MEMORY-AUGMENTED DESIGN OF GRAPH NEURAL NETWORKS

Anonymous authors
Paper under double-blind review

ABSTRACT

The expressive power of graph neural networks (GNN) has drawn much interest recently. Most existent work focused on measuring the expressiveness of GNN through the task of distinguishing between graphs. In this paper, we inspect the representation limits of locally unordered messaging passing (LUMP) GNN architecture through the lens of *node classification*. For GNNs based on permutation invariant local aggregators, we characterize graph-theoretic conditions under which such GNNs fail to discriminate simple instances, regardless of underlying architecture or network depth. To overcome this limitation, we propose a novel framework to augment GNNs with global graph information called *memory augmentation*. Specifically, we allow every node in the original graph to interact with a group of memory nodes. For each node, information from all the other nodes in the graph can be gleaned through the relay of the memory nodes. For proper backbone architectures like GAT and GCN, memory augmented GNNs are theoretically shown to be more expressive than LUMP GNNs. Empirical evaluations demonstrate the significant improvement of memory augmentation. In particular, memory augmented GAT and GCN are shown to either outperform or closely match state-of-the-art performance across various benchmark datasets.

1 INTRODUCTION

Graph neural networks (GNN) are a powerful tool for learning with graph-structured data, and has achieved great success on problems like node classification (Kipf & Welling, 2016), graph classification (Duvenaud et al., 2015) and link prediction (Grover & Leskovec, 2016). GNNs typically follow a recursive neighborhood aggregation (or message passing) scheme (Xu et al., 2019) such that within each aggregation step, each node collects information from its neighborhood (usually feature vectors), then apply aggregation and combination mechanism to compute its new feature vector. Typically, GNN architectures differ in their design of aggregation and combination mechanisms. Popular architectures like GCN (Kipf & Welling, 2016), GraphSAGE (Hamilton et al., 2017), and GAT (Veličković et al., 2018) fall into this paradigm.

Despite their empirical success, there are a couple of limitations of GNNs that update node features only based on local information. One important issue is their limited expressive power. In the graph classification setting (Xu et al., 2019), it was shown that message passing neural networks are at most as powerful as Weisfeiler Lehman graph isomorphism tests. A more recent line of work has suggested using variants of message passing scheme that incorporates the layout of local neighborhoods (Sato et al., 2019; Klicpera et al., 2020) or spatial information of the graph (You et al., 2019).

Another problem is due to the phenomenon that the performance of GNN does not improve, or even degrades when layer size increases (Kipf & Welling, 2016; Xu et al., 2018; Li et al., 2018; Oono & Suzuki, 2020), known as the problem of *over-smoothing* that makes extending the receptive path of message passing GNNs a difficult task. Many successful GNN architectures are based on stacking a few number of layers like 2 or 3 (Kipf & Welling, 2016), which could be viewed as an implicit inductive bias that node labels are determined up to neighborhoods that are a few hops away. However this assumption may not hold for many real-world data—for example, structurally similar nodes may offer strong predictive power for very distant node pairs (Donnat et al., 2018). Several techniques are proposed for aggregating node information from a wider range (Xu et al., 2018; Klicpera et al., 2019a;b).

In this paper, we investigate the expressive power of GNNs through the task of node classification. We characterize cases where GNNs that builds on LUMP protocol fail, regardless of underlying implementation or aggregation range. We then propose a novel architecture that aggregates information beyond the local neighborhood. Through making use of global feature information, we can distinguish a wide range of cases that LUMP type GNNs inherently fail. Our main contributions are summarized as follows:

- We discuss the expressive power of GNNs for node classification tasks under the premise that node labels are not solely determined by first-order neighborhood information, and show an indistinguishable scenario where LUMP algorithms fail to discriminate nodes in structurally different graphs even if infinite rounds of message passing is performed.
- We develop a novel framework that extends GNN with *global graph information* called *memory augmentation*, motivated by memory networks (Graves et al., 2014; Weston et al., 2014). With proper choice of backbone architectures like GAT or GCN, the augmented architectures are provably more expressive than LUMP type GNNs in that they discriminate a wide range of cases that LUMP type GNNs fail with a compact architecture of two layers.
- We derive two representative memory augmented architectures, MemGAT and MemGCN, and evaluate their performance on standard datasets. Empirical results show that the memory augmented architectures significantly improves their corresponding backbone architectures across all tasks, either outperforming or closely matching state-of-the-art performance.

2 Representation limits of locally unordered message passing

In this paper we consider the task of *node classification* over an undirected graph $G = (V, E)$ with node set V and edge set E. Let $N = |V|$ be the number of nodes and A, D be its associated adjacency matrix and degree matrix. For each node $v \in V$, let $\mathcal{N}_v = \{u \mid (u, v) \in E\}$ be its neighborhood set and $X_v \in \mathcal{X} \subset \mathbb{R}^d$ be its associating feature. Each node $v \in V$ is associated with a label Y_v. Node classification algorithms make predictions of Y_v based on the information given by G and the node feature matrix X. In this paper we will be interested in situations where node labels are not determined solely by their first order neighborhood information. i.e., $\mathbb{P}(Y_v|G, X) \neq \mathbb{P}(Y_v|X_v, X_u, u \in \mathcal{N}_v), \forall v \in V$. For a collection of elements $\mathcal{C}$ that are not necessarily distinct, we use $\{\mathcal{C}\}$ to denote its set representation and $\{\!\{\mathcal{C}\}\!\}$ to denote its multiset representation. For each $c \in \{\mathcal{C}\}$, let $r_{\mathcal{C}}(c)$ be the multiplicity of c in $\{\!\{\mathcal{C}\}\!\}$. A popular tool for encoding higher order graph information is to utilize the *locally unordered message passing (LUMP)* protocol (Garg et al., 2020) to build GNNs. For node v, its (hidden) representation $h_v^{(l)}$ is updated using an *aggregation and combine* strategy:

$$h_v^{(l)} = \mathsf{COMBINE}\left(h_v^{(l-1)}, \mathsf{AGG}\left(\left\{\!\!\left\{h_u^{(l-1)}, u \in \mathcal{N}_v\right\}\!\!\right\}\right)\right) \tag{1}$$

The protocol is unordered in the sense that no spatial information (like the relative orientation of neighbors) is used throughout the message passing procedure, and the aggregator AGG is often chosen as a permutation invariant function. After k rounds of message passing, each node will have a feature vector that encodes the information of its *height k rooted subtree*. Aggregation strategies that extend to arbitrary node were suggested in pioneering works of GNNs (Scarselli et al., 2009) that use a learnable, *contractive* aggregator, and perform infinite rounds of message passing till convergence.

Next we discuss the expressive power of the above mentioned mechanisms. Let $\mathsf{G}(v)$ be the subgraph of G that contains $v, \mathcal{N}_v$ and their associated edges. We consider two graphs, $G = (V, E)$ and $G' = (V', E')$, with corresponding feature matrices X and X'.

Definition 1. (Gross & Tucker, 2001) A graph map $f : G \mapsto G'$ is called a *local isomorphism* if for every node $v \in V$, the restriction of f to $\mathsf{G}(v)$ is an isomorphism onto $\mathsf{G}(f(v))$.

We say a graph G is *locally indistinguishable* to graph G', if there exists a surjective local isomorphism f from G to G', and if in addition the feature matrices are related as $X_v = X'_{f(v)}$. The following theorem states a specific situation where LUMP type GNNs fail to distinguish between nodes in different graph contexts.

Theorem 1. *If G is locally indistinguishable to graph G' under map f, then under the LUMP protocol equation 1, we have $h_v^{(l)} = h_{f(v)}^{(l)'}$ for any $v \in V$ and $l \in \mathbb{Z}_+$.*

The proof will be given in appendix A.2. We give a pictorial illustration in figure 1, with node color indicating feature value. The local isomorphism is constructed as $f(a_1) = f(a_4) = b_1, f(a_2) = f(a_5) = b_2, f(a_3) = f(a_6) = b_3$. Theorem 1 implies that representations of nodes with same color remain identical for arbitrary rounds of LUMP. Hence for the node label assignment $l_{a_1} = 1, l_{b_1} = 0$, GNNs based on LUMP fail to express this difference.

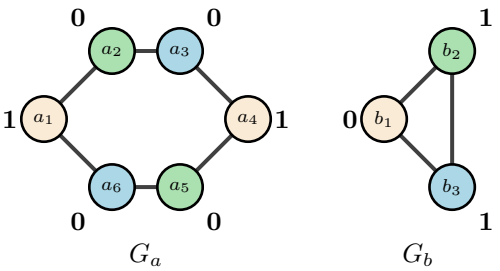

Figure 1: Graph structures that LUMP GNNs fail to discriminate in terms of node classification tasks. (with respect to the assignment annotated near the corresponding node)

3 Our model

3.1 Memory augmentation

Our model is constructed via augmenting a *backbone GNN* whose updating rule could be written as a simplified version of equation 1: the hidden representation in the lth layer satisfies $h_v^{(l+1)} = \mathsf{AGG}_{\vartheta^{(l)}}\left(\left\{\!\!\left\{h_v^{(l)}, h_{\mathcal{N}_v}^{(l)}\right\}\!\!\right\}\right)$. The aggregation function AGG is parameterized by $\vartheta^{(l)}$. We will assume the set $\mathcal{X}$ to be countable, then by (Xu et al., 2019, Lemma 4), the range of $h_v^{(l)}$, denoted $\mathcal{H}^{(l)}$, is also countable for any given parameterization and any layer l.

Augmentation Inspired by memory networks (Graves et al., 2014; Weston et al., 2014), the memory component is formulated as a collection of learnable objects corresponding to $M \ll N$ auxiliary nodes, called *memory nodes*. Let $V_{\mathcal{M}}$ be the set of memory nodes. We define memory as an $M \times d$ matrix $\mathbf{m}$ with row vector $\mathbf{m}_v$ serving as the *memory embedding* of memory node v. We allow every node in the original graph to interact with the memory component. In particular, we augment the original graph G into a slightly larger graph $G^* = (V^*, E^*)$, where $V^* = V \bigcup V_{\mathcal{M}}$ and

$$E_{ij}^* = \begin{cases} E_{ij}, & \text{if } i, j \in V \\ 1, & \text{if } i \in V, j \in V_{\mathcal{M}} \text{ or } i \in V_{\mathcal{M}}, j \in V \\ 0, & \text{if } i, j \in V_{\mathcal{M}} \end{cases} \tag{2}$$

so that each memory node is connected to every node in the original graph G. Performing message passing using the backbone architecture would thus allow every node in the original graph to aggregate information from both its neighbors and the memory component, while the memory component aggregates information from the whole graph:

$$\forall v \in V, h_v^{(l+1)} = \mathsf{AGG}_{\vartheta^{(l)}}\left(\left\{\!\!\left\{h_v^{(l)}, h_{\mathcal{N}_v}^{(l)}, \mathbf{m}_{\mathcal{V}_{\mathcal{M}}}\right\}\!\!\right\}\right)$$
$$\forall v \in V_{\mathcal{M}}, \mathbf{m}_v^{(l+1)} = \mathsf{AGG}_{\vartheta^{(l)}}\left(\left\{\!\!\left\{\mathbf{m}_v^{l}, h_V^{(l)}\right\}\!\!\right\}\right)$$

Figure 2 illustrates how memory augmentation does "symmetry breaking" in locally indistinguishable graphs: although the height k rooted subtree remains identical for G_a and G_b for any k, starting from the second layer, messages from the memory component would become distinct thus makes discrimination possible. More rigorously, we characterize conditions for the backbone architecture under which the memory augmented architecture achieves stronger expressive power:

Definition 2. We say two multisets $\{\!\!\{X\}\!\!\}$ and $\{\!\!\{X'\}\!\!\}$ are distributionally equivalent if they share the same underlying set, i.e. $\{X\} = \{X'\}$, and the corresponding multiplicities are proportional to each other: for $\forall x \in \{X\}, r_X(x)/r_{X'}(x) \equiv m, m \in \mathbb{Z}_+$.

Note that distributional equivalence is a well defined equivalence relation. We require the following condition on the backbone GNN:

[$\mathbf{C}_1$] The backbone GNN identifies bounded size multisets in $\mathcal{X}$ up to distributional equivalence.

Using a single memory node ω, we have the following result:

Lemma 1. *Under condition $\mathbf{C}_1$, there exists a vector $\tilde{\mathbf{m}}_\omega$ such that if the memory embedding is initialized with $\mathbf{m}_{\omega,G}^{(0)} = \tilde{\mathbf{m}}_\omega$ for any graph G, then the following holds:*

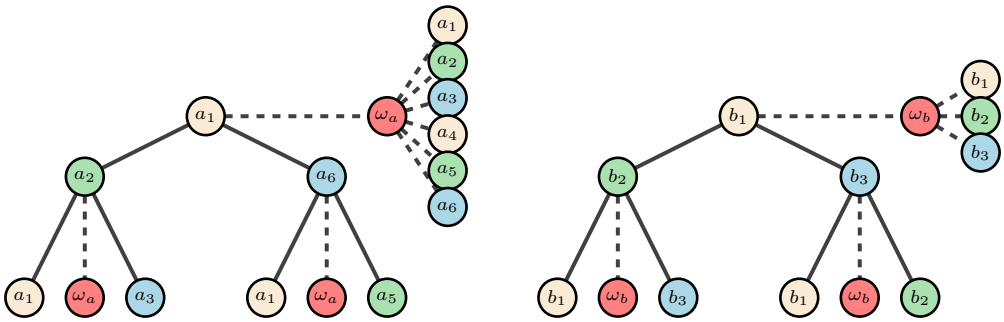

Figure 2: Illustration of memory augmentation applied to the two graphs in figure 1: G_a (left) and G_b (right). Both graphs are augmented with one memory node. Aggregation paths of a_1 and b_1 are presented as two hop subtree structures, corresponding to a two round message passing using the memory augmented attention mechanism. Memory connections are drawn as dashed lines and original connections as solid lines. The figure implies that, although the local subtree structures rooted at a_1 and b_1 are identical, the message carried by the memory node distinguishes a_1 from b_1, provided that the message passing is performed for more than one round.

(i) For any $l \in \mathbb{Z}_+$, there exists a parameter configuration $\vartheta_*^{(l)}$ such that the map $\mathsf{AGG}_{\vartheta_*^{(l)}}$ is injective over all multisets that takes the form: $\left\{\!\!\left\{ C, \mathbf{m}_{\omega,G}^{(l)} \right\}\!\!\right\}$, where $\{C\}$ is a bounded subset of $\mathcal{X}$

(ii) For graph pairs (G, G') with $\{\!\{X\}\!\} \neq \{\!\{X'\}\!\}$, the updated memory representations after 1 round of message passing are different, i.e. $\mathbf{m}_{\omega,G}^{(1)} \neq \mathbf{m}_{\omega,G'}^{(1)}$.

It then follows from (Xu et al., 2019, Theorem 3) that memory augmentation enhances proper backbone architectures to achieve the strongest expressive power among LUMP GNNs. Lemma 1 suggests that as long as the overall network architecture contains more than one layer, starting from the second layer, nodes in the original graph would receive messages that contain some kind of *aggregated global information* that *identifies* the multiset feature representation of the original graph. This could be also viewed as a *graph specific bias* term that discriminates that message passing protocol from LUMP. As a consequence, we have the following theorem for binary node labels:

Theorem 2. *For locally indistinguishable graphs (G, G') under map f, there exists a two layer memory augmented GNN with its backbone architecture satisfying condition $\mathbf{C}_1$, such that with a proper memory initialization (i.e. the one in lemma 1), the final output h satisfies: if $\{\!\{X\}\!\} \neq \{\!\{X'\}\!\}$, then $h_v > 0.5$ and $h_{f(v)} \leq 0.5$ for any $v \in V$.*

Theorem 2 suggests that memory augmentation helps to identify locally indistinguishable graphs with different multiset feature representations. The following corollary justifies using two most popular GNN architecture as the backbone architecture for memory augmentations. We adopt the formulation in Xu et al. (2019):

Corollary 1. Both GAT and GCN satisfies condition $\mathbf{C}_1$.

We defer all proofs to appendix A.2. For an empirical verification, we conducted a simple experiment corresponding to the setup in figure 1. We use categorical features (with a cardinality of three) and compared memory augmented GAT with GAT. We use the reference GAT architecture in (Veličković et al., 2018) and augment it with one memory node. We use a learnable multi-layer perceptron as the nonlinear readout function for both models. Figure 3 shows the result. It could be seen from the figure that the training accuracy of GAT never exceeds $2/3$, which is its theoretical limit in this example. While memory augmented GAT fits the data perfectly after sufficient rounds of gradient updates, thereby verifies our theoretical findings.

3.2 Two canonical designs: MemGAT and MemGCN

Now we derive two canonical architectures that enhances two popular GNNs GAT (Veličković et al., 2018) and GCN (Kipf & Welling, 2016). The resulting architectures are termed MemGAT and MemGCN. For both designs, we introduce several improvements to the vanilla memory augmentation described in section 3.1 to balance the contribution from the original graph and the memory component. We also make a discussion on architectures that may not benefits from memory augmentation, see appendix A.4 for details.

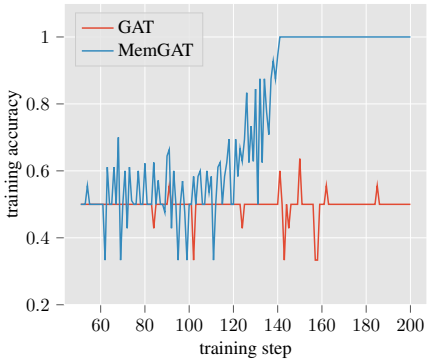

Figure 3: Comparison of memory augmented GAT with GAT on synthesized data, performance measured in training accuracy, results for initial 50 gradient iterations are discarded

MemGAT We enhance attention-based GNNs (Veličković et al., 2018) via using a modified attention mechanism. The attention weights are calculated as:

$$\alpha_{ij}^{(l)} = \begin{cases} \lambda\psi_{ij}\exp\left(\beta_{ij}^{(l)}\right)/Z_i^{(l)}, & \text{if } i,j \in V \\ \frac{1-\lambda}{M}\exp\left(\beta_{ij}^{(l)}\right)/Z_i^{(l)}, & \text{if } i \in V, j \in V_{\mathcal{M}} \\ \exp\left(\beta_{ij}^{(l)}\right)/Y_i^{(l)}, & \text{if } i \in V_{\mathcal{M}}, j \in V \\ \mathbf{1}_{(i=j)}\exp\left(\beta_{ij}^{(l)}\right)/Y_i^{(l)} & \text{if } i,j \in V_{\mathcal{M}} \end{cases} \tag{3}$$

where $h_i^{(l)} \in \mathbb{R}^{d_l}$ denotes the hidden feature of node $i \in V$ in the l th layer with $h_i^{(0)} = X_i$, and $\beta_{ij}^{(l)} = \phi^{(l)}(\sigma^{(l)}(h_i^{(l)}), \sigma^{(l)}(h_j^{(l)}))$ with $\sigma^{(l)}$ a (possibly) nonlinear function for the l th layer. $\{\psi_{ij}\}_{i,j,\in V}$ are edge weights over the original graph, and $\lambda \in (0,1)$ is a parameter that balances the contribution between information provided by the original neighbor and the memory component. $\{Z_i^{(l)}\}_{i\in V}, \{Y_j^{(l)}\}_{j\in V_{\mathcal{M}}}$ are normalizing factors such that $\sum_j \alpha_{ij}^{(l)} = 1, \forall l$. We write separately the updating equation for hidden representations corresponding to nodes in the original graph and memory embeddings:

$$\forall v \in V, \quad h_i^{(l+1)} = \sum_{k\in\{i\}\bigcup\mathcal{N}_i} \alpha_{ik}^{(l)}\sigma^{(l)}(h_k^{(l)}) + \sum_{v\in V_{\mathcal{M}}} \alpha_{iv}^{(l)}\sigma^{(l)}(\mathbf{m_v}^{(l)}) \tag{READ}$$

$$\forall v \in V_{\mathcal{M}}, \quad \mathbf{m_v}^{(l+1)} = \alpha_{vv}^{(l)}\sigma^{(l)}(\mathbf{m_v}^{(l)}) + \sum_{j\in V} \alpha_{vj}^{(l)}\sigma^{(l)}(h_j^{(l)}) \tag{WRITE}$$

Connections to memory based network design The above equations interprets the memory component as content addressing memory design in memory-based neural networks (Graves et al., 2014; Weston et al., 2014): For nodes in the original graph, the aggregation step is interpreted as a *reading* step with a selective focus over both its immediate neighbors and the memory obtained from the previous layer. Alternatively, for the memory nodes, aggregation operations are hence interpreted as a *writing* step.

MemGCN The augmented GCN adopts similar design aspects as MemGAT with balancing parameter λ. We adopt the GCN formulation in Dehmamy et al. (2019), by using edge weights $\{\psi_{ij}\}_{i,j,\in V}$ to characterize the aggregation scheme:

$$\forall v \in V, \quad h_i^{(l+1)} = \lambda\left(\psi_{ii}W^{(l)}\sigma^{(l)}(h_i^{(l)}) + \sum_{j\in\mathcal{N}_i}\psi_{ij}W^{(l)}\sigma^{(l)}(h_i^{(l)})\right) + \frac{1-\lambda}{M}\sum_{v\in V_{\mathcal{M}}} W\mathbf{m}_v^{(l)}$$

$$\forall v \in V_{\mathcal{M}}, \quad \mathbf{m_v}^{(l+1)} = \lambda W^{(l)}\mathbf{m_v}^{(l)} + \frac{1-\lambda}{M}\sum_{j\in V} W^{(l)}\sigma^{(l)}(h_i^{(l)})$$

where $W^{(l)}$s are learnable weight matrices. We identify two canonical types of GCN: Kipf & Welling (2016) uses $\psi_{ij} = 1/\sqrt{(d_i+1)(d_j+1)}$, and a slightly modified method of Hamilton et al. (2017) uses $\psi_{ij} = 1/(d_i+1)$.

Remark 1. *Recently several variants of LUMP that take ordered information into account are proposed, among which CPCGNN (Sato et al., 2019) and DimeNet (Klicpera et al., 2020) are based on modifications to the first-order message passing scheme, and are provably more expressive than LUMP type GNNs (Garg et al., 2020). We give a detailed comparison of memory augmented GNN architectures with CPCGNN and DimeNet in appendix A.5. We found that regarding node classification, there's no dominating choice between these methods. And it would be of interest to extend the idea of memory augmentation to other kinds of message passing schemes. We leave this direction to future research.*

4 Other Related Work

The expressive power of GNNs has been extensively studied recently (Xu et al., 2019; Maron et al., 2019; Garg et al., 2020; Sato et al., 2020; Zhang & Xie, 2020), with the majority of these works defining the expressive power of GNN with its ability to distinguish different graphs. (Xu et al., 2019) proposed the GIN model that is theoretically as powerful as the first order Weisfeiler Lehman (1-WL) test and is thus the most powerful among LUMP GNNs. (Zhang & Xie, 2020) suggests accounting for cardinality information for attention based GNNs. More powerful GNN architectures are then proposed via extending the message passing formulation: higher order GNNs or k-GNNs (Morris et al., 2019; Maron et al., 2019) operate on k- tuple of nodes and aggregate information from neighborhoods defined on k- tuples of nodes. While such kind of GNN variants could be shown to be as powerful as k-WL test, they are not directly applicable to node level tasks. Another line of work (Sato et al., 2019; Klicpera et al., 2020) generalizes the LUMP protocol to allow nodes passing messages depending on their relative position in their belonging neighborhoods. Such kind of variants are also provably more powerful than LUMP GNNs and are applicable to node level tasks. See section 6 for a more detailed comparison between our approach and these variants. Position aware GNN (PGNN) (You et al., 2019) is another class of powerful GNN variants that uses random anchor sets to break local structural symmetry, and integrates positional information into a modified message passing architecture, with an inductive bias suitable for relational classification. rGIN (Sato et al., 2020) pairs each node with a random feature, thereby breaks locally indistinguishable subgraphs. Our method is different from previous approaches in that we exploit the information of entire graphs' node features, achievable with a minimal network depth of 2.

Memory-based neural networks date back to the 1990s (for example (Schmidhuber, 1992)). Notable recent developments include Neural Turing Machine (Graves et al., 2014) and Memory Networks (Weston et al., 2014). Our memory augmentation mechanism could be viewed as a GNN implementation of the design principle in (Weston et al., 2014). There has been a couple of works on GNNs that adopted memory-based design lately (Khasahmadi et al., 2020; Ma et al., 2019a), under different problem contexts.

5 Experiments

In this section we evaluate the performance of memory augmented GNNs using two canonical architectures MemGAT and MemGCN. We also conduct an ablation study to decompose performance gains under different design aspects. We use three types of benchmark datasets, with their characteristic statistics summarized in appendix A.1:

Citation networks Cora, Citeseer, and Pubmed are standard citation network benchmark datasets. For all the three datasets, graphs are constructed by treating documents as nodes and citation links as edges, the task is to classify academic papers into different subjects using the papers' bag-of-words representation as features

Actor co-occurrence networks This dataset is the actor-only induced subgraph of the film-director-actor-writer network. Where nodes correspond to actors, and an edge between two nodes denotes co-occurrence on the same Wikipedia page. Node features correspond to some keywords in the Wikipedia pages and labels are actors categories

Web networks We use Cornell, Texas, and Wisconsin dataset (Pei et al., 2020). where nodes and edges represent web pages and hyperlinks, respectively. The feature of each node is the bag-of-words representation of the corresponding page, and labels are categories of web pages.

5.1 EXPERIMENTAL SETUP

Architecture and parameters We mostly adopt parameters for backbone architectures from their original implementation, respectively GAT (Veličković et al., 2018) and GCN (Kipf & Welling, 2016), detailed in appendix A.1. We applied entropy regularization (Grandvalet & Bengio, 2005) with tuning parameter 0.6 for Cora and Citeseer datasets and 0.1 for Pubmed dataset. We use the following truncated diffusion matrix (Klicpera et al., 2019b) to generate edge weights:

$$S := S(\theta, K, T) := \sum_{k=0}^{K} \theta_k T^k \tag{4}$$

In our experiments we use two sets of θs: *personalized page rank*: $\theta_k = \alpha(1-\alpha)^k$ with $\alpha = 0.15$ and *heat kernel*: $\theta_k = \frac{t^k e^{-t}}{k!}$ with $t = 5$. We choose the transition matrix T to be either $A_{\text{rw}} = (D+I)^{-1}(A+I)$ or $A_{\text{sym}} = (D+I)^{-1/2}(A+I)(D+I)^{-1/2}$. We report the best performing combination for each experiment. We tune the number of random walk steps $K \in \mathbb{N}$. Note that for large K, the resulting matrix is an approximation to the diffusion matrix (Klicpera et al., 2019b).
For the memory component, we used $\lambda = 0.9$, we choose the number of memory nodes to be the number of classes across all tasks, a study of this hyperparameter is presented in appendix A.3. Finally, we applied skip connection using the form proposed in Zhang & Meng (2019) to stabilize training. The detailed tuning strategies are listed in appendix A.1.

Baseline comparisons We compare MemGAT and MemGCN with there backbone architectures across all benchmarks. We also report two state-of-the-art models that are able to aggregate neighborhood information from distant nodes: APPNP (Klicpera et al., 2019a) and GCNII (Chen et al., 2020). For the citation networks, we additionally report GraphSage (Hamilton et al., 2017) with max-pooling aggregation and GIN (Xu et al., 2019) and G^3NN (Ma et al., 2019b). For actor and wiki networks, we additionally report GEOM-GCN (Pei et al., 2020) with its best performing variant. Besides, we report the performance of a two-layer MLP with 256 hidden units on these datasets, as was recently noted that MLP performs very well on actor and wiki networks (Zhu et al., 2020).

Evaluation strategy it was reported in (Shchur et al., 2018) that the performance of current state-of-the-art graph neural networks are typically unstable with respect to different train/validation/test split of the dataset. Hence for fair comparison of algorithm performances, in addition to results on the original, standard split, we provide extensive results over 10 random splits of Cora, Citeseer and Pubmed datasets.

5.2 RESULTS

We use accuracy for performance metrics across all the tasks. The results of our evaluation experiments are summarized in Tables 1 and 2. We report training details in appendix A.1. As shown in table 1, MemGAT and MemGCN consistently outperforms GAT and GCN by a significant margin and is competitive to current state-of-the-art algorithms for graph learning. In particular, on the web network datasets, MemGCN improves GCN with a relative accuracy gain of over 50% while using the same network depth, and outperforms GCNII, which uses deep architectures (16 and 32 layers) on these datasets. Results from random split experiments demonstrate that memory augmented architectures have fairly stable performance.

Ablation study We analyze the contribution of different components of MemGAT or MemGCN in a progressive way, with (I) backbone GNN (original GAT or GCN) (II) backbone GNN with edge weights. (III) backbone GNN with edge weights and skip connections. (IV) full MemGAT or MemGCN, which corresponds to adding memory component to the setup in (III). Table 3 records results on the citation datasets. The results demonstrate significant improvement provided by incorporating the memory component.

6 CONCLUSIONS

We introduced memory augmentation, a framework that extends GNNs with a memory component that aims at incorporating global information of the graph. Our method has the advantage of

Table 1: Summary of results (%mean $\pm$ %standard deviation test set accuracy) for the citation datasets, with 10 random train/validation/test split of datasets as well as the standard split. We annotate the original paper where we take the results from aside the algorithm name

Algorithm	**Cora**	**Citeseer**	**Pubmed**
Random split			
GCN (Shchur et al., 2018)	81.5 ± 1.3	71.9 ± 1.9	77.8 ± 2.9
GraphSage(maxpool) (Shchur et al., 2018)	76.6 ± 1.9	67.5 ± 2.3	76.1 ± 2.3
GAT (Shchur et al., 2018)	81.8 ± 1.3	71.4 ± 1.9	78.7 ± 2.3
GIN (Klicpera et al., 2019b)	73.96 ± 0.46	61.09 ± 0.58	72.38 ± 0.63
APPNP (our run)	82.45 ± 1.89	70.60 ± 1.31	77.28 ± 3.14
MemGAT(ours)	$\mathbf{83.94 \pm 1.04}$	$\mathbf{74.07 \pm 1.1}$	$\mathbf{79.18 \pm 2.11}$
MemGCN(ours)	82.94 ± 1.61	72.87 ± 1.69	78.53 ± 1.91
Standard split			
GCN (Shchur et al., 2018)	81.9 ± 0.8	69.5 ± 0.9	79.0 ± 0.5
GraphSage(maxpool) (Shchur et al., 2018)	77.4 ± 1.0	67.0 ± 1.0	76.6 ± 0.8
GAT (Shchur et al., 2018)	82.8 ± 0.5	71.0 ± 0.6	77.0 ± 0.3
APPNP (our run)	83.60 ± 0.61	72.46 ± 0.52	79.04 ± 0.53
G^3NN(GAT) (Ma et al., 2019b).	82.9 ± 0.3	74.0 ± 0.3	77.4 ± 0.4
G^3NN(GCN) (Ma et al., 2019b).	82.2 ± 0.3	74.5 ± 0.3	78.4 ± 0.4
GCNII (Chen et al., 2020)	$\mathbf{85.5}$	73.4	80.3
MemGAT(ours)	84.65 ± 0.52	74.20 ± 0.73	79.18 ± 0.56
MemGCN(ours)	84.30 ± 0.53	$\mathbf{75.12 \pm 0.22}$	$\mathbf{80.46 \pm 0.27}$

Table 2: Summary of results (%mean $\pm$ %standard deviation test set accuracy) for actor co-occurence and web network datasets, with 10 random train/validation/test split of datasets. We annotate the original paper where we take the results from aside the algorithm name

Algorithm	**Actor**	**Cornell**	**Texas**	**Wisconsin**
GCN (Pei et al., 2020)	26.86	52.70	52.16	45.88
GAT (Pei et al., 2020)	28.45	54.32	58.38	49.41
APPNP (Chen et al., 2020)	31.31	73.51	65.41	69.02
GEOM-GCN (Pei et al., 2020)	31.63	60.81	67.57	64.12
GCNII (Chen et al., 2020)	NA	76.49	77.84	81.57
MLP (our run)	$\mathbf{37.01 \pm 1.02}$	80.65 ± 6.34	80.84 ± 4.21	84.31 ± 3.41
MemGAT(ours)	31.93 ± 1.69	70.37 ± 7.57	72.08 ± 8.02	73.80 ± 5.05
MemGCN(ours)	36.95 ± 0.95	$\mathbf{81.92 \pm 6.00}$	$\mathbf{82.19 \pm 5.19}$	$\mathbf{84.96 \pm 3.59}$

stronger expressive power than GNNs based on LUMP protocol and is applicable to standard GNN architectures like GAT and GCN. Experimental results reflect our theoretical motivations.

Table 3: Summary of the ablation study results (%mean $\pm$ %standard deviation test set accuracy) for the citation datasets over 100 trials. For MemGCN, we did not use skip connection on the citation datasets; for MemGAT, we did not incorporate edge weights on the pubmed dataset.

Model	**MemGAT**			**MemGCN**		
Dataset	Cora	Citeseer	Pubmed	Cora	Citeseer	Pubmed
(I)	82.32 ± 0.57	71.11 ± 0.97	77.00 ± 0.30	81.99 ± 0.52	69.50 ± 0.90	79.00 ± 0.50
(II)	83.54 ± 0.55	71.76 ± 0.48	N/A	82.26 ± 0.88	72.68 ± 0.27	79.95 ± 0.23
(III)	83.77 ± 0.72	73.21 ± 0.66	78.97 ± 1.05	N/A	N/A	N/A
(IV)	84.65 ± 0.52	74.20 ± 0.73	79.18 ± 0.56	84.30 ± 0.53	75.12 ± 0.22	80.46 ± 0.27

REFERENCES

Ming Chen, Zhewei Wei, Zengfeng Huang, Bolin Ding, and Yaliang Li. Simple and deep graph convolutional networks. *arXiv preprint arXiv:2007.02133*, 2020.

Nima Dehmamy, Albert-László Barabási, and Rose Yu. Understanding the representation power of graph neural networks in learning graph topology. In *Advances in Neural Information Processing Systems*, pp. 15413–15423, 2019.

Claire Donnat, Marinka Zitnik, David Hallac, and Jure Leskovec. Learning structural node embeddings via diffusion wavelets. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pp. 1320–1329, 2018.

David K Duvenaud, Dougal Maclaurin, Jorge Iparraguirre, Rafael Bombarell, Timothy Hirzel, Alán Aspuru-Guzik, and Ryan P Adams. Convolutional networks on graphs for learning molecular fingerprints. In *Advances in neural information processing systems*, pp. 2224–2232, 2015.

Vikas K Garg, Stefanie Jegelka, and Tommi Jaakkola. Generalization and representational limits of graph neural networks. *arXiv preprint arXiv:2002.06157*, 2020.

Yves Grandvalet and Yoshua Bengio. Semi-supervised learning by entropy minimization. In *Advances in neural information processing systems*, pp. 529–536, 2005.

Alex Graves, Greg Wayne, and Ivo Danihelka. Neural turing machines. *arXiv preprint arXiv:1410.5401*, 2014.

Jonathan L Gross and Thomas W Tucker. *Topological graph theory*. Courier Corporation, 2001.

Aditya Grover and Jure Leskovec. node2vec: Scalable feature learning for networks. In *Proceedings of the 22nd ACM SIGKDD international conference on Knowledge discovery and data mining*, pp. 855–864, 2016.

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

Amir Hosein Khasahmadi, Kaveh Hassani, Parsa Moradi, Leo Lee, and Quaid Morris. Memory-based graph networks. In *International Conference on Learning Representations*, 2020. URL https://openreview.net/forum?id=r1laNeBYPB.

Thomas N Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. *arXiv preprint arXiv:1609.02907*, 2016.

Johannes Klicpera, Aleksandar Bojchevski, and Stephan Günnemann. Combining neural networks with personalized pagerank for classification on graphs. In *International Conference on Learning Representations*, 2019a. URL https://openreview.net/forum?id=H1gL-2A9Ym.

Johannes Klicpera, Stefan Weiß enberger, and Stephan Günnemann. Diffusion improves graph learning. In *Advances in Neural Information Processing Systems 32*, pp. 13354–13366. Curran Associates, Inc., 2019b. URL http://papers.nips.cc/paper/9490-diffusion-improves-graph-learning.pdf.

Johannes Klicpera, Janek Groß, and Stephan Günnemann. Directional message passing for molecular graphs. In *International Conference on Learning Representations*, 2020. URL https://openreview.net/forum?id=B1eWbxStPH.

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

Chen Ma, Liheng Ma, Yingxue Zhang, Jianing Sun, Xue Liu, and Mark Coates. Memory augmented graph neural networks for sequential recommendation, 2019a.

Jiaqi Ma, Weijing Tang, Ji Zhu, and Qiaozhu Mei. A flexible generative framework for graph-based semi-supervised learning. In *Advances in Neural Information Processing Systems 32*, pp. 3281–3290. Curran Associates, Inc., 2019b. URL http://papers.nips.cc/paper/8590-a-flexible-generative-framework-for-graph-based-semi-supervised-learning.pdf.

Haggai Maron, Heli Ben-Hamu, Hadar Serviansky, and Yaron Lipman. Provably powerful graph networks. In *Advances in Neural Information Processing Systems 32*, pp. 2156–2167. Curran Associates, Inc., 2019. URL http://papers.nips.cc/paper/8488-provably-powerful-graph-networks.pdf.

Christopher Morris, Martin Ritzert, Matthias Fey, William L Hamilton, Jan Eric Lenssen, Gaurav Rattan, and Martin Grohe. Weisfeiler and leman go neural: Higher-order graph neural networks. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pp. 4602–4609, 2019.

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

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*, 2020. URL https://openreview.net/forum?id=S1e2agrFvS.

Ryoma Sato, Makoto Yamada, and Hisashi Kashima. Approximation ratios of graph neural networks for combinatorial problems. In *Advances in Neural Information Processing Systems*, pp. 4083–4092, 2019.

Ryoma Sato, Makoto Yamada, and Hisashi Kashima. Random features strengthen graph neural networks, 2020.

F. Scarselli, M. Gori, A. C. Tsoi, M. Hagenbuchner, and G. Monfardini. The graph neural network model. *IEEE Transactions on Neural Networks*, 20(1):61–80, 2009.

Jürgen Schmidhuber. Learning to control fast-weight memories: An alternative to dynamic recurrent networks. *Neural Computation*, 4(1):131–139, 1992.

Oleksandr Shchur, Maximilian Mumme, Aleksandar Bojchevski, and Stephan Günnemann. Pitfalls of graph neural network evaluation. *arXiv preprint arXiv:1811.05868*, 2018.

Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, and Yoshua Bengio. Graph attention networks. In *International Conference on Learning Representations*, 2018. URL https://openreview.net/forum?id=rJXMpikCZ.

Jason Weston, Sumit Chopra, and Antoine Bordes. Memory networks. *arXiv preprint arXiv:1410.3916*, 2014.

Keyulu Xu, Chengtao Li, Yonglong Tian, Tomohiro Sonobe, Ken-ichi Kawarabayashi, and Stefanie Jegelka. Representation learning on graphs with jumping knowledge networks. In Jennifer Dy and Andreas Krause (eds.), *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pp. 5453–5462, Stockholmsmässan, Stockholm Sweden, 10–15 Jul 2018. PMLR. URL http://proceedings.mlr.press/v80/xu18c.html.

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

Jiaxuan You, Rex Ying, and Jure Leskovec. Position-aware graph neural networks. In Kamalika Chaudhuri and Ruslan Salakhutdinov (eds.), *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pp. 7134–7143, Long Beach, California, USA, 09–15 Jun 2019. PMLR. URL http://proceedings.mlr.press/v97/you19b.html.

Jiawei Zhang and Lin Meng. Gresnet: Graph residual network for reviving deep gnns from suspended animation. *ArXiv, abs/1909.05729*, 2019.

Shuo Zhang and Lei Xie. Improving attention mechanism in graph neural networks via cardinality preservation. *International Joint Conferences on Artificial Intelligence*, 2020.

Jiong Zhu, Yujun Yan, Lingxiao Zhao, Mark Heimann, Leman Akoglu, and Danai Koutra. Generalizing graph neural networks beyond homophily, 2020.