT on $G'$.

5 Experiments

**Setup.** We evaluate SHADOW-GNN on seven graphs. Six of them are for the node classification task: Flickr [55], Reddit [12], Yelp [55], ogbn-arxiv, ogbn-products and ogbn-papers100M [16]. The remaining is for the link prediction task: ogbl-collab [16]. The sizes of the seven graphs range from 9K nodes (Flickr) to 110M nodes (ogbn-papers100M). Flickr, Reddit and Yelp are under the inductive setting. ogbn-arxiv, ogbn-products and ogbn-papers100M are transductive. Consistent with the original papers, for the graphs on node classification, we measure “F1-micro” score for Yelp and “accuracy” for the remaining five graphs. For the link prediction task, we use “Hits@50” as the metric. See Appendix E.1 for details. We use W&B [5] for experiment tracking.

We construct SHADOW with six backbone models: GCN [22], GraphSAGE [12], GAT [44], JK-Net [50], GIN [49], SGC [47]. The first five are representatives of the state-of-the-art GNN architectures, jointly covering various message aggregation functions as well as the skip connection design. SGC simplifies normal GCN by moving all the neighbor propagation to the pre-processing step. Therefore, SGC is suitable for evaluating oversmoothing. The non-SHADOW models are trained with both full-batch and GraphSAINT-minibatch [55]. Due to the massive size of the full $L$-hop neighborhood, we need to perform sampling when training normal GNNs in order to make the computation time tractable. GraphSaint is suitable for our purpose since 1. it is the state-of-the-art minibatch method which achieves high accuracy, and 2. it supports various GNN architectures and scales well to large graphs. On the other hand, for SHADOW-GNN, both training and inference are always executed in minibatches. One advantage of SHADOW-GNN is that the decoupling enables straightforward minibatch construction: each target just independently extracts the small subgraph on its own.

We implement two EXTRACT described in Section 3.4 1. “PPR”, where we set the node budget $K$ as {200, 400} for the largest ogbn-papers100M and $K \leq 200$ for all other graphs; 2. “$L$-hop”, where we set the depth as {1, 2}. We implement various subgraph pooling functions: “mean” and “max” evaluated in this section and others evaluated in Appendix E.3. For the model depth, since $L' = 3$ is the standard setting in the literature (e.g., see the benchmarking in OGB [16]), we start from $L' = 3$ and further evaluate a deeper model of $L' = 5$. All accuracy are measured by 5 runs without fixing random seeds. Hyperparameter tuning and architecture configurations are in Appendix E.4.

**SHADOW-GNN neighborhood.** For both normal and SHADOW GNNs, Figure 2 shows on average how many neighbors are $L$ hops away from the target. For a normal GNN, the size of the neighborhood grows rapidly with respect to $L$, and the nodes 4 hops away dominate the neighborhood. For SHADOW-GNN using the Table E.1 EXTRACT, most neighbors concentrate within 2 hops. A small number of nodes are 3 hops away. Almost no nodes are 4 or more hops away. Importantly, not
only does the composition of the two kinds of neighborhood differ significantly, but also the size of SHADOW-GNN scope is much smaller (see also Table[1]). Such small subgraphs are essential to high computation efficiency. Finally, we can ignore the very few distant neighbors ($L \geq 4$), and regard the (effective) depth of SHADOW-GNN subgraph as $L = 2$ (or at most 3). Under such practical value of $L$, a model with $L' \geq 3$ is indeed a SHADOW-GNN (see “Remark on decoupling” in Section 3.6).

Table 1: Comparison on test accuracy / F1-micro score and inference cost (tuned with DropEdge)

<table>
<thead>
<tr>
<th>Method</th>
<th>Flickr</th>
<th>Reddit</th>
<th>Yelp</th>
<th>ogbn-axiv</th>
<th>ogbn-products</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Accuracy</td>
<td>Cost</td>
<td>Accuracy</td>
<td>Cost</td>
<td>Accuracy</td>
</tr>
<tr>
<td>GCN</td>
<td>0.5159</td>
<td>0.0001</td>
<td>2E0</td>
<td>0.9532</td>
<td>0.0003</td>
</tr>
<tr>
<td>GCN-SAINT</td>
<td>0.5155</td>
<td>0.0007</td>
<td>2E0</td>
<td>0.9523</td>
<td>0.0003</td>
</tr>
<tr>
<td>sHADOW-GCN</td>
<td>0.5234</td>
<td>0.0009</td>
<td>1E0</td>
<td>0.9576</td>
<td>0.0005</td>
</tr>
<tr>
<td>(PPR)</td>
<td>0.5268</td>
<td>0.0008</td>
<td>1E0</td>
<td>0.9564</td>
<td>0.0004</td>
</tr>
<tr>
<td>+Pooling</td>
<td>0.5896</td>
<td>0.0013</td>
<td>1E0</td>
<td>0.9624</td>
<td>0.0002</td>
</tr>
<tr>
<td>GraphSAGE</td>
<td>0.5140</td>
<td>0.0004</td>
<td>3E0</td>
<td>0.9553</td>
<td>0.0002</td>
</tr>
<tr>
<td>SAGE-SAINT</td>
<td>0.5154</td>
<td>0.0002</td>
<td>2E0</td>
<td>0.9626</td>
<td>0.0004</td>
</tr>
<tr>
<td>sHADOW-SAGE-(2-hop)</td>
<td>0.5176</td>
<td>0.0002</td>
<td>3E0</td>
<td>0.9671</td>
<td>0.0003</td>
</tr>
<tr>
<td>(PPR)</td>
<td>0.5201</td>
<td>0.0002</td>
<td>2E0</td>
<td>0.9670</td>
<td>0.0009</td>
</tr>
<tr>
<td>+Pooling</td>
<td>0.5312</td>
<td>0.0019</td>
<td>1E0</td>
<td>0.9672</td>
<td>0.0003</td>
</tr>
<tr>
<td>GAT</td>
<td>0.5335</td>
<td>0.0015</td>
<td>2E0</td>
<td>0.9675</td>
<td>0.0005</td>
</tr>
<tr>
<td>GAT-SAINT</td>
<td>0.5356</td>
<td>0.0013</td>
<td>1E0</td>
<td>0.9688</td>
<td>0.0002</td>
</tr>
<tr>
<td>(PPR)</td>
<td>0.5417</td>
<td>0.0006</td>
<td>2E0</td>
<td>0.9692</td>
<td>0.0007</td>
</tr>
<tr>
<td>+Pooling</td>
<td>0.5395</td>
<td>0.0013</td>
<td>1E0</td>
<td>0.9703</td>
<td>0.0003</td>
</tr>
<tr>
<td>sHADOW-GAT</td>
<td>0.5070</td>
<td>0.0002</td>
<td>2E1</td>
<td>0.9970</td>
<td>0.0004</td>
</tr>
<tr>
<td>(PPR)</td>
<td>0.5164</td>
<td>0.0003</td>
<td>2E0</td>
<td>0.9971</td>
<td>0.0003</td>
</tr>
<tr>
<td>+Pooling</td>
<td>0.5153</td>
<td>0.0004</td>
<td>2E0</td>
<td>0.9971</td>
<td>0.0004</td>
</tr>
<tr>
<td>sHADOW-GAT</td>
<td>0.5349</td>
<td>0.0023</td>
<td>1E1</td>
<td>0.9707</td>
<td>0.0004</td>
</tr>
<tr>
<td>(PPR)</td>
<td>0.5352</td>
<td>0.0028</td>
<td>2E0</td>
<td>0.9713</td>
<td>0.0004</td>
</tr>
<tr>
<td>+Pooling</td>
<td>0.5364</td>
<td>0.0026</td>
<td>1E0</td>
<td>0.9700</td>
<td>0.0004</td>
</tr>
</tbody>
</table>

Comparison with baselines. Table[1] shows the performance comparison of SHADOW-GNN with the normal GNNs. All models on all datasets have uniform hidden dimension of 256. We define the metric “inference cost” as the average amount of computation to generate prediction for one test node. Inference cost is a measure of computation complexity (see Appendix B for the calculation) and is independent of hardware / implementation factors such as parallelization strategy, batch processing, etc. For cost of SHADOW-GNN, we do not include the overhead on computing EXTRACT, since it is hard to calculate such cost analytically. Empirically, subgraph extraction is much cheaper than model computation (see Figure[8][9] for time evaluation on CPU and GPU). During training, we apply DropEdge [39] to both the baseline and SHADOW models. DropEdge helps improve the baseline accuracy by alleviating over-smoothing, and benefits SHADOW-GNN due to its regularization effects. See Appendix C for results on other architectures including GIN and JK-Net.

ACCURACY. We aim at answering the following questions: 1. Can we improve accuracy by decoupling a baseline model? 2. What architecture components can we tune to improve accuracy of a decoupled model? 3. What EXTRACT can we tune to improve the accuracy of a decoupled model? To answer Q1, we fix the backbone architecture and remove pooling. Then we inspect “3-layer normal GNN vs. 3-layer SHADOW-GNN” and “5-layer normal GNN vs. 5-layer SHADOW-GNN”. Clearly, SHADOW-GNNs (with scope size no more than 200) in general achieve significantly higher accuracy than the normal GNNs. This indicates that a shallow neighborhood contains sufficient information,
and customizing the scope can benefit accuracy even without architecture changes (from Figure 2, a depth-3 \(g[v] \) differs significantly from the 3-hop neighborhood). To answer Q1, we focus on the PPR EXTRACT and thus compare the rows in blue background. We use the 3-layer \( \text{SHAOW-GNN} \) without pooling as the baseline and analyze the effects of 1. increasing the GNN depth without expanding scope, and 2. adding subgraph pooling. Comparing among the rows in light blue background, we observe that in many cases, simply increasing the depth from 3 to 5 leads to significant accuracy gain. Comparing the light blue rows with the dark blue rows, we observe that sometimes pooling can further improve the accuracy of a \( \text{SHAOW-GNN} \). In conclusion, both types of architecture tuning are effective ways of optimizing a \( \text{SHAOW-GNN} \). Finally, to answer Q3, we compare the light blue rows with the light yellow rows. In general, PPR EXTRACT leads higher accuracy than 2-hop EXTRACT, demonstrating the importance of designing a good EXTRACT.

**Inference cost**  
Inference cost of \( \text{SHAOW-GNN} \) is orders of magnitude lower than the normal GNNs (a 5-layer \( \text{SHAOW-GNN} \) is still much cheaper than a 3-layer normal GNN). The high cost of the baselines is due to the “neighborhood explosion” with respect to more layers. \( \text{SHAOW-GNN} \) is efficient and scalable as the cost only grows linearly with the model depth. Note that GraphSAINT only improves efficiency during training since its inference operates on the full \( L \)-hop neighborhood.

**Scaling to 100 million nodes.** We further scale \( \text{SHAOW-GNN} \) to ogbn-papers100M, one of the largest public dataset. Even through the full graph size is at least two orders of magnitude larger than the graphs in Table 1, the localized scope of \( \text{SHAOW-GNN} \) barely needs to increase. Since \( \text{SHAOW-GNN} \) performs minibatch computation, a low-end GPU with limited memory capacity can compute \( \text{SHAOW-GNN} \) on ogbn-papers100M efficiently. We show in Appendix E.1 that we can train and inference our model with as little as 4GB GPU memory consumption. This is infeasible using normal GNNs. Table 2 summarizes our comparison with the top leaderboard methods [45][11][47]. We only include those methods that do not use node labels as the model input (i.e., the most standard setup). We achieve at least 3 orders of magnitude reduction in neighborhood size without sacrificing accuracy. For SIGN-XL and SGC, their neighborhood is too large to count the exact size. Also, their preprocessing consumes 5× more CPU memory than \( \text{SHAOW-GNN} \) (Appendix E.1).

**Extending to link-level task.** We further show that \( \text{SHAOW-GNN} \) is general and can be extended to the link prediction task. There are two settings of ogbn-collab. We follow the one where validation edges cannot be used in training updates. This is the setting which most leaderboard methods follow. Table 3 extends Table 2 by showing the comparison with the top GNN models under the same setting. \( \text{SHAOW-SAGE} \) outperforms the rank-1 model with significant margin.

**Oversmoothing.** To validate Theorem 3.2, we pick SGC as the backbone architecture. SGC with power \( L \) is equivalent to \( L \)-layer GCN without activation. Performance comparison between SGC and \( \text{SHAOW-SGC} \) thus reveals the effect of oversmoothing without introducing other factors due to optimizing deep neural networks (e.g., vanishing gradients). In Figure 3, we vary the power of SGC and \( \text{SHAOW-SGC} \) from 1 to 40 (see Appendix E.5 for details). While SGC gradually collapses local information into global “white noise”, accuracy of \( \text{SHAOW-SGC} \) does not degrade. This validates our theory that extracting local subgraphs prevents oversmoothing.

### Conclusion

We have presented a design principle to decouple the depth and scope of GNNs. Applying such a principle on various GNN architectures simultaneously improves expressivity and computation scalability of the corresponding models. We have presented thorough theoretical analysis on expressivity from three different perspectives, and also rich design components (e.g., subgraph extraction functions, architecture extensions) to implement such design principle. Experiments show significant performance improvement over a wide range of graphs, GNN architectures and learning tasks.
References


[34] Zhewei Wei Ming Chen, Bolin Ding Zengfeng Huang, and Yaliang Li. Simple and deep graph convolutional networks. 2020.


Checklist

1. For all authors...
   (a) Do the main claims made in the abstract and introduction accurately reflect the paper’s contributions and scope? [Yes]
   (b) Did you describe the limitations of your work? [Yes]
   (c) Did you discuss any potential negative societal impacts of your work? [N/A]
   (d) Have you read the ethics review guidelines and ensured that your paper conforms to them? [Yes]

2. If you are including theoretical results...
   (a) Did you state the full set of assumptions of all theoretical results? [Yes]
   (b) Did you include complete proofs of all theoretical results? [Yes] See appendix A.

3. If you ran experiments...
   (a) Did you include the code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL)? [Yes] See the released code.
   (b) Did you specify all the training details (e.g., data splits, hyperparameters, how they were chosen)? [Yes] See the released code. It contains all configurations.
   (c) Did you report error bars (e.g., with respect to the random seed after running experiments multiple times)? [Yes]
   (d) Did you include the total amount of compute and the type of resources used (e.g., type of GPUs, internal cluster, or cloud provider)? [No]

4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets...
(a) If your work uses existing assets, did you cite the creators? [Yes] See citation after each dataset.
(b) Did you mention the license of the assets? [N/A]
(c) Did you include any new assets either in the supplemental material or as a URL? [No]
(d) Did you discuss whether and how consent was obtained from people whose data you're using/curating? [No]
(e) Did you discuss whether the data you are using/curating contains personally identifiable information or offensive content? [No]

5. If you used crowdsourcing or conducted research with human subjects...
   (a) Did you include the full text of instructions given to participants and screenshots, if applicable? [N/A]
   (b) Did you describe any potential participant risks, with links to Institutional Review Board (IRB) approvals, if applicable? [N/A]
   (c) Did you include the estimated hourly wage paid to participants and the total amount spent on participant compensation? [N/A]