Graph Meta Learning via Local Subgraphs

Kexin Huang
Harvard University
`kexinhuang@hsph.harvard.edu`

Marinka Zitnik
Harvard University
`marinka@hms.harvard.edu`

Abstract

Prevailing methods for graphs require abundant label and edge information for learning. When data for a new task are scarce, meta learning can learn from prior experiences and form much-needed inductive biases for fast adaption to new tasks. Here, we introduce G-META, a novel meta-learning algorithm for graphs. G-META uses local subgraphs to transfer subgraph-specific information and learn transferable knowledge faster via meta gradients. G-META learns how to quickly adapt to a new task using only a handful of nodes or edges in the new task and does so by learning from data points in other graphs or related, albeit disjoint label sets. G-META is theoretically justified as we show that the evidence for a prediction can be found in the local subgraph surrounding the target node or edge. Experiments on seven datasets and nine baseline methods show that G-META outperforms existing methods by up to 16.3%. Unlike previous methods, G-META successfully learns in challenging, few-shot learning settings that require generalization to completely new graphs and never-before-seen labels. Finally, G-META scales to large graphs, which we demonstrate on a new Tree-of-Life dataset comprising of 1,840 graphs, a two-orders of magnitude increase in the number of graphs used in prior work.

1   Introduction

Graph Neural Networks (GNNs) have achieved remarkable results on many tasks such as recommender systems [54], molecular predictions [63], and knowledge graphs [47]. Performance in such tasks is typically evaluated after extensive training on datasets where majority of labels are available [50, 52]. In contrast, many problems require rapid learning from only a few labeled nodes or edges in the graph. Such flexible adaptation, known as meta-learning, has been extensively studied for images and language, *e.g.*, [37, 49, 25]. However, meta-learning on graphs has received considerably less research attention and has remained a problem beyond the reach of prevailing GNN models.

Meta-learning on graphs generally refers to a scenario in which a model learns at two levels. In the first level, rapid learning occurs *within* a task. For example, when a GNN learns to classify nodes in a particular graph accurately. In the second level, this learning is guided by knowledge accumulated gradually *across* tasks that captures how task structure changes across target domains [33, 4, 32]. A powerful GNN trained to meta-learn can quickly learn never-before-seen labels and relations using a low number of data points. As an example, a key problem in biology is to translate insights from non-human organisms (such as yeast, zebrafish, and mouse) to humans [64]. How to train a GNN to effectively meta-learn on a large number of incomplete and scarcely labeled protein-protein interaction (PPI) graphs from various organisms, transfer the accrued knowledge to humans, and use it to predict the roles of protein nodes in the human PPI graph? While this is a hard unsolved task, it is only an instance of a particular graph meta-learning problem (Figure 1C). In this problem, the GNN needs to learn on a large number of graphs, each scarcely labeled with a unique label set. Then, it needs to quickly adapt to a never-before-seen graph (*e.g.*, a PPI graph from a new species) and never-before-seen labels (*e.g.*, newly discovered roles of proteins). Current methods are specialized techniques specifically designed for a particular problem and a particular task [61, 3, 22, 7, 51].
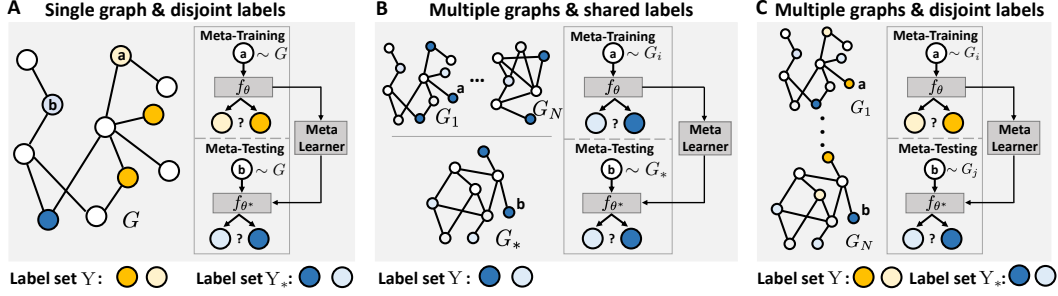
Figure 1: Graph meta-learning problems. **A.** Meta-learner classifies unseen label set by observing other label sets in the same graph. **B.** Meta-learner learns unseen graph by learning from other graphs with the same label set. **C.** Meta-learner classifies unseen label set by learning from other label sets across multiple graphs. Unlike existing methods, G-META solves all three problems and also works for link prediction (see Section 6.1).

While these methods provide a promising approach to meta learning in GNNs, their specific strategy does not scale well nor extend to other problems (Figure 1).

Present work. Here, we introduce G-META,[1] a general approach for a variety of graph meta-learning problems (Figure 1). The core principle of G-META is to represent every node with a local subgraph and use subgraphs to train GNNs to meta-learn. In contrast to G-META, current methods are trained to meta-learn on entire graphs. As we show theoretically and empirically, these methods are unlikely to succeed in few-shot learning settings when the labels are scarce and scattered around multiple graphs. Furthermore, these methods capture the overall graph structure, but at the loss of finer local structures. Our theoretical analysis (Section 4) suggests that the evidence for a particular prediction can be found in the subgraph surrounding the target node or edge when using GNNs. Besides, G-META's construction of local subgraphs gives local structural representations that enable direct structural similarity comparison using GNNs, based on its connection to Weisfeiler-Lehman test [52, 58]. Further, structural similarity enables G-META to form the much-needed inductive bias by using a metric-learning algorithm [35]. Moreover, local subgraphs also allow for effective feature propagation and label smoothing within a GNN.

(1) G-META is general. While previous meta graph learning methods [61, 3] apply only to one graph meta-learning problem (Figure 1), G-META works for all of them (Appendix B). (2) G-META yields powerful predictors. We demonstrate G-META's performance on seven datasets and nine baselines, where G-META considerably outperform baselines by up to 16.3%. (3) G-META is scalable. By operating on subgraphs, only small neighborhoods need to be examined by the GNN. We show how G-META scales to large graphs by applying it to our new Tree-of-Life dataset comprising of 1,840 graphs, a two-orders of magnitude increase in the number of graphs used in prior work.

2 Related Work

(1) <u>Few-shot meta learning</u>. Few-shot meta-learning learns from prior experiences and transfers them to quickly learn a new but related task with only a few examples [44, 39]. Meta-learning methods generally fall into three categories, model-based [14, 48, 32], metric-based [35, 36, 45], and optimization-based [30, 27, 12]. (2) <u>Meta learning for graphs</u>. Recent studies integrate graphs into meta-learning. Liu *et al.* [22] constructs image category graphs to improve few-shot learning. Zhou *et al.* [60] uses graph meta learning for fast network alignment. MetaR [7] and GMatching [51] use metric methods to generalize over new relations given from a handful of associative relations in a knowledge graph. While these methods learn from a single graph from a specific domain, G-META can handle general settings with many graphs and disjoint label sets. Further, Chauhan *et al.* [5] uses a super-class prototypical network for graph classification. In contrast, G-META focuses on node classification and link prediction. More relevant to G-META, Meta-GNN [61] uses gradient-based meta-learning for few-shot node classification in the single graph and disjoint label problem (Figure 1), GFL [53] focuses on the shared label multiple graph problem and Meta-Graph [3] uses graph signature functions for few-shot link prediction in multiple graph shared label problem. In contrast, G-META works for many more problems. Also, note that Meta-GNN operates a GNN on the

---

entire graph whereas G-META extracts relevant local subgraphs first and then applies GNN on each subgraph individually. For Meta-GNN, a task is a batch of nodes embedding, whereas for G-META, it is a batch of subgraphs. This difference allows for rapid adaptation to new tasks, improves scalability, and applies broadly to various meta learning problems. Meta-GNN only works for a single graph disjoint label node classification whereas G-META works for all problems in Figure 1, both for node classification and link prediction. (3) <u>Subgraphs and GNNs</u>. Modeling of subgraph structure is vital for numerous graph-related tasks, *e.g.*, [43, 9, 56, 38]. For example, Patchy-San [28] uses local receptive fields to extract useful features from graphs. Ego-CNN uses ego graphs to find critical structures [41]. SEAL [59] develops a theory showing that enclosing subgraphs capture graph heuristics, which we extend to GNNs here. Cluster-GCN [8] and GraphSAINT [57] use subgraphs to improve GNN scalability. In contrast, G-META is the first to use subgraphs for meta learning.

3 Background and Problem Formulation

Let $\mathcal{G} = \{G_1, \ldots, G_N\}$ denotes N graphs. For each graph $G = (\mathcal{V}, \mathcal{E}, \mathbf{X})$, $\mathcal{V}$ is the set of nodes, $\mathcal{E}$ is the set of edges and $\mathbf{X} = \{\mathbf{x}_1, \ldots, \mathbf{x}_n\}, \mathbf{x}_u \in \mathbb{R}^d$ is the d-dimensional node feature for each node in $\mathcal{V}$. We denote $\mathcal{Y} = \{Y_1, \ldots, Y_M\}$ as M labels and a label set Y is a set of labels selected from $\mathcal{Y}$. The G-META's core principle is to represent nodes with local subgraphs and then use subgraphs to easily transfer knowledge across tasks, graphs, and label sets. We use $\mathcal{S}$ to denote local subgraphs for nodes, $\mathcal{S} = \{S_1, \ldots, S_n\}$. Hence, the classification task is to learn a GNN $f_\theta : \mathcal{S} \mapsto \{1, \ldots, |Y|\}$, *i.e.*, to correctly map node u's local subgraph S_u to labels in Y given only a handful of labeled nodes.

Background on Graph Neural Networks. GNNs learn compact representations, *i.e.*, embeddings, that capture network structure and node features. The input to GNN is a graph G. The output is generated via a series of propagation layers [13]. Propagation at layer l consists of three steps: (1) <u>Message passing</u>. GNN computes a message $\mathbf{m}_{uv}^{(l)} = \text{MSG}(\mathbf{h}_u^{(l-1)}, \mathbf{h}_v^{(l-1)})$ for every linked nodes u, v based on their embeddings from the previous layer $\mathbf{h}_u^{(l-1)}$ and $\mathbf{h}_v^{(l-1)}$. (2) <u>Neighborhood aggregation</u>. The messages between node u and its neighbors $\mathcal{N}_u$ are aggregated as $\hat{\mathbf{m}}_u^{(l)} = \text{AGG}(\mathbf{m}_{uv}^{(l)} | v \in \mathcal{N}_u)$. (3) <u>Update</u>. Finally, GNN uses a non-linear function to update embeddings $\mathbf{h}_u^{(l)} = \text{UPD}(\hat{\mathbf{m}}_u^{(l)}, \mathbf{h}_u^{(l-1)})$ using the aggregated message and the embedding from the previous layer, for each node u.

Background on Meta-learning. In meta-learning, we have a meta-set $\mathscr{D}$ that consists of $\mathscr{D}_{\text{train}}, \mathscr{D}_{\text{val}}, \mathscr{D}_{\text{test}}$. This meta-set consists of many tasks. Each task $\mathcal{T}_i \in \mathscr{D}$ can be divided into $\mathcal{T}_i^{\text{support}}$ and $\mathcal{T}_i^{\text{query}}$. $\mathcal{T}_i^{\text{support}}$ has $\text{K}_{\text{support}}$ labeled data points in each label for learning and $\mathcal{T}_i^{\text{query}}$ has K_{query} data points in each label for evaluation. The size of the label set for the meta-set $|Y|$ is N. It is also called N-ways $\text{K}_{\text{support}}$-shots learning problem. During meta-training, for $\mathcal{T}_i$ from $\mathscr{D}_{\text{train}}$, the model first learns from $\mathcal{T}_i^{\text{support}}$ and then evaluates on $\mathcal{T}_i^{\text{query}}$ to see how well the model performs on that task. The goal of Model-Agnostic Meta-Learning (MAML) [12] is to obtain a parameter initialization θ_* that can adapt to unseen tasks quickly, such as $\mathscr{D}_{\text{test}}$, using gradients information learnt during meta-training. Hyperparameters are tuned via $\mathscr{D}_{\text{val}}$.

3.1 G-META: Problem Formulation

G-META is an approach that can tackle three fundamentally different meta-learning problems (Figure 1). Shared label refers that every task shares the same label set Y. Disjoint label refers to the situation where the label sets for tasks i and j are disjoint, *i.e.*, $Y_i \cap Y_j = \emptyset$. Each data point in a task $\mathcal{T}_i$ is a local subgraph S_u, along with its associated label Y_u. The overall goal is to adapt to an unseen task $\mathcal{T}_* \sim p(\mathcal{T})$ with a few examples after observing related tasks $\mathcal{T}_i \sim p(\mathcal{T})$.

Graph meta-learning problem 1: Single Graph and Disjoint Labels. We have a graph G with a distribution of label set $p(Y|G)$. The goal is to adapt to an unseen label set $Y_* \sim p(Y|G)$ by learning from tasks with other label sets $Y_i \sim p(Y|G)$, where $Y_i \cap Y_* = \emptyset$ for every label set Y_i.

Graph meta-learning problem 2: Multiple Graphs and Shared Labels. We have a distribution of graphs $p(G)$ but one label set Y. The goal is to learn from another graph $G_j \sim p(G)$ and quickly adapt to an unseen graph $G_* \sim p(G)$, where G_j and G_* are disjoint. All tasks share the same labels.

Graph meta-learning problem 3: Multiple Graphs and Disjoint Labels. We have a distribution of label sets $p(Y|\mathcal{G})$ conditioned on multiple graphs $\mathcal{G}$. In contrast to Problem 1, each task has its own label set Y_i but those label sets can appear in one or more graphs. The goal is to adapt to an unseen label set $Y_* \sim p(Y|\mathcal{G})$ after learning from some other label set $Y_i \sim p(Y|\mathcal{G})$, where $Y_i \cap Y_* = \emptyset$.

4 Local Subgraphs and Theoretical Motivation for G-META

We start by describing how to construct local subgraphs in G-META. We then provide a theoretical justification showing that local subgraphs preserve useful information from the entire graph. We then argue how subgraphs enable G-META to capture sufficient information on the graph structure, node features, and labels, and use that information for graph meta-learning.

For node u, a local subgraph is defined as a subgraph $S_u = (\mathcal{V}^u, \mathcal{E}^u, \mathbf{X}^u)$ induced from a set of nodes $\{v|d(u,v) \leq h\}$, where $d(u,v)$ is the shortest path distance between node u and v, and h defines the neighborhood size. In a meta-task, subgraphs are sampled from graphs or label sets, depending on the graph meta-learning problems defined in Section 3. We then use GNNs to encode the local subgraphs. However, one straightforward question raised is if this subgraph loses information by excluding nodes outside of it. Here, we show in theory that applying GNN on the local subgraph preserve useful information compared to using GNN on the entire graph.

Preliminaries and definitions. Next, we use GCN [20] as an exemplar GNN to understand how nodes influence each other during neural message passing. The assumptions are based on [46] and are detailed in Appendix C. We need the following definitions. (1) <u>Node influence</u> $I_{u,v}$ of v on u in the final GNN output is: $I_{u,v} = \|\partial \mathbf{x}_u^{(\infty)} / \partial \mathbf{x}_v^{(\infty)}\|$, where the norm is any subordinate norm and the Jacobian measures how a change in v translates to a change in u [46]. (2) <u>Graph influence</u> I_G on u is: $I_G(u) = \|[I_{u,v_1}, \ldots, I_{u,v_n}]\|_1$, where $[I_{u,v_1}, \ldots, I_{u,v_n}]$ is a vector representing the influence of other nodes on u. (3) <u>Graph influence loss</u> R_h is defined as: $R_h(u) = I_G(u) - I_{S_u}(u)$, where $I_G(u)$ is the influence of entire graph G, and $I_{S_u}(u)$ is the influence of local subgraph S_u. Next, we show how influence spreads between nodes depending on how far the nodes are from each other in a graph.

Theorem 1 (Decaying Property of Node Influence). *Let t be a path between node u and node v and let D_{GM}^t be a geometric mean of node degrees occurring on path t. Let $D_{\mathrm{GM}}^{t_*} = \min_t\{D_{\mathrm{GM}}^t\}$ and $h_* = d(u,v)$. Consider the node influence $I_{u,v}$ from v to u. Then, $I_{u,v} \leq C/(D_{\mathrm{GM}}^{t_*})^{h_*}$.*

The proof is deferred to Appendix C. Theorem 1 states that the influence of a node v on node u decays exponentially as their distance h_* increases. A side theoretical finding is that the influence is significantly decided by the accumulated node degrees of the paths between two nodes. In other words, if the paths are straight lines of nodes (low accumulated node degrees), then the node influence is high. Otherwise, if the paths consist of numerous connections to other nodes (high accumulated node degrees), the node influence is minimum. Intuitively, high accumulated node degrees can bring complicated messages along the paths, which dampen each individual node influence whereas low degrees paths can pass the message directly to the target node. Since real-world graphs are usually complicated graphs with relatively high node degrees, the node influence will be considerably low. We then show that for a node u, operating GNN on a local subgraph does not lose useful information than operating GNN on the entire graph.

Theorem 2 (Local Subgraph Preservation Property). *Let S_u be a local subgraph for node u with neighborhood size h. Let node v be defined as: $v = \mathrm{argmax}_w(\{I_{u,w}|w \in \mathcal{V} \setminus \mathcal{V}^u\})$. Let $\bar{t}$ be a path between u and v and let $D_{\mathrm{GM}}^{\bar{t}}$ be a geometric mean of node degrees occurring on path $\bar{t}$. Let $D_{\mathrm{GM}}^{\bar{t}_*} = \min_{\bar{t}}\{D_{\mathrm{GM}}^{\bar{t}}\}$. The following holds: $R_h(u) \leq C/(D_{\mathrm{GM}}^{\bar{t}_*})^{h+1}$.*

The proof is deferred to Appendix D. Theorem 2 states that the graph influence loss is bounded by an exponentially decaying term as h increases. In other words, the local subgraph formulation is an h-th order approximation of applying GNN for the entire graph.

Local subgraphs enable few-shot meta-learning. Equipped with the theoretical justification, we now describe how the local subgraph allows G-META to do few-shot meta-learning. The construction of local subgraphs captures the following. (1) <u>Structures</u>. Graph structures present an alternative source of strong signal for prediction [9, 43], especially when node labels are scarce. The GNN representations cannot fully capture large graphs structure because they are too complicated [3, 52]. However, they can learn representations that capture the structure of small graphs, such as our
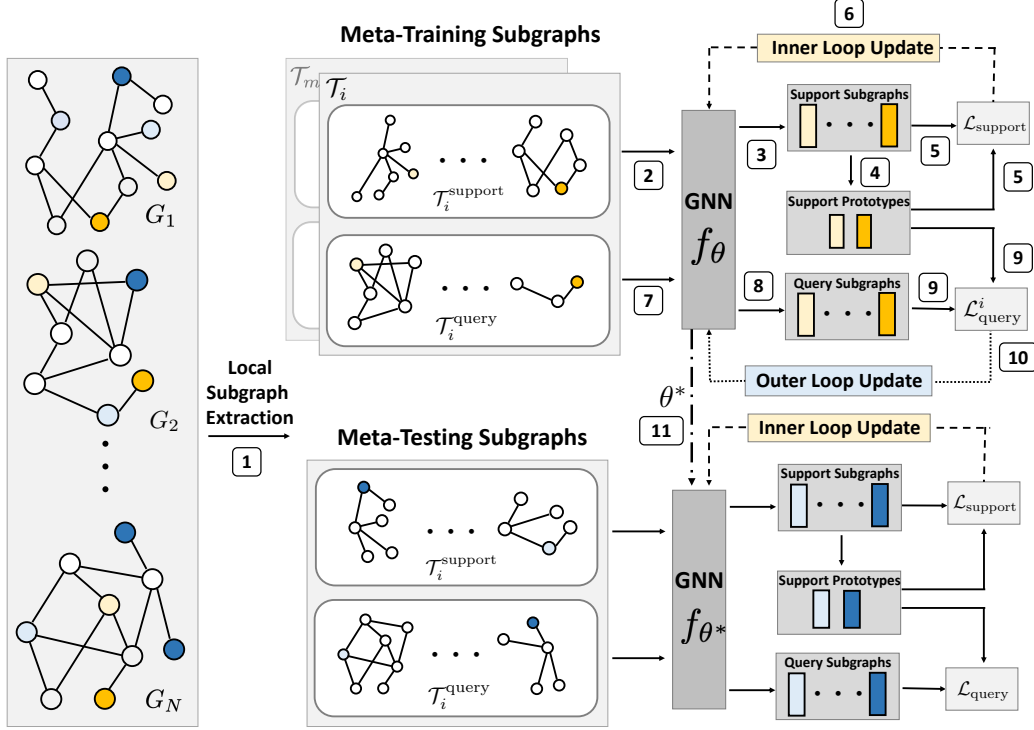
Figure 2: (1) We first construct a batch of m meta-training tasks and extract local subgraphs on the fly for nodes in the meta-tasks. For each task $\mathcal{T}_i$, (2) subgraphs from the support set are mini-batched and are fed into a GNN parameterized by θ. (3) The support set embeddings using the centroid nodes are generated, and (4) the prototypes are computed from the support centroid embeddings. Then, (5) the support set loss $\mathcal{L}_{\text{support}}$ is computed, and (6) back-propagates to update the GNN parameter. (7) $\mathcal{T}_i^{\text{query}}$ subgraphs then feed into the updated GNN to (8) generate query centroid embeddings. (9) Using the support prototypes and the query embeddings, the query loss $\mathcal{L}_{\text{query}}^i$ for task $\mathcal{T}_i$ is computed. Steps (2-9) are repeated for η update steps. The same process repeats for the other m sampled tasks, starting from the same GNN f_θ. (10) The last update step's query loss from all the tasks are summed up and used to update θ. Then, another batch of tasks are sampled, and step (1-10) are repeated. Then, for meta-testing tasks, steps (1-9) are repeated with the GNN using the meta-learned parameter θ_*, which enables generalization over unseen tasks. See Algorithm 1 (Appendix E).

local subgraphs, as is evidenced by the connection to the Weisfeiler-Lehman test [52, 58]. Hence, subgraphs enable G-META to capture structrual node information. (2) <u>Features</u>. Local subgraphs preserve useful information, as indicated by the theorems above. (3) <u>Labels</u>. When only a handful of nodes are labeled, it is challenging to efficiently propagate the labels through the entire graph [62, 21]. Metric-learning methods [35] learn a task-specific metric to classify query set data using the closest point from the support set. It has been proved as an effective inductive bias [35, 40]. Equipped with subgraph representations that capture both structure and feature information, G-META uses metric-learning by comparing the query subgraph embedding to the support subgraph embedding. As such, it circumvents the problem of having too little label information for effective propagation.

5 G-META: Meta Learning via Local Subgraphs

G-META (Figure 2) is an approach for meta-learning on graphs. Building on theoretical motivation from Section 4, G-META first constructs local subgraphs. It then uses a GNN encoder to generate embeddings for subgraphs. Finally, it uses prototypical loss for inductive bias and MAML for knowledge transfer across graphs and labels. The overview is in Algorithm 1 (Appendix E).

Neural encoding of subgraphs. In each meta-task, we first construct subgraphs for each node. While we use h-hops neighbors to construct subgraphs, they are not restricted to any other subgraph extraction algorithms [6, 11]. Then, each subgraph S_u is fed into a h-layer GNN to obtain an embedding for every node in the subgraph. h is set to be the subgraph neighborhood size. The

centroid node u's embedding is used to represent the subgraph $\mathbf{h}_u = \text{GNN}(S_u)_{\text{centroid}}$. Note that we emphasize on proposing a general framework for subgraph embedding where a centroid node embedding is an instantiation of this framework. One can try any subgraph representation such as a readout function over every node in the subgraph [52, 5] or through subgraph neural networks [1, 55]. Note that for faster computation, we mini-batch the subgraphs for each meta-task's support and query set. Notably, the idea of using local subgraphs in our context is different from the idea of computation graphs since local subgraphs are not used here for a new message passing scheme but are used as a novel GNN framework to unlock many graph meta-learning problems. Without constructing local subgraphs, it is unable to do many graph meta learning problems. We are also not limiting the formulation of subgraph and subgraph neural encoding whereas computation graph is a rigid operation on the h-hop neighbors.

Prototypical loss. After we obtain a sufficient subgraph representation, we further leverage the inductive bias between representation and labels to circumvent the problem of scarce label information for effective propagation in the few-shot settings. For each label k, we take the mean of the support set subgraph embeddings to obtain prototype $\mathbf{c}_k$: $\mathbf{c}_k = 1/N_k \sum_{y_j=k} \mathbf{h}_j$. The prototype $\mathbf{c}_k$ serves as a landmark for the label k. Then, for each local subgraph S_u in both support and query set, a class distribution vector $\mathbf{p}$ is computed via the euclidean distance between the support prototypes for each class and the subgraph centroid embeddings: $\mathbf{p}_k = \frac{\exp(-\|\mathbf{h}_u - \mathbf{c}_k\|)}{\sum_{\hat{k}} \exp\left(-\|\mathbf{h}_u - \mathbf{c}_{\hat{k}}\|\right)}$. After obtaining a prediction probability distribution over label classes $\mathbf{p}$ for each local subgraph S_u, a cross-entropy loss is used to backpropagate $\text{L}(\mathbf{p}, \mathbf{y}) = \sum_j \mathbf{y}_j \log \mathbf{p}_j$, where $\mathbf{y}$ is the true label's one hot encoding.

Optimization-based meta-learning. To transfer the structural knowledge across graphs and labels, we use MAML, an optimization-based meta-learning approach. We break the node's dependency on a graph by framing nodes into independent local subgraph classification tasks. It allows direct adaptation to MAML since individual subgraphs can be considered as an individual image in the classic few-shot meta-learning setups. More specifically, we first sample a batch of tasks, where each task consists of a set of subgraphs. During meta-training inner loop, we perform the regular stochastic gradient descent on the support loss for each task $\mathcal{T}_i$: $\theta_j = \theta_{j-1} - \alpha \nabla \mathcal{L}_{\text{support}}$. The updated parameter is then evaluated using the query set, and the query loss for task i is recorded as $\mathcal{L}^i_{\text{query}}$. The above steps are repeated for η update steps. Then, the $\mathcal{L}^i_{\text{query}}$ from the last update step is summed up across the batch of tasks, and then we perform a meta-update step: $\theta = \theta - \beta \nabla \sum_i \mathcal{L}^i_{\text{query}}$. Next, new tasks batch are sampled, and the same iteration is applied on the meta-updated θ. During meta-testing, the same procedure above is applied using the final meta-updated parameter θ_*. θ_* is learned from knowledge across meta-training tasks and is the optimal parameter to adapt to unseen tasks quickly.

Attractive properties of G-META. (1) <u>Scalability:</u> G-META operates on a mini-batch of local subgraphs where the subgraph size and the batch size (few-shots) are both small. This allows fast computation and low memory requirement because G-META's aggregation field is smaller than that of previous methods that operate on entire graphs [3, 61]. We further increase scalability by sub-sampling subgraphs with more than 1,000 nodes. Also note that graph few-shot learning does not evaluate all the samples in the graph since few-shot learning means majority of labels do not exist. Thus, each mini-batch consists only of a few samples with labels (e.g., 9 in 3-way 3-shot learning), which are quick operations. (2) <u>Inductive learning:</u> Since G-META operates on different subgraphs in each GNN encoding, it forces inductiveness over unseen subgraphs. This inductiveness is crucial for few-shot learning, where the trained model needs to adapt to unseen nodes. Inductive learning also allows for knowledge transfer from meta-training subgraphs to meta-testing subgraphs. (3) <u>Over-smoothing regularization:</u> One limitation of GNN is that connected nodes become similar after iterations of propagation on the same entire graph. In contrast, each iteration for G-META consists of a batch of different subgraphs with various structures, sizes and nodes, where each subgraph is fed into the GNN individually. This prevents GNN to over-smooth on a single graph structure. (4) <u>Few-shot learning:</u> G-META needs only a tiny number of labeled nodes for successful learning, as demonstrated in experiments. This property is in contrast with prevailing GNNs, which require a large fraction of labeled nodes to propagate neural messages in the graph successfully. (5) <u>Broad applicability:</u> G-META applies to many graph meta-learning problems (Figure 1) whereas previous methods apply to at most one [3, 61]. Unlike earlier methods, G-META works for node classification and also few-shot link prediction (*i.e.*, via local subgraphs for a pair of nodes).

Table 1: Dataset statistics. Fold-PPI and Tree-of-Life are new datasets introduced in this study.

Dataset	Task	# Graphs	# Nodes	# Edges	# Features	# Labels
Synthetic Cycle	Node	10	11,476	19,687	N/A	17
Synthetic BA	Node	10	2,000	7,647	N/A	10
ogbn-arxiv	Node	1	169,343	1,166,243	128	40
Tissue-PPI	Node	24	51,194	1,350,412	50	10
FirstMM-DB	Link	41	56,468	126,024	5	2
Fold-PPI	Node	144	274,606	3,666,563	512	29
Tree-of-Life	Link	1,840	1,450,633	8,762,166	N/A	2

6 Experiments

Synthetic datasets. We have two synthetic datasets whose labels depend on nodes' structural roles [16], which we use to confirm that G-META captures local graph structure (Table 1). (1) <u>Cycle</u>: we use a cycle basis graph and attach a distribution of shapes: House, Star, Diamond, Fan [9]. The label of each node is the structural role define by the shape. We also add random edges as noise. In the multiple graphs problem, each graph is with varying distribution of number of shapes. (2) <u>BA</u>: to model local structural information under a more realistic homophily graph, we construct a Barabási-Albert (BA) graph and then plant different shapes to the graph. Then, we compute the Graphlet Distribution Vector [29] for each node, which characterizes the local graph structures and then we apply spectral clustering on this vector to generate the labels. For multiple graphs problem, a varying distribution of numbers of shapes are used to plant each BA graph. Details are in Appendix F.

Real-world datasets and new meta-learning datasets. We use three real world datasets for node classification and two for link prediction to evaluate G-META (Table 1). (1) <u>ogbn-arxiv</u> is a CS citation network, where features are titles, and labels are the subject areas [17]. (2) <u>Tissue-PPI</u> consists of 24 protein-protein interaction networks from different tissues, where features are gene signatures and labels are gene ontology functions [65, 15]. (3) <u>Fold-PPI</u> is a novel dataset, which we constructed for the multiple graph and disjoint label problem. It has 144 tissue PPI networks [65], and the labels are protein structures defined in SCOP database [2]. The features are conjoint triad protein descriptor [34]. We screen fold groups that have more than 9 unique proteins across the networks. It results in 29 unique labels. Like many real-world graphs, in Fold-PPI, the majority of the nodes do not have associated labels. This also shows the importance of graph few-shot learning. (4) <u>FirstMM-DB</u> [26] is the standard 3D point cloud data for link prediction across graphs, which consists of 41 graphs. (5) <u>Tree-of-Life</u> is a new dataset that we constructed based on 1,840 protein interaction networks, each originating from a different species [64]. Since node features are not provided, we use node degrees instead. Details are in Appendix F.

Experimental setup. We follow the standard episode training for few-shot learning [40]. We refer the readers to Section 3.1 for task setups given various graph meta learning problems. Note that in order to simulate the real-world few-shot graph learning settings, we do not use the full set of labels provided in the dataset. Instead, we select a partition of it by constructing a fixed number of meta-tasks. Here, we describe the various parameters used in this study. (1) <u>Node classification.</u> For disjoint label setups, we sample 5 labels for meta-testing, 5 for meta-validation, and use the rest for meta-training. In each task, 2-ways 1-shot learning with 5 gradient update steps in meta-training and 10 gradient update steps in meta-testing is used for synthetic datasets. 3-ways 3-shots learning with 10 gradient update steps in meta-training and 20 gradient update steps in meta-testing are used for real-world datasets disjoint label settings. For multiple graph shared labels setups, 10% (10%) of all graphs are held out for testing (validation). The remaining graphs are used for training. For fold-PPI, we use the average of ten 2-way protein function tasks. (2) <u>Link prediction.</u> 10% of graphs are held out for testing and another 10% for validation. For each graph, the support set consists of 30% edges and the query set 70%. Negative edges are sampled randomly to match the same number of positive edges, and we follow the experiment setting from [59] to construct the training graph. We use 16-shots for each task, *i.e.*, using only 32 node pairs to predict links for an unseen graph. 10 gradient update steps in meta-training and 20 gradient update steps in meta-testing are used. Each experiment is repeated five times to calculate the standard deviation of the results. Hyperparameters selection and a recommended set of them are in Appendix G.

Table 2: Graph meta-learning performance on synthetic datasets. Reported is multi-class classification accuracy (five-fold average) for 1-shot node classification in various graph meta-learning problems. N/A means the method does not apply for the problem. Disjoint label problems use a 2-way setup. In the shared label problem, the cycle graph has 17 labels and the BA graph has 10 labels. The results use 5 gradient update steps in meta-training and 10 gradient update steps in meta-testing. Full table with standard deviations is in Appendix K.

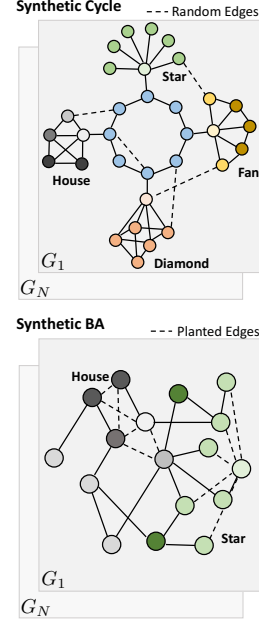| Graph Meta-Learning Problem	Single graph Disjoint labels		Multiple graphs Shared labels		Multiple graphs Disjoint labels	
Prediction Task	Node		Node		Node	
Base graph	Cycle	BA	Cycle	BA	Cycle	BA
G-META (Ours)	0.872	**0.867**	**0.542**	**0.734**	**0.767**	**0.867**
Meta-Graph	N/A	N/A	N/A	N/A	N/A	N/A
Meta-GNN	0.720	0.694	N/A	N/A	N/A	N/A
FS-GIN	0.684	0.749	N/A	N/A	N/A	N/A
FS-SGC	0.574	0.715	N/A	N/A	N/A	N/A
KNN	**0.918**	0.804	0.343	0.710	0.753	0.769
No-Finetune	0.509	0.567	0.059	0.265	0.592	0.577
Finetune	0.679	0.671	0.385	0.517	0.599	0.629
ProtoNet	0.821	0.858	0.282	0.657	0.749	0.866
MAML	0.842	0.848	0.511	0.726	0.653	0.844

Figure 3: Synthetic Cycle and Barabási-Albert datasets.

Table 3: Graph meta-learning performance on real-world datasets. Reported is multi-class classification accuracy (five-fold average) and standard deviation. N/A means the method does not work in the graph meta-learning problem. Note that ProtoNet and MAML can be considered as ablation studies of G-META. Further details on performance are in Appendix I.

Graph Meta-Learning Problem	Single graph Disjoint labels	Multiple graphs Shared labels	Multiple graphs Disjoint labels	Multiple graphs Shared labels	Multiple graphs Shared labels
Prediction Task	Node	Node	Node	Link	Link
Dataset	ogbn-arxiv	Tissue-PPI	Fold-PPI	FirstMM-DB	Tree-of-Life
G-META (Ours)	$0.451_{\pm0.032}$	$0.768_{\pm0.029}$	$0.561_{\pm0.059}$	$0.784_{\pm0.028}$	$0.722_{\pm0.032}$
Meta-Graph	N/A	N/A	N/A	$0.719_{\pm0.020}$	$0.705_{\pm0.004}$
Meta-GNN	$0.273_{\pm0.122}$	N/A	N/A	N/A	N/A
FS-GIN	$0.336_{\pm0.042}$	N/A	N/A	N/A	N/A
FS-SGC	$0.347_{\pm0.005}$	N/A	N/A	N/A	N/A
KNN	$0.392_{\pm0.015}$	$0.619_{\pm0.025}$	$0.433_{\pm0.034}$	$0.603_{\pm0.072}$	$0.649_{\pm0.012}$
No-Finetune	$0.364_{\pm0.014}$	$0.516_{\pm0.006}$	$0.376_{\pm0.017}$	$0.509_{\pm0.006}$	$0.505_{\pm0.001}$
Finetune	$0.359_{\pm0.010}$	$0.521_{\pm0.013}$	$0.370_{\pm0.022}$	$0.511_{\pm0.007}$	$0.504_{\pm0.003}$
ProtoNet	$0.372_{\pm0.017}$	$0.546_{\pm0.025}$	$0.382_{\pm0.031}$	$0.779_{\pm0.020}$	$0.697_{\pm0.010}$
MAML	$0.389_{\pm0.021}$	$0.745_{\pm0.051}$	$0.482_{\pm0.062}$	$0.758_{\pm0.025}$	$0.719_{\pm0.012}$

Nine baseline methods. Meta-Graph [3] injects graph signature in VGAE [19] to do few-shot multi-graph link prediction. Meta-GNN [61] applies MAML [12] to Simple Graph Convolution (SGC) [50]. Few-shot Graph Isomorphism Network (FS-GIN) [52] applies GIN on the entire graph and retrieve the few-shot nodes to propagate loss and enable learning. Similarly, Few-shot SGC (FS-SGC) [50] switches GIN to SGC for GNN encoder. Note that the previous four baselines only work in a few graph meta-learning problems. We also test on different meta-learning models, using the top performing ones in [40]. We operate on subgraph level for them since it allows comparison in all problems. No-Finetune performs training on the support set and use the trained model to classify each query example, using only meta-testing set. KNN [10, 40] first trains a GNN using all data in the meta-training set and it is used as an embedding function. Then, it uses the label of the voted K-closest example in the support set for each query example. Finetune [40] uses the embedding function generated from meta-training set and the models are then finetuned on the meta-testing set. ProtoNet [35] applies prototypical learning on each subgraph embeddings, following the standard few-shot learning setups. MAML [12] switches ProtoNet to MAML as the meta-learner. Note that the baselines ProtoNet and MAML can be considered as an ablation of G-META removing MAML and Prototypical loss respectively. All experiments use the same number of query points per label. We use multi-class accuracy metric.

6.1 Results

Overview of results. Synthetic dataset results are reported in Table 2 and real-world datasets in Table 3. We see that G-META can consistently achieve the best accuracy in almost all tested graph meta-learning problems and on both node classification and link prediction tasks. Notably, G-META achieves a 15.1% relative increase in the single graph disjoint setting, a 16.3% relative increase in the multiple graph disjoint label setting, over the best performing baseline. This performance boost suggests that G-META can learn across graphs and disjoint labels. Besides, this increase also shows G-META obtains good predictive performance, using only a few gradient update steps given a few examples on the target tasks.

G-META captures local structural information. In the synthetic datasets, we see that Meta-GNN, FS-GIN, and FS-SGC, which base on the entire graph, are inferior to subgraph-based methods, such as G-META. This finding demonstrates that subgraph embedding can capture the local structural roles, whereas using the entire graph cannot. It further supports our theoretical motivation. In the single graph disjoint label setting, KNN achieves the best result. This result suggests that the subgraph representation learned from the trained embedding function is sufficient to capture the structural roles, further confirming the usefulness of local subgraph construction.

G-META is highly predictive, general graph meta-learning method. Across meta-learning models, we observe G-META is consistently better than others such as MAML and ProtoNet. MAML and ProtoNet have volatile results across different problems and tasks, whereas G-META is stable. This result confirms our analysis of using the prototypical loss to leverage the label inductive bias and MAML to transfer knowledge across graphs and labels. By comparing G-META to two relevant existing works Meta-GNN and Meta-Graph, we see G-META can work across different problems. In contrast, the previous two methods are restricted by the meta graph learning problems. In real-world datasets, we observe No-Finetune is better than Finetune in many problems (ogbn-arxiv & Fold-PPI). This observation shows that the meta-training datasets bias the meta-testing result, suggesting the importance of meta-learning algorithm to achieve positive transfer in meta graph learning.

Local subgraphs are vital for successful graph few-shot learning. We observe that Meta-GNN, FS-GIN, and FS-SGC, which operate on the entire graph, perform poorly. In contrast, G-META can learn transferable signal, even when label sets are disjoint. Since the problem requires learning using only 3-shots in a large graph of around 160 thousands nodes and 1 million edges (ogbn-arxiv), we posit that operating GNN on the entire graph would not propagate useful information. Hence, the performance increase suggests that local subgraph construction is useful, especially in large graphs. Finally, in the Tree-of-Life link prediction problem, we studied the largest-ever graph meta-learning dataset, spanning 1,840 PPI graphs. This experiment also supports the scalability of G-META, which is enabled by the local subgraphs.

Ablation and parameter studies. We conduct ablations on two important components, optimization-based meta-learning and prototypical loss. Note that baseline ProtoNet and MAML are the ablations of G-META. We find that both components are important to improve predictive performance. To study parameters, we first vary the size of subgraph size h. We find that $h = 2$ has a better performance across datasets than $h = 1$. When $h = 3$, in some dataset, it introduces noise and decreases the performance while in other one, it can improve performance since it includes more information. We suggest to use $h = 2$ since it has the most stable performance. We then vary the number of shots k, and observes a linear trend between k and the predictive performance. The detailed quantitative results on the parameter studies are provided in the Appendix J.

7 Conclusion

We introduce G-META, a scalable and inductive meta-learning method for graphs. Unlike earlier methods, G-META excels at difficult few-shot learning tasks and can tackle a variety of meta-learning problems. The core principle in G-META is to use local subgraphs to identify and transfer useful information across tasks. The local subgraph approach is fundamentally different from prior work that uses entire graphs capturing global graph structure at the loss of finer topological structure. Local subgraphs are theoretically justified and stem from our observation that evidence for a prediction can be found in the local subgraph surrounding the target entity. G-META outperforms nine baselines across seven datasets, including a new and original dataset of 1,840 graphs.

Broader Impact

Graphs represent an incredibly powerful data representation that has proved useful for numerous domains and application areas. At the same time, meta learning is a key area of machine learning research that has already demonstrated great potential for problems, such as few-shot image recognition and neural architecture search.

G-META advances ML research. Our present work is at an exciting yet underexplored intersection of graph ML and meta learning, advancing the state-of-the-art and enabling practical applications of meta learning for large graph datasets. State-of-the-art GNN methods do not work when there are only a handful of labels available. Our work fills in this gap by providing a novel approach to leverage related information such as related graphs or other labels sets to aid the graph few-shot learning. Numerous real-world graphs are only associated with scarce labels of nodes or have a large percentage of missing links. Low-resource constraint (*e.g.*, the sparsity of graphs/scarcity of labels) is a fundamental challenge in real-world graph applications. The goal of graph meta-learning is to solve key graph ML tasks, such as node classification and link prediction, under these constraints.

We envision numerous impactful applications of our work, as already demonstrated in the paper. We provide a brief overview of key applications for graph meta-learning, and G-META in particular.

G-META can expedite scientific discovery. Science is filled with complex systems that are modeled by graphs. The labels are usually obtained through resource-intensive experiments in laboratories. Many experiments, such as those to measure protein-protein interactions (studied in the paper), cannot yet be automated. Because of that, labels (*i.e.*, a variety of molecular properties that need to be discovered for downstream problems like drug discovery [42]) are expensive to obtain and thus are very scarce. G-META can accelerate scientific discovery by learning from related sources and quickly adapting to a new, never-before-seen task of interest given only a handful of examples. For example, G-META can better annotate rare disease pathways by transferring knowledge from the mouse PPI network to the human PPI network [31]. Further, PPI networks of many species are highly incomplete, even for humans, less than 30% of all pairwise combinations of proteins have been tested for interaction so far [23]. Graph meta-learning can help label sparse networks by learning from a handful of other networks that are well-annotated.

G-META can bring economic values. While graphs have helped businesses make better products (*e.g.*, [54, 24]), many graph-structured data resources remain under-utilized because of the sparsity and scarcity of labels. G-META can help tackle this problem. For example, it can improve recommendation for a new set of products that only have a few user behavior records by transferring knowledge from previous product sets in the user products recommendation networks, or help a business expand to new locations by learning from the interconnected data about the current location.

G-META can improve equality. G-META can be used to facilitate the development of world regions by quickly learning from the developed regions. While this has previously been successfully demonstrated on satellite image data [18], we see many new opportunities for graph data, such as identifying what transportation lines (links) should be prioritized to add in a rural region by learning from infrastructure networks of developed regions that were similarly rural before.

Finally, we briefly comment on potential risks of having a powerful graph meta-learning predictor. (1) Negative transfer: Meta-learning models are tricky to train and may vary across datasets and domains. It can lead to negative transfer. (2) Misuse of the methodology: Meta-learning is not universal. It works in specific settings. For example, G-META is applied to few-shot settings. When labels are abundant, it might not yield considerable benefits. (3) Adversarial attacks: There are no studies on how adversarial attacks would affect meta-learning algorithms and graph algorithms in particular. We posit that in the few-shot setting, it may be particularly easy to attack the graph since each labeled example is vital for prediction.

Acknowledgments and Disclosure of Funding

References

[1] Emily Alsentzer, Samuel G Finlayson, Michelle M Li, and Marinka Zitnik. Subgraph neural networks. NeurIPS, 2020.

[2] Antonina Andreeva, Eugene Kulesha, Julian Gough, and Alexey G Murzin. The scop database in 2020: expanded classification of representative family and superfamily domains of known protein structures. Nucleic Acids Research, 48(D1):D376–D382, 2020.

[3] Avishek Joey Bose, Ankit Jain, Piero Molino, and William L Hamilton. Meta-Graph: few shot link prediction via meta learning. arXiv:1912.09867, 2019.

[4] Rich Caruana. Multitask learning. Machine Learning, 28(1):41–75, 1997.

[5] Jatin Chauhan, Deepak Nathani, and Manohar Kaul. Few-shot learning on graphs via super-classes based on graph spectral measures. In ICLR, 2020.

[6] Jie Chen and Yousef Saad. Dense subgraph extraction with application to community detection. IEEE Transactions on knowledge and data engineering, 24(7):1216–1230, 2010.

[7] Mingyang Chen, Wen Zhang, Wei Zhang, Qiang Chen, and Huajun Chen. Meta relational learning for few-shot link prediction in knowledge graphs. In EMNLP-IJCNLP, Hong Kong, China, November 2019.

[8] 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 KDD, pages 257–266, 2019.

[9] Claire Donnat, Marinka Zitnik, David Hallac, and Jure Leskovec. Learning structural node embeddings via diffusion wavelets. In KDD, pages 1320–1329, 2018.

[10] Sahibsingh A Dudani. The distance-weighted k-nearest-neighbor rule. IEEE Transactions on Systems, Man, and Cybernetics, 0(4):325–327, 1976.

[11] Karoline Faust, Pierre Dupont, Jérôme Callut, and Jacques Van Helden. Pathway discovery in metabolic networks by subgraph extraction. Bioinformatics, 26(9):1211–1218, 2010.

[12] Chelsea Finn, Pieter Abbeel, and Sergey Levine. Model-agnostic meta-learning for fast adaptation of deep networks. In ICML, pages 1126–1135, 2017.

[13] Justin Gilmer, Samuel S Schoenholz, Patrick F Riley, Oriol Vinyals, and George E Dahl. Neural message passing for quantum chemistry. In ICML, pages 1263–1272. JMLR. org, 2017.

[14] Alex Graves, Greg Wayne, and Ivo Danihelka. Neural turing machines. arXiv:1410.5401, 2014.

[15] Will Hamilton, Zhitao Ying, and Jure Leskovec. Inductive representation learning on large graphs. In NeurIPS, pages 1024–1034, 2017.

[16] Keith Henderson, Brian Gallagher, Tina Eliassi-Rad, Hanghang Tong, Sugato Basu, Leman Akoglu, Danai Koutra, Christos Faloutsos, and Lei Li. Rolx: structural role extraction & mining in large graphs. In KDD, pages 1231–1239, 2012.

[17] 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. arXiv:2005.00687, 2020.

[18] Neal Jean, Marshall Burke, Michael Xie, W Matthew Davis, David B Lobell, and Stefano Ermon. Combining satellite imagery and machine learning to predict poverty. Science, 353(6301):790–794, 2016.

[19] Thomas N Kipf and Max Welling. Variational graph auto-encoders. Bayesian Deep Learning Workshop at NIPS, 2016.

[20] Thomas N. Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. In ICLR, 2017.

[21] Qimai Li, Zhichao Han, and Xiao-Ming Wu. Deeper insights into graph convolutional networks for semi-supervised learning. In AAAI, 2018.

[22] LU LIU, Tianyi Zhou, Guodong Long, Jing Jiang, and Chengqi Zhang. Learning to propagate for graph meta-learning. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d Alché-Buc, E. Fox, and R. Garnett, editors, NeurIPS, pages 1039–1050, 2019.

[23] Katja Luck, Dae-Kyum Kim, Luke Lambourne, Kerstin Spirohn, Bridget E Begg, Wenting Bian, Ruth Brignall, Tiziana Cafarelli, Francisco J Campos-Laborie, Benoit Charloteaux, et al. A reference map of the human binary protein interactome. Nature, 580(7803):402–408, 2020.

[24] Julian McAuley, Christopher Targett, Qinfeng Shi, and Anton Van Den Hengel. Image-based recommendations on styles and substitutes. In SIGIR, pages 43–52, 2015.

[25] Bryan McCann, Nitish Shirish Keskar, Caiming Xiong, and Richard Socher. The natural language decathlon: Multitask learning as question answering. arXiv:1806.08730, 2018.

[26] Marion Neumann, Plinio Moreno, Laura Antanas, Roman Garnett, and Kristian Kersting. Graph kernels for object category prediction in task-dependent robot grasping. In Online Proceedings of the Eleventh Workshop on Mining and Learning with Graphs, pages 0–6, 2013.

[27] Alex Nichol, Joshua Achiam, and John Schulman. On first-order meta-learning algorithms. arXiv:1803.02999, 2018.

[28] Mathias Niepert, Mohamed Ahmed, and Konstantin Kutzkov. Learning convolutional neural networks for graphs. In ICML, page 2014–2023, 2016.

[29] Nataša Pržulj. Biological network comparison using graphlet degree distribution. Bioinformatics, 23(2):e177–e183, 2007.

[30] Sachin Ravi and Hugo Larochelle. Optimization as a model for few-shot learning. In ICLR, 2017.

[31] Thomas Rolland, Murat Taşan, Benoit Charloteaux, Samuel J Pevzner, Quan Zhong, Nidhi Sahni, Song Yi, Irma Lemmens, Celia Fontanillo, Roberto Mosca, et al. A proteome-scale map of the human interactome network. Cell, 159(5):1212–1226, 2014.

[32] Adam Santoro, Sergey Bartunov, Matthew Botvinick, Daan Wierstra, and Timothy Lillicrap. Meta-learning with memory-augmented neural networks. In ICML, pages 1842–1850, 2016.

[33] Jürgen Schmidhuber, Jieyu Zhao, and Marco Wiering. Shifting inductive bias with success-story algorithm, adaptive Levin search, and incremental self-improvement. Machine Learning, 28(1):105–130, 1997.

[34] Juwen Shen, Jian Zhang, Xiaomin Luo, Weiliang Zhu, Kunqian Yu, Kaixian Chen, Yixue Li, and Hualiang Jiang. Predicting protein–protein interactions based only on sequences information. PNAS, 104(11):4337–4341, 2007.

[35] Jake Snell, Kevin Swersky, and Richard Zemel. Prototypical networks for few-shot learning. In NeurIPS, pages 4077–4087, 2017.

[36] F. Sung, Y. Yang, L. Zhang, T. Xiang, P. H. S. Torr, and T. M. Hospedales. Learning to compare: Relation network for few-shot learning. In CVPR, pages 1199–1208, 2018.

[37] Flood Sung, Yongxin Yang, Li Zhang, Tao Xiang, Philip HS Torr, and Timothy M Hospedales. Learning to compare: Relation network for few-shot learning. In CVPR, 2018.

[38] Komal K Teru, Etienne Denis, and William L Hamilton. Inductive relation prediction by subgraph reasoning. ICML, 2020.

[39] Sebastian Thrun and Lorien Pratt. Learning to learn. Springer Science & Business Media, 2012.

[40] Eleni Triantafillou, Tyler Zhu, Vincent Dumoulin, Pascal Lamblin, Utku Evci, Kelvin Xu, Ross Goroshin, Carles Gelada, Kevin Swersky, Pierre-Antoine Manzagol, et al. Meta-dataset: A dataset of datasets for learning to learn from few examples. ICLR, 2020.

[41] Ruo-Chun Tzeng and Shan-Hung Wu. Distributed, egocentric representations of graphs for detecting critical structures. In Kamalika Chaudhuri and Ruslan Salakhutdinov, editors, ICML, pages 6354–6362, 2019.

[42] Jessica Vamathevan, Dominic Clark, Paul Czodrowski, Ian Dunham, Edgardo Ferran, George Lee, Bin Li, Anant Madabhushi, Parantu Shah, Michaela Spitzer, et al. Applications of machine learning in drug discovery and development. Nature Reviews Drug Discovery, 18(6):463–477, 2019.

[43] Petar Veličković, William Fedus, William L. Hamilton, Pietro Liò, Yoshua Bengio, and R Devon Hjelm. Deep graph infomax. In ICLR, 2019.

[44] Ricardo Vilalta and Youssef Drissi. A perspective view and survey of meta-learning. Artificial Intelligence Review, 18(2):77–95, 2002.

[45] Oriol Vinyals, Charles Blundell, Timothy Lillicrap, koray kavukcuoglu, and Daan Wierstra. Matching networks for one shot learning. In D. D. Lee, M. Sugiyama, U. V. Luxburg, I. Guyon, and R. Garnett, editors, NeurIPS, pages 3630–3638, 2016.

[46] Hongwei Wang and Jure Leskovec. Unifying graph convolutional neural networks and label propagation. arXiv:2002.06755, 2020.

[47] Xiao Wang, Houye Ji, Chuan Shi, Bai Wang, Yanfang Ye, Peng Cui, and Philip S Yu. Heterogeneous graph attention network. In The World Wide Web Conference, pages 2022–2032, 2019.

[48] Jason Weston, Sumit Chopra, and Antoine Bordes. Memory networks. arXiv:1410.3916, 2014.

[49] Mitchell Wortsman, Kiana Ehsani, Mohammad Rastegari, Ali Farhadi, and Roozbeh Mottaghi. Learning to learn how to learn: Self-adaptive visual navigation using meta-learning. In CVPR, pages 6750–6759, 2019.

[50] Felix Wu, Amauri Souza, Tianyi Zhang, Christopher Fifty, Tao Yu, and Kilian Weinberger. Simplifying graph convolutional networks. In ICML, pages 6861–6871, 2019.

[51] Wenhan Xiong, Mo Yu, Shiyu Chang, Xiaoxiao Guo, and William Yang Wang. One-shot relational learning for knowledge graphs. In EMNLP, Brussels, Belgium, 2018.

[52] Keyulu Xu, Weihua Hu, Jure Leskovec, and Stefanie Jegelka. How powerful are graph neural networks? In ICLR, 2019.

[53] Huaxiu Yao, Chuxu Zhang, Ying Wei, Meng Jiang, Suhang Wang, Junzhou Huang, Nitesh V Chawla, and Zhenhui Li. Graph few-shot learning via knowledge transfer. AAAI, 2020.

[54] Rex Ying, Ruining He, Kaifeng Chen, Pong Eksombatchai, William L Hamilton, and Jure Leskovec. Graph convolutional neural networks for web-scale recommender systems. In KDD, 2018.

[55] Rex Ying, Zhaoyu Lou, Jiaxuan You, Chengtao Wen, Arquimedes Canedo, and Jure Leskovec. Neural subgraph matching. arXiv:2007.03092, 2020.

[56] Zhitao Ying, Dylan Bourgeois, Jiaxuan You, Marinka Zitnik, and Jure Leskovec. Gnnexplainer: Generating explanations for graph neural networks. In NeurIPS, pages 9240–9251, 2019.

[57] Hanqing Zeng, Hongkuan Zhou, Ajitesh Srivastava, Rajgopal Kannan, and Viktor Prasanna. Graphsaint: Graph sampling based inductive learning method. In ICLR, 2020.

[58] Muhan Zhang and Yixin Chen. Weisfeiler-lehman neural machine for link prediction. In KDD, pages 575–583, 2017.

[59] Muhan Zhang and Yixin Chen. Link prediction based on graph neural networks. In NeurIPS, pages 5165–5175, 2018.

[60] Fan Zhou, Chengtai Cao, Goce Trajcevski, Kunpeng Zhang, Ting Zhong, and Ji Geng. Fast network alignment via graph meta-learning. In IEEE INFOCOM 2020-IEEE Conference on Computer Communications, pages 686–695. IEEE, 2020.

[61] Fan Zhou, Chengtai Cao, Kunpeng Zhang, Goce Trajcevski, Ting Zhong, and Ji Geng. Meta-GNN: On few-shot node classification in graph meta-learning. In CIKM, pages 2357–2360, 2019.

[62] Xiaojin Zhu, Zoubin Ghahramani, and John D Lafferty. Semi-supervised learning using gaussian fields and harmonic functions. In ICML, pages 912–919, 2003.

[63] Marinka Zitnik, Monica Agrawal, and Jure Leskovec. Modeling polypharmacy side effects with graph convolutional networks. Bioinformatics, 34(13):i457–i466, 2018.

[64] Marinka Zitnik, Marcus W Feldman, Jure Leskovec, et al. Evolution of resilience in protein interactomes across the tree of life. PNAS, 116(10):4426–4433, 2019.

[65] Marinka Zitnik and Jure Leskovec. Predicting multicellular function through multi-layer tissue networks. Bioinformatics, 33(14):i190–i198, 2017.