BONSAI: GRADIENT-FREE GRAPH DISTILLATION FOR NODE CLASSIFICATION

Anonymous authors
Paper under double-blind review

ABSTRACT

Graph distillation has emerged as a promising avenue to enable scalable training of GNNS by compressing the training dataset while preserving essential graph characteristics. Our study uncovers significant shortcomings in current graph distillation techniques. First, the majority of the algorithms paradoxically require training on the full dataset to perform distillation. Second, due to their gradient-emulating approach, these methods require fresh distillation for any change in hyper-parameters or GNN architecture, limiting their flexibility and reusability. To address these challenges, we present BONSAI, a novel graph distillation method empowered by the observation that *computation trees* form the fundamental processing units of message-passing GNNS. BONSAI distills datasets by encoding a careful selection of *exemplar* trees that maximize the representation of all computation trees in the training set. This unique approach imparts BONSAI as the first linear-time, model-agnostic graph distillation algorithm for node classification that outperforms existing baselines across 6 real-world datasets on accuracy, while being 22 times faster on average. BONSAI is grounded in rigorous mathematical guarantees on the adopted approximation strategies making it robust to GNN architectures, datasets, and parameters.

1 INTRODUCTION AND RELATED WORKS

Graph Neural Networks (GNNS) have shown remarkable success in various predictive tasks on graph data (Veličković et al., 2018; Kipf & Welling, 2017; Hamilton et al., 2017). However, real world graphs often contain millions of nodes and edges making the training pipeline slow and computationally demanding (Hu et al., 2021). This limitation hinders their adoption in resource-constrained environments (Miao et al., 2021) and applications dealing with massive datasets (Hu et al., 2021). *Graph distillation* (also called condensation) has emerged as a promising way to bypass this bottleneck (Jin et al., 2021; Liu et al., 2024a;b; Zheng et al., 2023). The objective in graph distillation is to synthesize a significantly smaller distilled data set that retains the essential information of the original data. By training GNNS on these distilled datasets, we can achieve comparable performance while reducing computational overhead and storage costs. This makes GNNS more accessible and practical for a wider range of applications, including those with limited computational resources and large-scale datasets. In this work, we study the problem of graph distillation for node classification.

1.1 EXISTING WORKS AND THEIR LIMITATIONS

Table 1 presents existing graph distillation algorithms proposed for node classification and their characterization across various dimensions. We omit listing DOSCOND (Jin et al., 2022), MIRAGE (Gupta et al., 2024), and KIDD (Xu et al., 2023) in Table 1 since they are designed for graph classification, whereas we focus on node classification. While both MIRAGE and the proposed algorithm BONSAI share the paradigm of distillation through computation trees, the algorithms are entirely different. Due to relying on tree isomorphisms, MIRAGE is limited to graphs where nodes are annotated with a single discrete label. Hence, it does not work on general purpose graphs where nodes are characterized through feature vectors. For a more detailed discussion on this limitation and how BONSAI overcomes them, we refer to App. A.1.

Table 1 presents the strengths and limitations of existing distillation methods for node classification.
• **Full GNN training is a pre-requisite:** The fundamental requirement of data distillation is that it should require less computational resources and time than training on the full dataset. However, this basic premise is violated by majority of the techniques (See Table 1) since they adopt a design where training the target GNN on the full trainset is a prerequisite to distillation. These algorithms adopt a

Table 1: Characterization of existing graph distillation algorithms. Cells shaded in Red indicate the presence of an undesirable property, while Green represents their absence.

Algorithm	**Requires training Gnn on full-dataset**	**Distills to a fully-connected graph**[1]	**Model-specific distillation**
GCond (Jin et al., 2021)	✓	✓	✓
Sgdd (Yang et al., 2023)	✓	✓	✓
Sfgc (Zheng et al., 2023)	✓	✓	✓
GC-Sntk (Wang et al., 2024)	✓	✗	✗
Exgc (Fang et al., 2024)	✓	✓	✓
Geom (Zhang et al., 2024)	✓	✗	✓
Gdem (Liu et al., 2024a)	✗	✓	✗
Gcsr (Liu et al., 2024b)	✓	✓	✓
Bonsai	✗	✗	✗

gradient-dependent optimization framework. They first train a Gnn on the full trainset and extract gradients of model parameters over epochs (Sfgc and Gcsr require training on the full dataset up to 200 and 100 times, respectively). The distillation process is formulated as an optimization problem to create a distilled dataset that replicates the gradient trajectory observed in the original training set. This need for full dataset training for distillation contradicts its fundamental premise, which we address in our work.

• **Distilling to a fully-connected graph**: In a message-passing Gnn, the computation cost of each forward pass is $\mathcal{O}(|\mathcal{E}|)$, where $\mathcal{E}$ denotes the set of edges. Consequently, the computational effectiveness of graph distillation is primarily determined by the reduction in edge count between the original and distilled graphs, rather than node count alone. However, current graph distillation algorithms (see Table 1) quantify the condensation ratio based on the node count. Specifically, given a compression ratio r_n, it synthesizes a weighted, fully-connected dense adjacency matrix for the distilled graph of the size $\frac{|\mathcal{V}|}{r_n} \times \frac{|\mathcal{V}|}{r_n}$, where $\mathcal{V}$ denotes the node set. Some of these algorithms sparsify by removing edges with edge weights below a certain threshold. This threshold is chosen by studying the drop in accuracy at various sparsity levels and choosing a point providing good accuracy-efficiency trade-off. Consequently, the sparsification process itself requires training on the fully connected initial distilled graph. This design, which is inconsistent with the computation structure of Gnns, can lead to a distilled graph with small reduction in edge count and, in some cases, may even result in a graph with more edges than the original dataset (See Table 2).

Table 2: We present the number of edges in the distilled graphs produced by GCond (Jin et al., 2021) and Gdem (Liu et al., 2024a) and compare them to the full dataset. Red cells indicate cases where the distilled graph have more edges than the full dataset. The indicated node ratios are taken from the values used in GCond. For GCond, we report the number of edges after sparsification as reported in their github repository at https://github.com/ChandlerBang/GCond/tree/main/saved_ours .

Dataset	**Distilled**		**Full Dataset**
(r_n or node ratio)	GCond	GDEM	#edges
Cora (5.2%)	15,074	19,600	10,556
Citeseer (3.6%)	10,996	14,400	9,104
Pubmed (0.3%)	3,557	3,600	88,648
Flickr (1%)	23,556	795,664	899,756
Ogbn-arxiv (0.5%)	15,110	715,716	2,315,598
Reddit (0.2%)	5,756	216,225	23,213,838

• **Model-specific distillation:** Gradients of model weights are influenced by the specific Gnn architecture and hyper-parameters (e.g., number of layers, hidden dimensions, etc.). Consequently, any architectural change, such as switching from a Gcn (Kipf & Welling, 2017) to a Gat (Veličković et al., 2018), or hyper-parameter adjustments, necessitates a new round of distillation.

1.2 CONTRIBUTIONS

To address the limitations outlined in Table 1, we present Bonsai[2].

- **Gradient-free distillation:** Instead of replicating the gradient trajectory, Bonsai emulates the distribution of input data processed by message-passing Gnns. By shifting the computation task to the *pre-learning phase*, Bonsai achieves independence from hyper-parameters and model architectures as long as it adheres to a message-passing Gnn framework like Gat (Veličković et al., 2018), Gcn (Kipf & Welling, 2017), GraphSage (Hamilton et al., 2017), Gin (Xu et al., 2019),

---

[1] Some algorithms sparsify the fully-connected graph based on edge weights. But this sparsification process requires training on the fully connected graph itself to identify the pruning threshold.

[2] Inspired by the art of Bonsai, which transforms large trees into miniature forms while preserving their essence, our graph distillation algorithm gracefully prunes redundant computation trees, creating a distilled graph that is significantly smaller yet maintains comparable performance.

etc. Moreover, this addresses a critical limitation of existing graph distillation algorithms that necessitates training on the entire training dataset.

- **Novel algorithm design:** BONSAI is empowered by the observation that any message-passing GNN decomposes a graph of n nodes into n rooted *computation trees*. Furthermore, topologically similar computation trees generate similar embeddings regardless of the GNN being used (Xu et al., 2019; Togninalli et al., 2019). BONSAI exploits this observation to identify a small subset of diverse computation trees, called *exemplars*, that are located in dense regions and thereby representative of the full set. Hence, the induced subgraph spanned by the exemplars forms an effective distilled set.
- **Empirical evaluation:** We perform rigorous benchmarking incorporating state-of-the-art graph distillation algorithms on 6 real-world datasets containing up to hundreds of millions of edges. Our analysis establishes that BONSAI *(1)* achieves higher prediction accuracy, *(2)* produces at least 7-times faster distillation times despite being CPU-bound in contrast to GPU-bound distillation of baselines, and *(3)* exhibits superior robustness to GNN architectures and datasets.

2 PROBLEM FORMULATION AND PRELIMINARIES

Definition 1 (Graph)**.** *$\mathcal{G} = (\mathcal{V}, \mathcal{E}, \boldsymbol{X})$ denotes a graph over a finite, non-empty node set $\mathcal{V}$ and edge set $\mathcal{E} = \{(u, v) \mid u, v \in \mathcal{V}\}$. $\boldsymbol{X} \in \mathbb{R}^{|\mathcal{V}| \times |F|}$ denotes node attributes encoded using F-dimensional feature vectors. We denote the attributes of node v as $\mathbf{x}_v$.*

Two graphs are identical if they are *isomorphic* to each other.

Definition 2 (Graph Isomorphism)**.** *Graph $\mathcal{G}_1$ is isomorphic to graph $\mathcal{G}_2$ if there exists a bijective mapping between their node sets that preserves both edge connectivity and node features. Specifically, $\mathcal{G}_1$ is isomorphic to $\mathcal{G}_2 \iff \exists f : \mathcal{V}_1 \rightarrow \mathcal{V}_2$ such that: (1) f is a bijection, (2) $\mathbf{x}_v = \mathbf{x}_{f(v)}$, where $v \in \mathcal{V}_1, f(v) \in \mathcal{V}_2$ and (3) $(u, v) \in \mathcal{E}_1$ if and only if $(f(u), f(v)) \in \mathcal{E}_2$.*

The problem of graph distillation for node classification is defined as follows.

Problem 1 (Graph Distillation)**.** *Given train and validation graphs, $\mathcal{G}_{tr}$ and $\mathcal{G}_{val}$, respectively, and a memory budget b in bytes, synthesize a graph $\mathcal{G}_s$ from $\mathcal{G}_{tr}$ within the budget, while minimizing the error gap between $\mathcal{G}_s$ and $\mathcal{G}_{tr}$ on the validation set, i.e., minimize $\{|\epsilon_{\mathcal{G}_s} - \epsilon_{\mathcal{G}_{tr}}|\}$. $\epsilon_{\mathcal{G}}$ represents the node classification error on the validation set when trained on graph $\mathcal{G}$.*

2.1 COMPUTATION STRUCTURE OF GNNS

GNNS operate through an iterative process of information exchange between nodes. Let $\mathbf{x}_v \in \mathbb{R}^{|F|}$ represent the initial feature vector of node $v \in \mathcal{V}$. The propagation mechanism proceeds as follows:

Initialization: Set $\mathbf{h}_v^0 = \mathbf{x}_v, \forall v \in \mathcal{V}$.

Message creation: In layer ℓ, *collect* and *aggregate* messages for each neighbor.

$$\mathbf{m}_v^{\ell}(u) = \textsc{Message}^{\ell}(\mathbf{h}_u^{\ell-1}, \mathbf{h}_v^{\ell-1}), \quad \forall u \in \mathcal{N}_v = \{u \mid (u, v) \in \mathcal{E}\}$$

$$\overline{\mathbf{m}}_v^{\ell} = \textsc{Aggregate}^{\ell}(\{\!\{\mathbf{m}_v^{\ell}(u) : u \in \mathcal{N}_v\}\!\})$$

Update embedding: $\mathbf{h}_v^{\ell} = \textsc{Update}^{\ell}(\mathbf{h}_v^{\ell-1}, \overline{\mathbf{m}}_v^{\ell})$

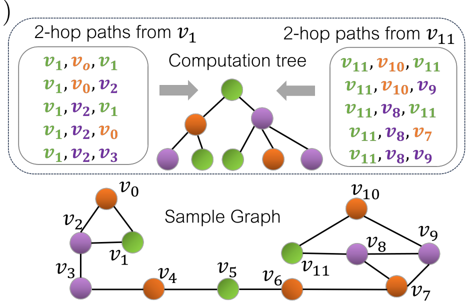


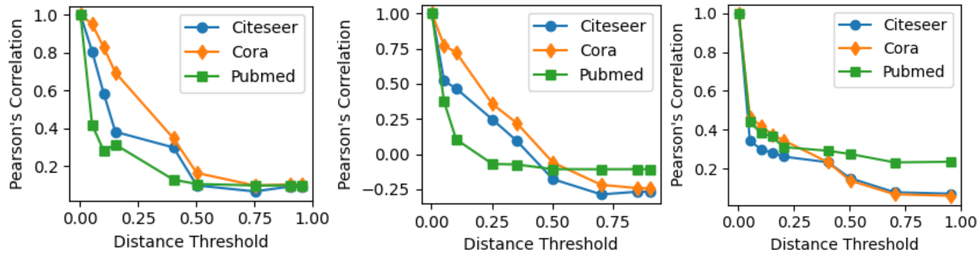
Figure 1: The figure depicts the construction of computation trees for nodes v_1 and v_{11} in the sample graph, at depth $L = 2$. Node colors indicate their labels. Despite being distant from each other in the graph and embedded in non-isomorphic L-hop neighborhoods, v_1 and v_{11} have isomorphic computation trees.

Here, $\textsc{Message}^{\ell}$, $\textsc{Aggregate}^{\ell}$, and $\textsc{Update}^{\ell}$ may be predefined operations (e.g., mean pooling) or learnable neural networks. $\{\!\{\cdot\}\!\}$ denotes a multiset. This process repeats for L layers, yielding the final node representations $\mathbf{h}_v^L$.

A *computation tree* describes how information propagates through the graph during the neural network's operation (Xu et al., 2019).

Definition 3 (Computation Tree)**.** *Given a graph $\mathcal{G} = (\mathcal{V}, \mathcal{E}, \boldsymbol{X})$, a node $v \in \mathcal{V}$ and the number of layers L in the GNN, the computation tree T_v^L rooted at v is constructed as follows:*

- *Enumerate all paths of length L (including those with repeated vertices) starting from v.*
- *Merge these paths into a tree structure where:*



(a) GCN (b) GIN (c) GAT

Figure 2: Pearson correlations between the L_2 distances of node pairs in the GNN embedding space and the unsupervised embeddings derived from the WL-kernel, computed for pairs with a specific distance threshold in the WL-space (x-axis).

1. *The root is always v, and*
2. *Two nodes from different paths are merged if:* ***(i)*** *They are at the same depth in their respective paths, and* ***(ii)*** *All their ancestors in the paths have already been merged.*

Fig. 1 illustrates the idea of a computation tree with an example.

Properties of computation trees: We now highlight two key properties of computation trees, formally established in Xu et al. (2019), that form the core of our algorithm design. These properties hold regardless of the underlying message-passing GNN (GCN, GAT, GRAPHSAGE, GIN, etc.).

Property 1 (Sufficiency)**.** *In an L-layered GNN, the computation tree $\mathcal{T}_v^L$ is sufficient to compute node embedding $\mathbf{h}_v^L$, $\forall v \in \mathcal{V}$. Hence, given a graph $\mathcal{G} = (\mathcal{V}, \mathcal{E}, \boldsymbol{X})$, we may treat it as a (multi)set of computation trees $\mathbb{T} = \{\mathcal{T}_v^L \mid \forall v \in \mathcal{V}\}$. Clearly, $|\mathbb{T}| = |\mathcal{V}|$.*

Property 2 (Equivalence)**.** *If $\mathcal{T}_v^L$ is isomorphic to $\mathcal{T}_u^L$, then $\mathbf{h}_v^L = \mathbf{h}_u^L$. This follows from the property that the expressive power of a message-passing GNN is upper bounded by the Weisfeiler-Lehman test (1-WL) (Xu et al., 2019). This implies that if the L-hop neighborhoods of two nodes are indistinguishable by 1-WL, then their representations will be the same. The 1-WL test cannot differentiate between isomorphic computation trees (Shervashidze et al., 2011).*

Motivated with the above observations, we explore a relaxation of the Equivalence property: *do nodes rooted at topologically similar computation trees generate similar node embeddings?* To explore this hypothesis, we need a method to quantify the distance between computation trees. Given that the Equivalence property is derived from the 1-WL test, the *Weisfeiler-Lehman kernel* (Togninalli et al., 2019) presents itself as the natural choice for this distance metric.

Definition 4 (Weisfeiler-Lehman (WL) Kernel (Togninalli et al., 2019))**.** *Given graph $\mathcal{G} = (\mathcal{V}, \mathcal{E}, \boldsymbol{X})$, WL-kernel constructs unsupervised embeddings over each node in the graph via a message-passing aggregation. Like in GNNs, the initial embedding $\mathbf{a}_v^0$ in layer 0 is initialized to $\mathbf{x}_v$. Subsequently, the embeddings in any layer ℓ is defined as:*

$$\mathbf{a}^{\ell}(v) = \frac{1}{2}\left(\mathbf{a}_v^{\ell-1} + \frac{1}{\deg(v)} \sum_{u \in \mathcal{N}(v)} w((v,u)) \cdot \mathbf{a}_u^{\ell-1}\right) \tag{1}$$

Here, $\mathcal{N}(v)$ denotes the neighbors of v, $\deg(v) = |\mathcal{N}(v)|$ and $w((v,u)) = 1$ for unweighted graphs.

By emulating the same message-passing framework employed by GNNs, the unsupervised embedding $\mathbf{a}_v^L$ jointly encapsulates the topology and node attributes within the computation tree $\mathcal{T}_v^L$. Hence, the distance between two computation trees $\mathcal{T}_v^L$ and $\mathcal{T}_u^L$ is defined to be:

$$d\left(\mathcal{T}_v^L, \mathcal{T}_v^L\right) = \left\|\mathbf{a}_v^L - \mathbf{a}_u^L\right\|_2 \tag{2}$$

While we use the L_2 norm, one may use other distance functions over vectors as well. The following corollary follows from the Eq. 2.

Corollary 1. *If $\mathcal{T}_v^L$ is isomorphic to $\mathcal{T}_u^L$, then $\mathbf{a}_v^L = \mathbf{a}_u^L$.*

Hypothesis 1 (Relaxed Equivalence)**.** *If $\mathcal{T}_v^L \sim \mathcal{T}_u^L$, then $\mathbf{a}_v^L \sim \mathbf{a}_u^L$, and, therefore $\mathbf{h}_v^L \sim \mathbf{h}_u^L$ irrespective of the specific message-passing architecture employed.*

To test our hypothesis, in Fig. 2, we examine all node pairs within a specified distance threshold based on the Weisfeiler-Lehman (WL) kernel. We then analyze the correlation between their unsupervised

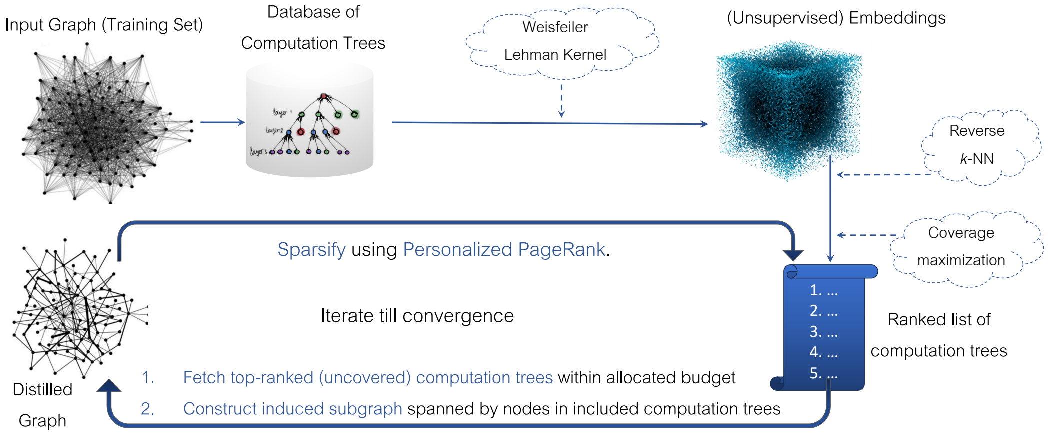

Input Graph (Training Set)
Database of Computation Trees
Weisfeiler Lehman Kernel
(Unsupervised) Embeddings
Reverse k-NN
Coverage maximization
Sparsify using Personalized PageRank.
Iterate till convergence
Distilled Graph
1. ...
2. ...
3. ...
4. ...
5. ...
Ranked list of computation trees
1. Fetch top-ranked (uncovered) computation trees within allocated budget
2. Construct induced subgraph spanned by nodes in included computation trees

Figure 3: Pipeline of the proposed algorithm for graph distillation

WL-distance and the L_2 distance between their GNN embeddings (GCN with 2 layers, trained with cross-entropy loss). The results reveal a compelling trend: as we decrease the distance threshold, we observe a strengthening correlation between WL-distance and GNN embedding distance. This pattern strongly supports our hypothesis, indicating that computation trees that are proximate in the WL-space also exhibit proximity in the GNN embedding space.

3 Bonsai: Proposed Methodology

Fig. 3 presents the pipeline of Bonsai, which is founded on the following logical progression:

1. Similar computation trees produce similar GNN embeddings (Hypotheses 1).
2. Similar GNN embeddings generate comparable outputs, resulting in similar impacts on the loss function and, consequently, on the gradients Table 3 presents empirical data supporting this relationship, showing statistically significant correlations ($p < 0.05$) between WL-embedding similarities and training gradients.

Table 3: Correlation between WL-embedding similarities and training gradients.

Dataset	Correlation	p-value
Cora	0.74	≈ 0
Citeseer	0.83	≈ 0
Pubmed	0.38	0.02
Reddit	0.42	≈ 0

Building on this reasoning, Bonsai aims to identify a set of b *exemplar* computation trees that optimally represent the full training set[1]. These exemplars are selected based on two critical criteria:

- **Representativeness:** Each exemplar should be similar to a large number of other computation trees from the input training set, ensuring it captures common structural patterns. We quantify the representative power of a computation tree using the idea of *reverse k nearest neighbors* (§ 3.1). Specifically, if tree T_1 is among the k-NN of tree T_2 for a small k, this indicates these two trees are similar in WL-embedding. Hence, we seek to include those trees in the distilled set that reside in the k-NN of lots of other trees. Consequently, if these trees are selected, their GNN embeddings are also likely similar to their k-NN neighbors from the WL space. As a result, they can effectively approximate the GNN embeddings of the filtered-out nodes. Since similar GNN embeddings lead to similar gradients (Table 3), we minimize the information lost from nodes that are filtered out.
- **Diversity:** The set of exemplars should be diverse, maximizing the coverage of different structural patterns present in the original graph (§ 3.2). To achieve this objective, we develop a greedy tree selection algorithm (Alg. 1), which begins with the reverse k-NN set and iteratively selects the computation tree that appears in the k-NN sets of the maximum number of currently uncovered trees. This approach ensures two key properties:
    1. It systematically captures trees that are centrally located within the graph's embedding space.
    2. It progressively selects trees that provide maximum coverage of the remaining, yet-unselected computation trees.

By prioritizing trees with the highest marginal impact on uncovered trees, our method naturally creates a diverse subset that comprehensively represents the original graph's structural and computational characteristics. The induced subgraph spanned by these exemplar trees forms the initial

---

[1] We overload the notation for the budget by denoting the number of exemplar trees that fit within the input budget constraint (Recall Problem. 1) as b.

distilled set. This initial version undergoes further refinement through an iterative process of *edge sparsification* and *enrichment* to reduce $O(|\mathcal{E}|)$ complexity of GNNS.

3.1 QUANTIFYING REPRESENTATIVENESS THROUGH REVERSE k-NN

Utilizing Sufficiency (Property 1), we decompose the input graph $\mathcal{G} = (\mathcal{V}, \mathcal{E}, \boldsymbol{X})$ into a set of $|\mathcal{V}|$ computation trees $\mathbb{T}$ and then embed them into a feature space using WL-kernel (See Fig. 3). We now identify representative exemplars by analyzing this space.

The k nearest neighbors of a computation tree $\mathcal{T}_v^L \in \mathbb{T}$, denoted as k-NN($\mathcal{T}_v^L$), are its k nearest computations trees from $\mathbb{T}$ in the WL-kernel space (Def. 4).

Definition 5 (Reverse k-NN and Representative Power)**.** *The reverse k-NN of $\mathcal{T}_v^L$ denotes the set of trees that contains $\mathcal{T}_v^L$ in their k-NNs. Formally, Rev-k-NN($\mathcal{T}_v^L$) $= \left\{\mathcal{T}_u^L \mid \mathcal{T}_v^L \in k\text{-}NN\left(\mathcal{T}_u^L\right)\right\}$. The representative power of $\mathcal{T}_v^L$ is:*

$$\Pi\left(\mathcal{T}_v^L\right) = \frac{\left|Rev\text{-}k\text{-}NN(\mathcal{T}_v^L)\right|}{|\mathbb{T}|} \tag{3}$$

If $\mathcal{T}_v^L \in k$-NN($\mathcal{T}_u^L$) for a small k, it suggests that $\mathbf{h}_v^L \sim \mathbf{h}_u^L$ (Hypothesis 1). Consequently, a high $\Pi\left(\mathcal{T}_v^L\right)$ indicates that $\mathcal{T}_v^L$ frequently appears in the k-NN lists of many other trees and is therefore positioned in a dense region capturing shared characteristics across many trees. Hence, it serves as a strong candidate of being an exemplar.

3.1.1 SAMPLING FOR SCALABLE COMPUTATION OF REVERSE k-NN

Computing k-NN for each tree consumes $\mathcal{O}(n \log k) \approx \mathcal{O}(n)$ time (since $k \ll n$), where $n = |\mathcal{V}|$. Since $|\mathbb{T}| = n$, computing k-NN for all trees consumes $\mathcal{O}(n^2)$ time. Hence, computing reverse k-NN for all trees consumes $\mathcal{O}(n^2)$ time as well, which may be prohibitively expensive for graphs containing millions of nodes. To address this challenge, we employ a *sampling* technique that offers a provable approximation guarantee on accuracy. This is achieved as follows. Let us sample $z \ll n$ trees uniformly at random from $\mathbb{T}$, which we denote as $\mathbb{S}$. Now, we compute the k-NN of only trees in $\mathbb{S}$, which incurs $\mathcal{O}(zn)$ computation cost. We next approximate reverse k-NN of all trees in $\mathbb{T}$ based only on $\mathbb{S}$. Specifically, $\widetilde{\text{Rev-}k\text{-NN}}\left(\mathcal{T}_v^L\right) = \left\{\mathcal{T}_u^L \in \mathbb{S} \mid \mathcal{T}_v^L \in k\text{-NN}\left(\mathcal{T}_u^L\right)\right\}$. The approximated representative power is therefore:

$$\widetilde{\Pi}\left(\mathcal{T}_v^L\right) = \frac{\left|\widetilde{\text{Rev-}k\text{-NN}}(\mathcal{T}_v^L)\right|}{|\mathbb{S}|} \tag{4}$$

The sample size z balances the trade-off between accuracy and computational efficiency. By applying *Chernoff bounds*, we demonstrate in Lemma 1 that z can be precisely determined to maintain the error within a specified error threshold θ, with a confidence level of $1 - \delta$.

Lemma 1. *Given a desirable error upper bound θ and a confidence interval of $\mathbf{1} - \delta$, if we sample at least $z = \frac{\ln\left(\frac{2}{\delta}\right)(2+\theta)}{\theta^2}$ trees in $\mathbb{S}$, then for any $\mathcal{T}_v^L \in \mathbb{T}$:*

$$P(\left|\widetilde{\Pi}(\mathcal{T}_v^L) - \Pi\left(\mathcal{T}_v^L\right)\right| \leq \theta) \geq 1 - \delta \tag{5}$$

PROOF. For the formal proof, see App. A.2. Lemma 1 induces the following positive implications.

- The number of samples needed is independent of the size of $\mathbb{T}$.
- Because z grows logarithmically with $ln(\frac{2}{\delta})$, even a small sample size provides high confidence. Hence, the computation cost of Rev-k-NN reduces to $\mathcal{O}(n)$ since $z \ll n$.

3.2 COVERAGE MAXIMIZATION

We aim to find the set of *exemplar* computation trees with the maximum representative power.

Definition 6 (The exemplars)**.** *Let the representative power of a set of trees $\mathbb{A}$ be denoted by:*

$$\Pi\left(\mathbb{A}\right) = \left|\bigcup_{\forall \mathcal{T}_v^L \in \mathbb{A}} Rev\text{-}k\text{-}NN\left(\mathcal{T}_v^L\right)\right| \Big/ |\mathbb{T}| \tag{6}$$

Then, given the set of computation trees $\mathbb{T}$ and the budget b, we seek to identify the subset of computations trees, termed exemplars, $\mathbb{A}^*$, by maximizing the following objective:

$$\mathbb{A}^* = \max_{\forall \mathbb{A} \subseteq \mathbb{T}, |\mathbb{A}| = b} \Pi(\mathbb{A}) \tag{7}$$

For brevity of discourse, we proceed assuming the true reverse k-NN set.

Algorithm 1 The greedy approach

```
Require: Graph $\mathcal{G}$, budget $b$, $Rev{-}\text{k}{-}NN\left(\mathcal{T}_v^L\right)$
Ensure: solution set $\mathbb{A}$, $|\mathbb{A}| = b$
1: $\mathbb{A} \leftarrow \emptyset$
2: while $size(\mathbb{A}) \leq b$ (within budget) do
3:     $\mathcal{T}_{v^*}^L \leftarrow \arg\max_{\mathcal{T}_v^L \in \mathbb{T} \setminus \mathbb{A}} \{\Pi(\mathbb{A} \cup \{\mathcal{T}_v^L\}) - \Pi(\mathbb{A})\}$
4:     $\mathbb{A} \leftarrow \mathbb{A} \cup \{\mathcal{T}_{v^*}^L\}$
5: Return $\mathbb{A}$
```

Theorem 1. *Maximizing the representative power of exemplars (Eq. 7) is NP-hard.*

PROOF. App. A.3 presents the formal proof by reducing to the *Set Cover problem* (Cormen et al., 2009). □

Fortunately, Eq. 6 is *monotone* and *submodular*, which allows the greedy hill-climbing algorithm (Alg. 1) to approximate Eq. 7 within a provable error bound.

Theorem 2. *The exemplars,* $\mathbb{A}_{greedy}$*, selected by Alg. 1 provides an* $1 - 1/e$ *approximation, i.e.,* $\Pi(\mathbb{A}_{greedy}) \geq \left(1 - \frac{1}{e}\right) \Pi(\mathbb{A}^*)$,

PROOF. App. A.4 presents the proofs of monotonicity and submodularity of $\Pi(\mathbb{A})$. For monotonic and submodular functions, greedy selection provides $1 - 1/e$ approximation (Feige, 1998). □.

Alg. 1 begins with the reverse k-NN set of each computation tree as input. It then iteratively selects trees based on their *marginal cardinality* - specifically, choosing the tree that appears in the k-NN sets of the largest number of yet-uncovered trees (lines 3-4). A tree is considered uncovered if none of its k-nearest neighbors have been selected for the distilled set. This focus on marginal contribution naturally promotes diversity. Consider two similar trees, T_1 and T_2, both with high reverse k-NN cardinality. Due to the transitivity of distance functions, these trees likely share many of the same neighbors in their reverse k-NN sets. Consequently, if T_1 is selected, T_2's marginal cardinality significantly decreases despite its high initial reverse k-NN cardinality, preventing redundant selection of similar trees.

We further enhance the efficiency of Alg. 1 by exploiting the property of monotonically decreasing *marginal gains* in submodular optimization (Leskovec et al., 2007).

Initial distilled graph: The initial distilled graph $\mathcal{G}_s = (\mathcal{V}_s, \mathcal{E}_s, \boldsymbol{X}_s)$ is formed by extracting the induced subgraph spanned by the exemplar computation trees, i.e., $\mathcal{V}_s = \left\{u \mid \exists \mathcal{T}_v^L \in \mathbb{A}_{greedy},\ u \in \mathcal{N}_L(v)\right\}$, $\mathcal{E}_s = \{(u, v) \mid u \in \mathcal{V}_s, v \in \mathcal{V}_s\}$ and $\boldsymbol{X}_s = \{\mathbf{x}_v) \mid v \in \mathcal{V}_s\}$. Set $\mathcal{N}_L(v)$ contains nodes within L hops from v.

Sparsification and Enrichment: In the final step, we seek to sparsify the graph induced by the exemplar trees. Furthermore, in the additional space created due to sparsfication, we include more exemplar trees, resulting in further magnification of the representative power. Further details of our implementation is provided in App. A.5.

3.3 PROPERTIES OF BONSAI

Complexity analysis: The computation complexity of BONSAI is $\mathcal{O}(|\mathcal{V}| + |\mathcal{E}|)$ (details in App. A.6).

CPU-bound and Parallelizable: BONSAI does not involve learning any parameters through gradient descent or its variants. Hence, the entire process is CPU-bound. Furthermore, all steps are embarrassingly parallelizable leading distillation of datasets with hundreds of millions of edges within minutes.

Independence from model-architecture and hyper-parameters: Unlike majority of existing distillation algorithms that require knowledge of the GNN architecture and all training hyper-parameters, BONSAI only requires approximate knowledge of the number of layers L. An approximate knowledge is sufficient since the influence of far-away neighbors gets squashed (Topping et al., 2022). For the same reason, GNNS are typically run with $L \leq 3$. Hence, we recommend setting $L \sim 2$.

4 EXPERIMENTS

In this section, we benchmark BONSAI and establish:

- **Superior Accuracy:** BONSAI consistently outperforms existing baselines in terms of accuracy across various compression factors, datasets, and GNN architectures.
- **Enhanced Efficiency:** On average, BONSAI is 350, 22, and 1500 times faster than the state-of-the-art baselines GCOND, GDEM and GCSR, respectively, in distilling datasets.
- **Increased Robustness:** Unlike existing methods that require tuning distillation-specific hyper-parameters for each combination of GNN architecture, dataset, and compression ratio, BONSAI achieves superior performance using a single set of parameters across all scenarios.

Our implementation is available at https://anonymous.4open.science/r/bonsai.

Table 5: Accuracy achieved by the various baselines on benchmark datasets across various compression ratios over byte consumption (denoted as $\mathbf{S_r}(\%)$), , on the GCN architecture. The best and the second-best accuracies in each row are highlighted by dark and lighter shades of **Green**, respectively. OOT indicates the scenario where the algorithm failed to distill within 5 hours.

Dataset	$\mathbf{S_r}(\%)$	**Random**	**Herding**	**GCond**	**GDEM**	**GCSR**	**EXGC**	**GCSNTK**	**GEOM**	BONSAI	**Full**
Cora	0.5	39.90±1.46	45.61±0.01	77.30±0.30	57.49±6.87	74.83±0.75	34.87±0.83	77.85±0.76	33.76±0.96	83.95±0.39	88.56±0.18
	1	27.75±2.05	52.07±0.00	77.30±0.31	69.32±4.71	77.56±0.74	35.24±0.60	67.52±0.77	33.02±0.92	85.76±0.24	
	3	52.26±1.69	67.60±0.00	81.73±0.48	81.70±3.10	77.20±0.48	35.42±0.77	75.64±0.82	54.80±1.89	86.38±0.22	
CiteSeer	0.5	33.90±2.16	22.82±0.00	74.17±0.68	70.05±2.40	67.03±0.61	23.42±0.98	69.82±0.68	24.92±0.89	77.00±0.15	78.53±0.15
	1	44.90±2.32	49.10±0.02	77.62±0.71	72.48±2.13	74.77±0.78	23.67±1.00	69.22±0.71	28.03±0.81	77.03±0.33	
	3	44.50±1.27	67.69±0.01	77.02±0.22	76.20±0.55	77.27±0.28	25.07±0.92	64.26±0.53	33.48±0.83	75.89±0.26	
Pubmed	0.5	62.58±0.25	78.29±0.00	80.63±1.20	80.72±0.92	79.43±0.25	45.77±0.73	53.04±1.99	OOT	87.27±0.03	87.22±0.00
	1	79.19±0.09	78.59±0.00	79.92±0.00	80.80±1.07	79.11±0.15	46.24±0.43	62.81±1.32	OOT	87.08±0.04	
	3	82.50±0.09	78.09±0.00	77.00±0.15	81.07±0.90	79.94±0.16	47.62±0.67	67.72±2.01	OOT	87.64±0.09	
Flickr	0.5	44.78±0.00	47.98±0.01	44.06±1.05	46.25±1.02	46.41±0.00	45.47±0.85	31.49±0.75	OOT	48.73±0.27	50.93±0.17
	1	44.21±0.03	46.72±0.01	39.88±5.60	46.99±1.38	OOT (5 hrs)	45.97±0.82	42.50±0.99	OOT	49.05±0.17	
	3	46.56±0.01	46.54±0.01	46.04±1.88	47.35±0.98	OOT	48.44±0.65	36.58±0.82	OOT	49.66±0.27	
Ogbn-arxiv	0.5	42.01±0.01	53.37±0.00	52.63±0.63	54.73±0.66	OOT	60.66±1.66	61.55±1.17	OOT	58.49±0.17	68.97±0.10
	1	49.27±0.64	54.91±0.00	53.49±0.63	51.45±1.14	OOT	61.73±1.43	62.31±0.88	OOT	58.35±0.09	
	3	51.11±0.19	57.28±0.00	53.01±0.64	53.37±1.04	OOT	62.96±1.33	56.38±0.79	OOT	64.31±0.06	
Reddit	0.5	36.00±4.09	81.72±0.63	38.94±0.79	90.51±0.55	OOT	78.47±0.52	37.15±1.51	OOT	80.33±0.46	92.14±0.04
	1	38.55±2.00	83.48±0.83	43.98±0.35	90.63±0.84	OOT	81.69±1.12	38.87±2.00	OOT	85.65±0.08	
	3	44.97±2.97	88.51±0.13	48.78±0.83	85.75±0.80	OOT	OOM	47.48±1.98	OOT	88.90±0.07	

4.1 EXPERIMENTAL SETUP

The specifics of our experimental setup, including hardware and software environment, and hyper-parameters are detailed in App. B. For the baseline algorithms, we use the code shared by their respective authors. We conduct each experiment 5 times and report the means and standard deviations. Across all datasets, we maintain a train-validation-test split ratio of 60 : 20 : 20.

Datasets: Table 4 lists the benchmark datasets used.

Baselines: We compare BONSAI with GCOND (Jin et al., 2021), GDEM (Liu et al., 2024a), and GCSR (Liu et al., 2024b), EXGC (Fang et al., 2024), GCSNTK (Wang et al., 2024) and GEOM (Zhang et al., 2024). We omit SFGC (Zheng et al., 2023) and SGDD (Yang et al., 2023) since both GDEM and GCSR have been shown to outperform them. Among non-neural, baselines, we compare with selecting the induced subgraph spanned by a random selection of nodes, and Herding (Welling, 2009).

Table 4: Datasets.

Dataset	**# Nodes**	**# Edges**	**# Classes**	**# Features**
Cora (Kipf & Welling, 2017)	2,708	10,556	7	1,433
Citeseer (Kipf & Welling, 2017)	3,327	9,104	6	3,703
Pubmed (Kipf & Welling, 2017)	19,717	88,648	3	500
Flickr (Zeng et al., 2020)	89,250	899,756	7	500
Ogbn-arxiv (Hu et al., 2021)	169,343	2,315,598	40	128
Reddit (Hamilton et al., 2017)	232,965	23,213,838	41	602

Metrics: Prediction quality is measured through Accuracy on the test set, i.e., the percentage of correct predictions. Compression ratio is quantified as $S_r = \frac{\text{size of distilled dataset in bytes}}{\text{size of full data set in bytes}}$.

4.2 PREDICTION ACCURACY

Table 5 presents the accuracies obtained by BONSAI and the baselines on GCN. BONSAI achieves the best accuracy in 12 out of 18 scenarios, with substantial improvements ($\geq 5\%$) over the second-best performer in several cases. This demonstrates that the unsupervised gradient-agnostic approach adopted by BONSAI does not come at the cost of accuracy. GDEM emerges as the second best perfomer. GCSR fails to complete in the two largest datasets of Ogbn-arxiv and Reddit, since it trains on the entire dataset 100 times to distill, which exceeds 5 hours. GEOM is even slower since it trains on full train set 200 times.

4.3 CROSS-ARCHITECTURE GENERALIZATION

GDEM and BONSAI are both model-agnostic, meaning their distilled datasets are independent of the downstream GNN architecture used. In contrast, GCOND and GCSR produce architecture-specific distillation datasets, requiring separate training for each architecture. This distinction raises two important questions: First, *how well do the distilled datasets generated by GDEM and BONSAI generalize across different architectures?* Second, if we apply the distillation datasets produced by GCOND and GCSR for GCN to other architectures, how significantly does it impact performance? We explore these questions in Table 6.

Consistent with the earlier trend, BONSAI continues to outperform all baselines. We further observe that the performance gap between BONSAI and the baselines is wider in GAT and GIN compared to

Table 6: Accuracies achieved by the various baselines on GAT and GIN.

Dataset	%	GNN	Random	Herding	GCond	GDEM	GCSR	BONSAI	Full
Cora	0.5	GAT	41.44±1.73	33.80±0.07	13.21±1.99	63.91±5.91	15.09±6.19	75.42±1.61	85.70±0.09
	1	GAT	42.73±1.03	46.09±0.86	35.24±0.00	73.49±2.64	37.60±1.34	78.67±0.89	
	3	GAT	60.22±0.67	56.75±0.45	35.24±0.00	75.28±4.86	36.72±0.81	80.66±0.80	
	0.5	GIN	49.04±0.50	34.39±1.03	14.13±6.80	63.65±7.11	76.05±0.44	85.42±0.74	86.62±0.28
	1	GIN	50.48±0.85	33.80±2.42	33.91±1.23	75.92±4.24	60.70±4.44	84.80±0.41	
	3	GIN	59.52±0.88	36.35±0.59	31.70±4.97	59.59±7.95	51.62±5.00	85.42±0.53	
CiteSeer	0.5	GAT	42.76±0.35	36.04±0.46	21.47±0.00	69.86±2.28	21.92±0.76	68.56±0.57	77.48±0.75
	1	GAT	46.19±1.38	52.07±0.11	21.47±0.00	23.87±3.05	21.50±0.06	69.43±0.82	
	3	GAT	61.65±0.51	65.17±0.00	21.26±0.22	22.90±1.20	21.50±0.06	69.94±1.15	
	0.5	GIN	44.86±0.43	22.97±0.30	21.47±0.00	67.69±3.28	50.66±1.17	71.80±0.26	75.45±0.23
	1	GIN	47.90±0.65	39.67±0.82	19.49±1.09	67.64±4.45	64.74±1.88	72.16±0.60	
	3	GIN	61.83±0.68	60.48±0.26	18.65±2.56	48.65±8.17	59.95±9.07	70.51±0.54	
PubMed	0.5	GAT	77.73±0.12	75.44±0.02	37.49±4.01	80.06±1.16	38.29±8.13	85.66±0.38	86.33±0.08
	1	GAT	78.85±0.09	76.64±0.02	41.55±3.18	80.75±0.47	40.47±0.00	85.88±0.28	
	3	GAT	82.84±0.11	78.48±0.03	37.77±3.61	65.08±9.53	40.27±0.20	85.62±0.36	
	0.5	GIN	77.45±0.14	48.48±1.33	30.91±4.57	78.78±0.91	36.88±12.06	84.32±0.33	84.66±0.05
	1	GIN	78.43±0.22	62.22±0.13	32.84±6.27	78.72±0.95	33.75±5.58	85.57±0.26	
	3	GIN	80.56±0.17	45.40±0.46	36.11±3.47	81.08±0.99	32.01±6.77	85.66±0.23	
Flickr	0.5	GAT	43.64±0.99	36.50±13.22	40.24±3.20	25.43±10.37	28.03±6.60	48.22±3.60	51.42±0.07
	1	GAT	43.56±1.06	36.34±1.14	40.85±1.08	18.44±9.42	OOT	45.62±1.85	
	3	GAT	45.71±1.87	42.70±1.17	41.51±9.81	25.83±11.39	OOT	47.80±2.06	
	0.5	GIN	42.67±0.83	39.98±7.21	13.65±7.54	14.10±5.68	05.92±1.01	44.97±2.23	45.37±0.57
	1	GIN	42.90±0.76	41.87±4.52	16.65±6.55	19.44±9.68	OOT	44.90±0.88	
	3	GIN	19.63±4.21	43.72±3.26	24.25±14.43	20.97±6.64	OOT	45.04±1.94	

GCN (Table 5). For GCOND and GCSR, this wider gap is not surprising since they are trained on the gradients of GCN, which are expected to differ from those of GIN and GAT. GIN uses a SumPool aggregation, while GAT employs attention to effectively dampen unimportant edges, with a potentially different eigen spectrum - the property that GDEM attempts to preserve. The approach of BONSAI, on the other hand, aligns more directly with the computational structure of GNNS. It focuses on analyzing the input space of computation trees and aims to represent as much of this space as possible within the given budget.

4.4 DISTILLATION EFFICIENCY

Table. 7 presents the time consumed by BONSAI and other baselines. We note that while BONSAI is CPU-bound, all other algorithms are GPU-bound. Despite being CPU-bound, BONSAI, on average, is more than 7-times faster than the fastest baseline EXGC. In addition, all baselines require training on the full dataset. This algorithm design shows up in the running times where the distillation time is higher than the full dataset training time, negating the very purpose of graph distillation. More worryingly, the distillation process of all algorithms, except BONSAI, have higher carbon emissions than training on the full dataset (See Table 9 in Appendix).

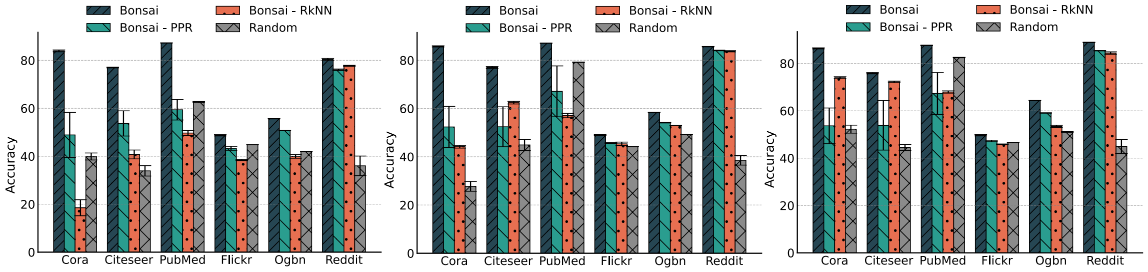
Table 7: Distillation times of various methods in seconds at 0.5%. Distillation times that are higher than training on the full dataset itself are highlighted in red. The fastest distillation time for each dataset is highlighted in green.

Dataset	GCOND	GDEM	GCSR	EXGC	GC-SNTK	GEOM	BONSAI	Full
Cora	2738	105	5260	34.87	82	12996	2.76	24.97
Citeseer	2712	167	6636	34.51	124	15763	2.61	24.87
Pubmed	2567	530	1319	114.96	117	OOT	24.84	51.06
Flickr	1935	3405	17445	243.28	612	OOT	118.23	180.08
Ogbn-arxiv	14474	569	OOT	1594.83	12218	OOT	348.24	524.67
Reddit	30112	20098	OOT	6903.47	29211	OOT	1445.00	2425.68

Component-wise analysis: We discuss the individual time consumption by each component of BONSAI in App. B.4.1.

4.5 ABLATION STUDY AND IMPACT OF PARAMETERS

Fig. 4 presents an ablation study comparing BONSAI against three variants: **(i)** BONSAI-PPR, which omits the use of Personalized PageRank, **(ii)** BONSAI-Rev-k-NN, which substitutes the Rev-k-NN-based coverage maximization with random exemplar selection, but performs PPR, **(iii)** and random that does not do either PPR or Rev-k-NN. Before analyzing the trends, we emphasize an important distinction between BONSAI-Rev-k-NN and Random. While in BONSAI-Rev-k-NN, we randomly add computation trees till the distillation budget is exhausted, in Random, we iteratively add random nodes, till their induced subgraph exhausts the budget. Consequently, Random covers more diverse



(a) 0.5% (b) 1% (c) 3%

Figure 4: Ablation study of Bonsai.

nodes with partial local neighborhood information, while Bonsai-Rev-k-NN selects a smaller node set with complete L-hop topology, enabling precise Gnn embedding computation.

Several important insights emerge from this experiment. Random performs significantly worse than Bonsai, demonstrating that the distilled graph's information content substantially exceeds that of an equally sized subgraph induced by random node selection. This finding, consistent with the trend observed in Table 5, underscores the effectiveness of Bonsai over random sampling.

Second, removal of either Rev-k-NN or PPR result in significant drops across most datasets, indicating both play important and complementary roles. While Rev-k-NN identifies the exemplars that are important, PPR sparsifies the graph making space for more exemplars to be added. Interestingly, the relative impact of these components varies with dataset compression and size. At higher compression rates (such as 0.5%), removing Rev-k-NN leads to higher drop in accuracy across most datasets. However, this trend reverses as the distilled dataset size increases. Closer examination reveals that higher-ranked exemplars typically have higher degrees in their L-hop neighborhoods. When the memory budget is constrained, high-ranked exemplars are often skipped because their L-hop neighborhoods exceed the available memory. This limitation diminishes as the budget increases. Consequently, at higher budgets, edge density increases, making graph sparsification (via PPR) increasingly critical. This explains why PPR becomes more important at higher budgets or in inherently dense graphs like Reddit.

Finally, we discuss the performance comparison of Random with Bonsai-Rev-k-NN. This is an interesting comparison since both represent two distinct mechanisms of random selection. In most cases, Bonsai-Rev-k-NN achieves a higher accuracy indicating that obtaining full L-hop topological information for a smaller set of nodes leads to better results than partial L-hop information over a broader set of nodes.

Impact of Parameters: We analyze the impact of sampling size and k on Rev-k-NN in App. B.4.2.

5 CONCLUSIONS AND FUTURE WORKS

In this work, we have developed Bonsai, a novel graph distillation method that addresses critical limitations in existing approaches. By leveraging the fundamental role of computation trees in message-passing Gnns, Bonsai achieves superior performance in node classification tasks across multiple real-world datasets. Our method stands out as the first linear-time, model-agnostic graph distillation algorithm, offering significant improvements in both accuracy and computational efficiency. Bonsai's unique approach of encoding exemplar trees that maximize representation of the full training set's computation trees has proven to be highly effective. This strategy not only overcomes the paradoxical requirement of training on full datasets for distillation but also eliminates the need for repeated distillation when changing hyper-parameters or Gnn architectures. Furthermore, Bonsai achieves substantial size reduction without resorting to fully-connected, edge-weighted graphs, thereby reducing computational demands. In contrast to baselines, Bonsai is completely CPU-bound, resulting in a smaller greenhouse footprint along with reduced computation demands, making it a more environmentally friendly option. These features collectively position Bonsai as a significant advancement in the field of graph distillation, offering both performance benefits and sustainability advantages.

Limitations and Future Works: While existing research on graph distillation has primarily focused on node and graph classification tasks, Gnns have demonstrated their versatility across a broader spectrum of applications. In our future work, we aim to expand the scope of graph distillation by developing task-agnostic data distillation algorithms.

REPRODUCIBILITY STATEMENT

To support the reproducibility of our work, we provide several resources in the paper and its supplementary materials. The source code of our models and algorithms is available at https://anonymous.4open.science/r/bonsai, which also includes details of how to train the baseline models in different settings. All theoretical results and assumptions are detailed in the Appendix A, ensuring clarity in our claims. Full experimental settings are documented in Appendix B.

REFERENCES

Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, and Clifford Stein. *Introduction to Algorithms, Third Edition*. The MIT Press, 3rd edition, 2009. ISBN 0262033844. (Cited on pp. **7 and 15** ←˒)

Junfeng Fang, Xinglin Li, Yongduo Sui, Yuan Gao, Guibin Zhang, Kun Wang, Xiang Wang, and Xiangnan He. Exgc: Bridging efficiency and explainability in graph condensation. In *Proceedings of the ACM Web Conference 2024*, WWW '24, pp. 721–732, New York, NY, USA, 2024. Association for Computing Machinery. ISBN 9798400701719. doi: 10.1145/3589334.3645551. URL https://doi.org/10.1145/3589334.3645551. (Cited on pp. **2 and 8** ←˒)

Uriel Feige. A threshold of ln n for approximating set cover. *J. ACM*, 45(4):634–652, 1998. (Cited on p. **7** ←˒)

Mridul Gupta, Sahil Manchanda, HARIPRASAD KODAMANA, and Sayan Ranu. Mirage: Model-agnostic graph distillation for graph classification. In *The Twelfth International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=78iGZdqxYY. (Cited on p. **1** ←˒)

William L. Hamilton, Rex Ying, and Jure Leskovec. Inductive representation learning on large graphs. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, NIPS'17, pp. 1025–1035, Red Hook, NY, USA, 2017. Curran Associates Inc. ISBN 9781510860964. (Cited on pp. **1, 2, and 8** ←˒)

Weihua Hu, Matthias Fey, Hongyu Ren, Maho Nakata, Yuxiao Dong, and Jure Leskovec. Ogb-lsc: A large-scale challenge for machine learning on graphs. *arXiv preprint arXiv:2103.09430*, 2021. (Cited on pp. **1 and 8** ←˒)

Wei Jin, Lingxiao Zhao, Shichang Zhang, Yozen Liu, Jiliang Tang, and Neil Shah. Graph condensation for graph neural networks. In *International Conference on Learning Representations*, 2021. (Cited on pp. **1, 2, and 8** ←˒)

Wei Jin, Xianfeng Tang, Haoming Jiang, Zheng Li, Danqing Zhang, Jiliang Tang, and Bing Yin. Condensing graphs via one-step gradient matching. In *Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pp. 720–730, 2022. (Cited on p. **1** ←˒)

Thomas N Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. *ICLR*, 2017. (Cited on pp. **1, 2, and 8** ←˒)

Alexandre Lacoste, Alexandra Luccioni, Victor Schmidt, and Thomas Dandres. Quantifying the carbon emissions of machine learning. *arXiv preprint arXiv:1910.09700*, 2019. (Cited on p. **22** ←˒)

Jure Leskovec, Andreas Krause, Carlos Guestrin, Christos Faloutsos, Jeanne VanBriesen, and Natalie Glance. Cost-effective outbreak detection in networks. In *Proceedings of the 13th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 420–429, 2007. (Cited on p. **7** ←˒)

Yang Liu, Deyu Bo, and Chuan Shi. Graph distillation with eigenbasis matching. In *Forty-first International Conference on Machine Learning*, 2024a. URL https://openreview.net/forum?id=DYN66IJCI9. (Cited on pp. **1, 2, and 8** ←˒)

Zhanyu Liu, Chaolv Zeng, and Guanjie Zheng. Graph data condensation via self-expressive graph structure reconstruction. In *Proceedings of the 30th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, KDD '24, pp. 1992–2002, 2024b. (Cited on pp. **1, 2, and 8** ↩)

Peter A. Lofgren, Siddhartha Banerjee, Ashish Goel, and C. Seshadhri. Fast-ppr: scaling personalized pagerank estimation for large graphs. In *Proceedings of the 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '14, pp. 1436–1445, 2014. (Cited on p. **17** ↩)

WeiWei Miao, Zeng Zeng, Mingxuan Zhang, Siping Quan, Zhen Zhang, Shihao Li, Li Zhang, and Qi Sun. Workload prediction in edge computing based on graph neural network. In *2021 IEEE Intl Conf on Parallel & Distributed Processing with Applications, Big Data & Cloud Computing, Sustainable Computing & Communications, Social Computing & Networking (ISPA/BDCloud/SocialCom/SustainCom)*, pp. 1663–1666, 2021. doi: 10.1109/ISPA-BDCloud-SocialCom-SustainCom52081.2021.00223. (Cited on p. **1** ↩)

Rajeev Motwani and Prabhakar Raghavan. *Randomized Algorithms*. Cambridge University Press, 1995. (Cited on p. **14** ↩)

Lawrence Page, Sergey Brin, Rajeev Motwani, and Terry Winograd. The PageRank Citation Ranking: Bringing Order to the Web. Technical report, 1998. (Cited on p. **17** ↩)

Nino Shervashidze, Pascal Schweitzer, Erik Jan van Leeuwen, Kurt Mehlhorn, and Karsten M. Borgwardt. Weisfeiler-lehman graph kernels. *J. Mach. Learn. Res.*, 12(null):2539–2561, nov 2011. ISSN 1532-4435. (Cited on p. **4** ↩)

Matteo Togninalli, Elisabetta Ghisu, Felipe Llinares-López, Bastian Rieck, and Karsten Borgwardt. Wasserstein weisfeiler–lehman graph kernels. In *Advances in Neural Information Processing Systems 32 (NeurIPS)*, pp. 6436–6446. 2019. (Cited on pp. **3 and 4** ↩)

Jake Topping, Francesco Di Giovanni, Benjamin Paul Chamberlain, Xiaowen Dong, and Michael M. Bronstein. Understanding over-squashing and bottlenecks on graphs via curvature. In *ICLR*, 2022. (Cited on p. **7** ↩)

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. (Cited on pp. **1 and 2** ↩)

Lin Wang, Wenqi Fan, Jiatong Li, Yao Ma, and Qing Li. Fast graph condensation with structure-based neural tangent kernel. In *Proceedings of the ACM Web Conference 2024*, WWW '24, pp. 4439–4448, New York, NY, USA, 2024. Association for Computing Machinery. ISBN 9798400701719. doi: 10.1145/3589334.3645694. URL https://doi.org/10.1145/3589334.3645694. (Cited on pp. **2 and 8** ↩)

Max Welling. Herding dynamical weights to learn. In *Proceedings of the 26th Annual International Conference on Machine Learning*, pp. 1121–1128, 2009. (Cited on p. **8** ↩)

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. (Cited on pp. **2, 3, and 4** ↩)

Zhe Xu, Yuzhong Chen, Menghai Pan, Huiyuan Chen, Mahashweta Das, Hao Yang, and Hanghang Tong. Kernel ridge regression-based graph dataset distillation. In *Proceedings of the 29th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, KDD '23, pp. 2850–2861, New York, NY, USA, 2023. Association for Computing Machinery. ISBN 9798400701030. doi: 10.1145/3580305.3599398. URL https://doi.org/10.1145/3580305.3599398. (Cited on p. **1** ↩)

Beining Yang, Kai Wang, Qingyun Sun, Cheng Ji, Xingcheng Fu, Hao Tang, Yang You, and Jianxin Li. Does graph distillation see like vision dataset counterpart? In *Thirty-seventh Conference on Neural Information Processing Systems*, 2023. URL https://openreview.net/forum?id=VqIWgUVsXc. (Cited on pp. **2 and 8** ↩)

Hanqing Zeng, Hongkuan Zhou, Ajitesh Srivastava, Rajgopal Kannan, and Viktor Prasanna. GraphSAINT: Graph sampling based inductive learning method. In *International Conference on Learning Representations*, 2020. URL https://openreview.net/forum?id=BJe8pkHFwS. (Cited on p. **8** ↩)

Yuchen Zhang, Tianle Zhang, Kai Wang, Ziyao Guo, Yuxuan Liang, Xavier Bresson, Wei Jin, and Yang You. Navigating complexity: Toward lossless graph condensation via expanding window matching. In *Forty-first International Conference on Machine Learning*, 2024. URL https://openreview.net/forum?id=gE7qZurGH3. (Cited on pp. **2 and 8** ↩)

Xin Zheng, Miao Zhang, Chunyang Chen, Quoc Viet Hung Nguyen, Xingquan Zhu, and Shirui Pan. Structure-free graph condensation: From large-scale graphs to condensed graph-free data. In *Thirty-seventh Conference on Neural Information Processing Systems*, 2023. URL https://openreview.net/forum?id=XkcufOcgUc. (Cited on pp. **1, 2, and 8** ↩)

A APPENDIX

A.1 DETAILED COMPARISON TO MIRAGE

The similarities between MIRAGE and Bonsai end at their shared goal of analyzing computation trees for distillation. The fundamental limitation of MIRAGE lies in relying on tree isomorphisms, which makes them limited to graphs annotated with only single, discrete node labels (such as atom-types in chemical compounds). Hence, MIRAGE does not work on general purpose graphs where nodes are annotated with high-dimensional feature vectors.

Summary of MIRAGE: Given a graph database, MIRAGE identifies all unique computation trees of a certain depth in the graph database. Two trees are non-unique if they are isomorphic to each other. Next, each graph is represented as a set of computation trees, on which frequent itemset mining is performed. Trees in these frequent itemsets form the distilled dataset.

What breaks when applied to our setting? Tree isomorphism does not work when each node is attributed with high-dimensional feature vectors since treating feature vectors as labels means two trees are isomorphic if they are topologically identical and the mapped nodes across the graphs are annotated with identical feature vectors. In BONSAI, we make no such assumption and hence any algorithm built on tree isomorphism is not feasible.

The algorithm design of BONSAI is entirely different. We make the following novel contributions:

- **Reverse k-NN to rank tree importance:** BONSAI embeds computation trees into a feature space using WL-kernel and **ranks** each tree based on the density in its neighborhood (Reverse k-NN).
- **Fast reverse k-NN through sampling:** Reverse k-NN is expensive ($O(n^2)$). BONSAI proposes an efficient sampling strategy with provable guarantees on the sample_size-approximation_error trade-off.
- **Coverage maximization:** The core idea is to select a subset of trees that are representative of the entire set. Hence, we select trees located in dense neighborhoods and diverse from each other. This ensures that all trees that are not selected in distilled set are likely to have a close neighbor (in the embedding space) in the distilled set. This is achieved through coverage maximization (§ 3.2).
- **Theoretical guarantees:** We prove coverage maximization is NP-hard, monotonic and submodular. Hence, greedy selection provides $1 - 1/e$ approximation guarantee.
- **Sparsification:** Sparsification of the distilled data set is performed by personalized page rank.

Empirical validation: To further leave no room for ambiguity, we applied Mirage on Cora and Citeseer, and in both dataset all computation trees were unique leading to no compression.

A.2 PROOF. OF THEOREM 1.

Recall the notation that $\mathbb{S} \subseteq \mathbb{T}$ denotes the subset of trees from which the representative power of all trees in $\mathbb{T}$ is being estimated. Let $|\mathbb{S}| = z$.

Corresponding to each $\mathcal{T}_u^L \in \mathbb{S}$, let us define an indicator random variable X_u^v denoting whether $\mathcal{T}_v^L \in k$-NN$(\mathcal{T}_u^L)$. Therefore, we have:

$$X^v = \sum_{\forall \mathcal{T}_u^L \in \mathbb{S}} X_u^v = z\widetilde{\Pi}\left(\mathcal{T}_v^L\right) \tag{8}$$

Since each tree in $\mathbb{S}$ is sampled uniformly at random from $\mathbb{T}$, we have $E(X^v) = z\Pi\left(\mathcal{T}_v^L\right)$. We also note that $X^v \sim Binomial(z, \Pi\left(\mathcal{T}_v^L\right))$. Given any $\epsilon \geq 0$, from Chernoff bounds (Motwani &

Raghavan, 1995), we have:

Upper tail bound:

$$P\left(X^v \geq (1+\epsilon) z \Pi\left(\mathcal{T}_v^L\right)\right) \leq \exp\left(-\frac{\epsilon^2}{2+\epsilon} z \Pi\left(\mathcal{T}_v^L\right)\right) \tag{9}$$

Lower tail bound:

$$P\left(X^v \leq (1-\epsilon) z \Pi\left(\mathcal{T}_v^L\right)\right) \quad \leq \exp\left(-\frac{\epsilon^2}{2} z \Pi\left(\mathcal{T}_v^L\right)\right) \tag{10}$$

Combining the upper and tail bounds, we obtain:

$$\begin{aligned}
P\left(\left|X^v - z\Pi\left(\mathcal{T}_v^L\right)\right| \geq \epsilon z \Pi\left(\mathcal{T}_v^L\right)\right) &\leq 2\exp\left(-\frac{\epsilon^2}{2+\epsilon} z \Pi\left(\mathcal{T}_v^L\right)\right) &(11)\\
\Leftrightarrow P\left(\left|z\widetilde{\Pi}(\mathcal{T}_v^L) - z\Pi\left(\mathcal{T}_v^L\right)\right| \geq \epsilon z \Pi\left(\mathcal{T}_v^L\right)\right) &\leq 2\exp\left(-\frac{\epsilon^2}{2+\epsilon} z \Pi\left(\mathcal{T}_v^L\right)\right) &(12)\\
\Leftrightarrow P\left(\left|\widetilde{\Pi}(\mathcal{T}_v^L) - \Pi\left(\mathcal{T}_v^L\right)\right| \geq \epsilon \Pi\left(\mathcal{T}_v^L\right)\right) &\leq 2\exp\left(-\frac{\epsilon^2}{2+\epsilon} z \Pi\left(\mathcal{T}_v^L\right)\right) &(13)
\end{aligned}$$

Plugging θ from Eq. 5 in Eq. 13, we obtain $\theta = \epsilon \Pi\left(\mathcal{T}_v^L\right)$. Hence,

$$\begin{aligned}
P\left(\left|\widetilde{\Pi}(\mathcal{T}_v^L) - \Pi\left(\mathcal{T}_v^L\right)\right| \geq \theta\right) &\leq 2\exp\left(-\frac{\left(\frac{\theta}{\Pi(\mathcal{T}_v^L)}\right)^2}{2+\frac{\theta}{\Pi(\mathcal{T}_v^L)}} z \Pi\left(\mathcal{T}_v^L\right)\right) &(14)\\
&\leq 2\exp\left(-\frac{\theta^2}{2\Pi\left(\mathcal{T}_v^L\right)+\theta} z\right) &(15)
\end{aligned}$$

For any $\mathcal{T}_v^L \in \mathbb{T},\ \Pi\left(\mathcal{T}_v^L\right) \leq 1$. Thus,

$$P\left(\left|\widetilde{\Pi}(\mathcal{T}_v^L) - \Pi\left(\mathcal{T}_v^L\right)\right| \geq \theta\right) \leq 2\exp\left(-\frac{\theta^2}{2+\theta} z\right) \tag{16}$$

Now, if we want a confidence interval of $1-\delta$ (as stated in Eq. 5), we have:

$$\begin{aligned}
\delta &\geq 2\exp\left(-\frac{\theta^2}{2+\theta} z\right) &(17)\\
\Leftrightarrow \frac{\delta}{2} &\geq \exp\left(-\frac{\theta^2}{2+\theta} z\right) &(18)\\
\Leftrightarrow \ln\frac{\delta}{2} &\geq -\frac{\theta^2}{2+\theta} z &(19)\\
\Leftrightarrow \ln\frac{2}{\delta} &\leq \frac{\theta^2}{2+\theta} z &(20)\\
\Leftrightarrow \ln\frac{2}{\delta} &\leq \frac{\theta^2}{2+\theta} z &(21)\\
\Leftrightarrow z &\geq \frac{\ln\left(\frac{2}{\delta}\right)(2+\theta)}{\theta^2} &(22)
\end{aligned}$$

Hence, if we sample at least $\frac{\ln\left(\frac{2}{\delta}\right)(2+\theta)}{\theta^2}$ trees in $\mathbb{S}$, then for any $\mathcal{T}_v^L \in \mathbb{T}$:

$$\widetilde{\Pi}(\mathcal{T}_v^L) \in \left[\Pi\left(\mathcal{T}_v^L\right) - \theta, \Pi\left(\mathcal{T}_v^L\right) + \theta\right] \text{ with probability at least } 1-\delta. \qquad \square \tag{23}$$

A.3 NP-HARDNESS: PROOF OF THEOREM 1

PROOF. To establish NP-hardness of our proposed problem we reduce it to the classical *Set Cover* problem (Cormen et al., 2009).

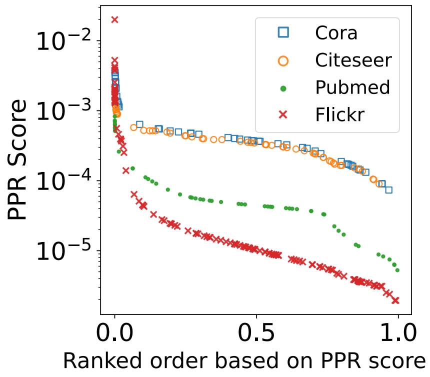


Figure 5: Distribution of PPR scores against node ranks.

Definition 7 (Set Cover)**.** *Given a budget b and a collection of subsets $\mathcal{S} = \{S_1, \cdots, S_m\}$ from a universe of items $U = \{u_1, \cdots, u_n\}$, i.e., $\forall S_i \in \mathcal{S}$, $S_i \subseteq U$, the Set Cover problem seeks to determine whether there exists a subset $\mathcal{S}' \subseteq \mathcal{S}$ such that $|\mathcal{S}'| = b$ and it covers all items in the universe, i.e., $|\bigcup_{\forall S_i \in \mathcal{S}'} S_i| = U$.*

We show that given any instance of a Set Cover problem $\langle \mathcal{S}, U \rangle, b$, it can be mapped to the problem of maximizing the representative power (Eq. 7).

Given an instance of the Set Cover problem, we construct a database of computation trees $\mathbb{T} = \mathbb{T}_U \cup \mathbb{T}_{\mathcal{S}}$. For each $u_j \in U$, we add a computation tree $\mathcal{T}_{u_j} \in \mathbb{T}_U$. For each $S_i \in \mathcal{S}$, we add a tree $\mathcal{T}_{S_i} \in \mathbb{T}_{\mathcal{S}}$. If an item $u_j \in S_i$, then we have $\mathcal{T}_{u_j} \in$ Rev-k-NN$(\mathcal{T}_{S_i})$ (equivalently $\mathcal{T}_{S_i} \in k$-NN$(\mathcal{T}_{u_j})$).

With this construction, we can state that there exists a Set Cover of size b if and only if there exists a solution set $\mathbb{A} \subseteq \mathbb{T}$ such that $\Pi(\mathbb{A}) = \frac{|\mathbb{T}_U|}{|\mathbb{T}|}$. ☐

A.4 PROOFS OF MONOTONICITY, SUBMODULARITY

Theorem 3 (Monotonicity)**.** $\forall\, \mathbb{A}' \supseteq \mathbb{A}$, $\Pi(\mathbb{A}') - \Pi(\mathbb{A}) \geq 0$, *where $\mathbb{A}$ and $\mathbb{A}'$ are sets of computation trees.*

Proof. Since the denominator in $\Pi(\mathbb{A})$ is constant, it suffices to prove that the numerator is monotonic. This reduces to establishing that:

$$\left| \bigcup_{\forall \mathcal{T}_v^L \in \mathbb{A}'} \text{Rev-}k\text{-NN}\left(\mathcal{T}_v^L\right) \right| \geq \left| \bigcup_{\forall \mathcal{T}_v^L \in \mathbb{A}} \text{Rev-}k\text{-NN}\left(\mathcal{T}_v^L\right) \right| \tag{24}$$

This inequality holds true because the union operation is a monotonic function. As $\mathbb{A}'$ is a superset of $\mathbb{A}$, the union over $\mathbb{A}'$ will always include at least as many elements as the union over $\mathbb{A}$. ☐

Theorem 4 (Submodularity)**.** $\forall \mathbb{A}' \supseteq \mathbb{A}$, $\Pi(\mathbb{A}' \cup \{\mathcal{T}\}) - \Pi(\mathbb{A}') \leq \Pi(\mathbb{A} \cup \{\mathcal{T}\}) - \Pi(\mathbb{A})$, *where $\mathbb{A}$ and $\mathbb{A}'$ are sets of computation trees and $\mathcal{T}$ is a computation tree.*

PROOF BY CONTRADICTION. Let us assume

$$\exists \mathbb{A}' \supseteq \mathbb{A},\ \Pi(\mathbb{A}' \cup \{\mathcal{T}\}) - \Pi(\mathbb{A}') > \Pi(\mathbb{A} \cup \{\mathcal{T}\}) - \Pi(\mathbb{A}) \tag{25}$$

Eq. 25 implies:

$$\text{Rev-}k\text{-NN}(\mathcal{T}) \setminus \bigcup_{\forall \mathcal{T}' \in \mathbb{A}'} \text{Rev-}k\text{-NN}\left(\mathcal{T}'\right) \supseteq \text{Rev-}k\text{-NN}(\mathcal{T}) \setminus \bigcup_{\forall \mathcal{T}' \in \mathbb{A}} \text{Rev-}k\text{-NN}\left(\mathcal{T}'\right) \tag{26}$$

$$\implies \mathbb{A} \supseteq \mathbb{A}',\ \text{which is a contradiction.}$$ ☐

A.5 SPARSIFICATION THROUGH PERSONALIZED PAGERANK (PPR)

Let $\mathcal{V}_S$ be the node set in the initial distilled graph, which can be partitioned into two disjoint subsets: $\mathcal{V}_{root}$ and $\mathcal{V}_{ego}$, where $\mathcal{V}_{root} = \{v \mid \exists \mathcal{T}_v^L \in \mathbb{A}\}$, the *root nodes*, that serve as the root node of some exemplar in $\mathbb{A}_{greedy}$, and $\mathcal{V}_{ego} = \mathcal{V}_S \setminus \mathcal{V}_{root}$, the *ego nodes*, that are included because they fall within the L-hop neighborhood of a root node. We propose to employ *Personalized PageRank (PPR)* (Page et al., 1998) with teleportation only to root nodes to identify and prune ego nodes with minimal impact on the root node embeddings. Starting with timestamp $t = 0$, we proceed as follows:

1. Compute the PPR distribution $\boldsymbol{\pi}_t$ for $t \geq 0$ defined by:

$$\boldsymbol{\pi}_t = (1-\beta)\mathbf{A}\boldsymbol{\pi}_{t-1} + \beta\mathbf{e} \tag{27}$$

   where: $\boldsymbol{\pi}_0[i] = \frac{1}{|\mathcal{V}_S|}$ for all i, $\mathbf{e}[i] = \frac{1}{|\mathcal{V}_{root}|}$ if $v_i \in \mathcal{V}_{root}$, otherwise 0, $\mathbf{A}$ is the normalized adjacency matrix, β is the teleportation probability.
2. Obtain the PPR vector $\boldsymbol{\pi}$ after convergence where $\boldsymbol{\pi}_t = \boldsymbol{\pi}_{t-1}$.
3. Sort the nodes in $\boldsymbol{\pi}$ by their PPR scores in descending order such that $\forall i,\ \boldsymbol{\pi}[i-1] \geq \boldsymbol{\pi}[i]$.
4. Define the knee point i_{knee} as:

$$i_{knee} = \arg\max_i \{\boldsymbol{\pi}[i+1] + \boldsymbol{\pi}[i-1] - 2 \cdot \boldsymbol{\pi}[i]\} \tag{28}$$

5. Prune all ego nodes with PPR scores below $\boldsymbol{\pi}[i_{knee}]$.
6. Use the freed-up space to include additional exemplar trees using Algorithm 1.
7. Repeat steps 1–6 iteratively until $i_{knee} \geq \theta$, where θ is a threshold setting the minimum number of nodes to be removed.

The rationale for this pruning process can be outlined as follows. The PPR distribution $\boldsymbol{\pi}_t$ iteratively updates based on the transition matrix $\mathbf{A}$ and teleportation vector $\mathbf{e}$. Given that $\boldsymbol{\pi}_0$ is initialized uniformly and $\mathbf{e}$ assigns higher probabilities to root nodes, $\boldsymbol{\pi}_t$ converges to a distribution where nodes more connected to root nodes have higher scores (See Fig. 5 in Appendix). After convergence, the vector $\boldsymbol{\pi}$ represents the steady-state probabilities of nodes in the graph. Sorting $\boldsymbol{\pi}$ by descending scores ensures that nodes with the highest influence on root nodes are prioritized. The knee point i_{knee} is identified as the maximum curvature in the sorted PPR scores. This point is where the rate of change in scores shifts most significantly, indicating a transition between nodes of high and low influence. Nodes with PPR scores below $\boldsymbol{\pi}[i_{knee}]$ have minimal impact on root node embeddings and removing these nodes and edges incident on them effectively sparsifies the graph without substantial loss of information. Additional exemplar trees can then be incorporated into the available space using the greedy algorithm as given by Algorithm 1.

A.6 COMPLEXITY ANALYSIS

For this analysis, we assume $|\mathcal{V}| = n$ and $|\mathcal{E}| = m$. In sync with Fig. 3, the computation structure of BONSAI can be decomposed into four components.

1. **Embed trees into WL-space:** This operation requires a pass through each twice, and is identical to the message-pass structure of a GNN. Hence, the computation cost is bounded to $\mathcal{O}(m)$. A crucial distinction from a GNN is that it does not require back-propagation and hence is fully CPU-bound.
2. **Rev-k-NN:** As discussed in § 3.1.1, computing Rev-k-NN consumes $\mathcal{O}(n)$ time with sampling.
3. **Coverage Maximization:** Each iteration in Alg. 1, iterates over all nodes (trees) in the graph ($\mathbb{T}$) that have not yet been added to the set of exemplars. For each node (equivalently, tree rooted at this node), we compute its marginal gain (line 3 in Alg. 1), which is bounded by the sample size z, since the cardinality of a Rev-k-NN set is bounded by the number of trees sampled for Rev-k-NN computation ((§ 3.1.1)). Since $z \ll n$, the complexity per iteration is $\mathcal{O}(zn) \approx \mathcal{O}(n)$. The number of iterations is bounded by the size of the distillation budget, which we typically set to less than 3% of the full dataset and hence has negligible impact on the complexity.
4. **PPR:** PPR can be computed in linear time (Lofgren et al., 2014). Since the size of the distilled set is at most the full graph, this bounds the cost of PPT to $\mathcal{O}(n)$.

Combining all these components, the cost of BONSAI is bounded by $\mathcal{O}(n+m)$.

B Experimental Setup

B.1 Hardware Configuration

All experiments were conducted on a high-performance computing system with the following specifications:

- **CPU**: 96 logical cores
- **RAM**: 512 GB
- **GPU**: NVIDIA A100-PCIE-40GB

B.2 Software Configuration

The software environment for our experiments was configured as follows:

- **Operating System**: Linux (Ubuntu 20.04.4 LTS (GNU/Linux 5.4.0-124-generic x86_64)))
- **PyTorch Version**: 1.13.1+cu117
- **CUDA Version**: 11.7
- **PyTorch Geometric Version**: 2.3.1

B.3 Additional Experimental Parameters

- **Number of layers in evaluation models**: 2 (with Relu in between) for Gcn, Gat, and Gin. The Mlp used in Gin is a simple linear transform with a bias defined by the following equation $\mathbf{WX}+\mathbf{b}$ where $\mathbf{X}$ is the input design matrix.
- **Value of k in Rev-k-NN**: 5
- **Hyper-parameters Baselines:** We use the config files shared by the authors. We note that the benchmark datasets are common between our experiments and those used in the baselines.

B.4 Additional Experiments

Table 8: Component-wise analysis of Bonsai's running times in seconds.

Dataset	WL	Rev-k-NN	Greedy	PPR	Full
Cora	0.87	0.05	0.01	1.68	2.61
Citeseer	0.97	0.07	0.02	1.71	2.77
Pubmed	8.89	1.39	0.64	13.93	24.85
Reddit	682.80	293.15	275.52	193.99	1445.45

B.4.1 Running time analysis of Bonsai

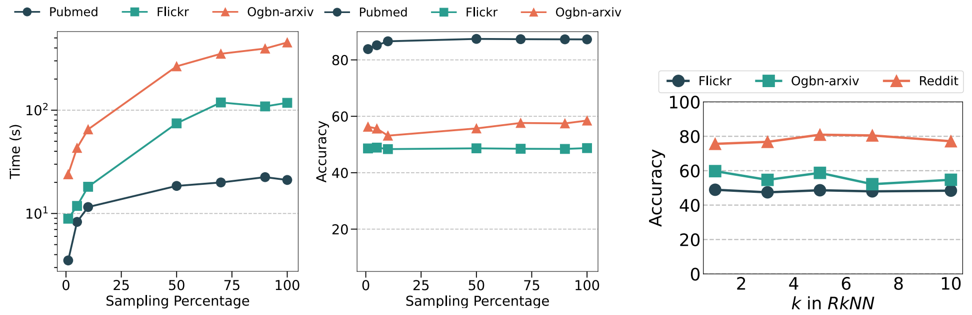
Fig. 8 presents the time consumed by the various components within Bonsai. We observe that most of the time is spent in PPR computation, except in Reddit, where Rev-k-NN is a dominant contributor. This trend can be explained by the high density of Reddit, where the average degree is 100 in contrast to < 5 in the other networks.

B.4.2 Impact of Parameters on Bonsai

Bonsai has two parameters, both related to the computation of Rev-k-NN. The value of k in Rev-k-NN and the sample size to accurately approximate the representative power.

Fig. 6a and Fig. 6b present the impact of sample size on accuracy and time, respectively. As expected, the general trend shows that increasing the sample size leads to higher accuracy and longer running times. However, we note that the increase in accuracy is minor. This observation is consistent with Lemma 1, which states that a small sample size is sufficient to achieve a high-quality approximation.

Fig. 6c presents the impact of k on accuracy. While the performance is generally stable against k, the peak accuracy is typically achieved when k is set to 5.

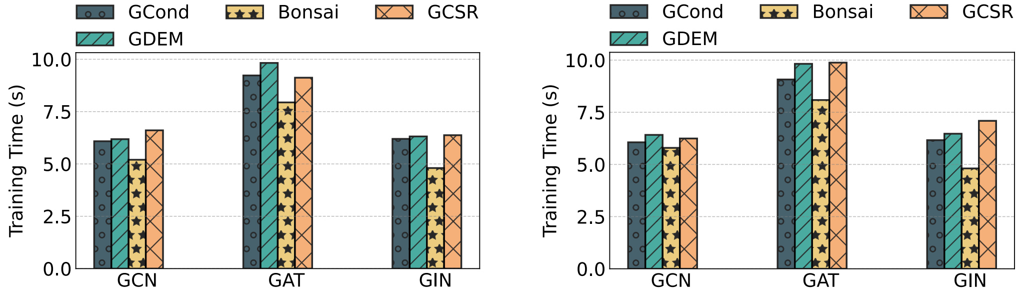


(a) Impact of sampling size on Accuracy

(b) Impact of sampling size on time

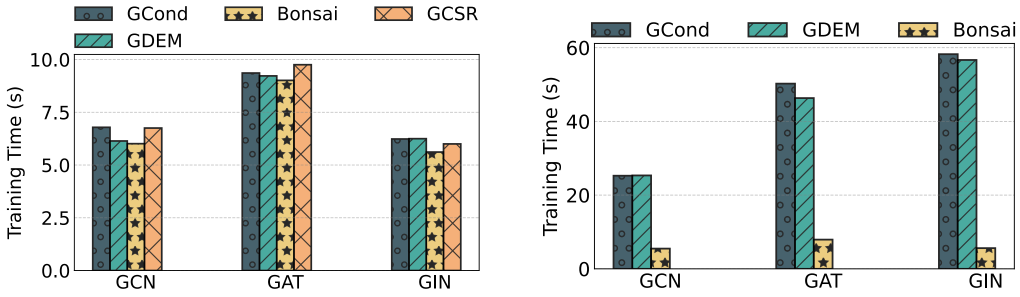
(c) Effect of varying k in the Rev-k-NN in Bonsai.

Figure 6: Impact of k in Rev-k-NN and sampling size for Rev-k-NN approximation on the performance of Bonsai.



(a) Cora

(b) Citeseer



(c) Pubmed

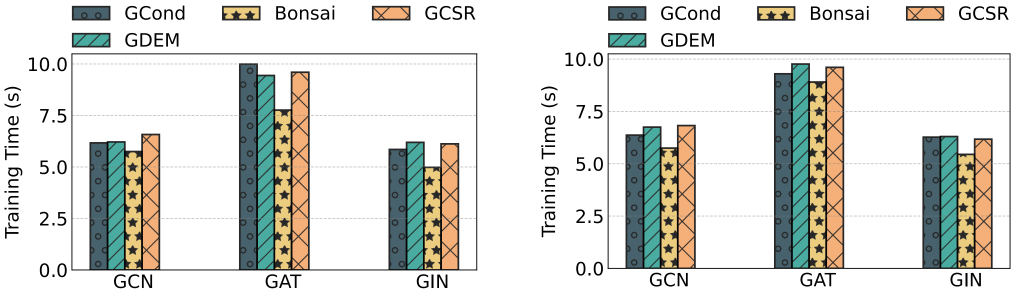
(d) Reddit

Figure 7: Comparing time required to train models on datasets distilled by GCond, Gdem, Bonsai, and Gcsr at 0.5% of Cora, Citeseer, Pubmed, and Reddit. Note that there is no distilled dataset for Reddit output by Gcsr due to OOT.

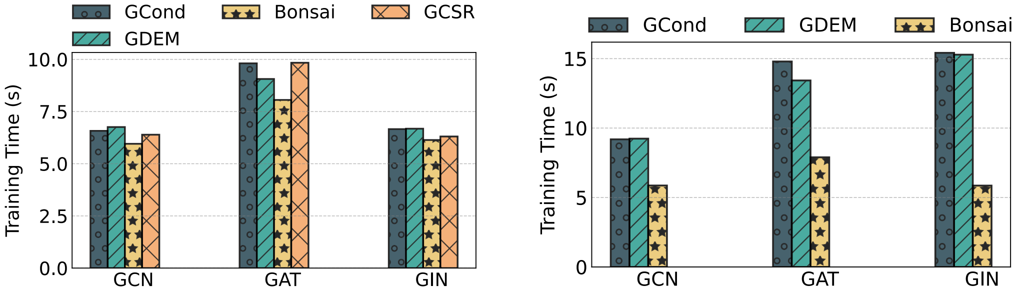
B.5 Additional Training Times

Fig. 7 presents the training times on distilled datasets in Cora, Citeseer, Pubmed, and Reddit at 0.5% size. Figs. 8 and 9 presents the training times on distilled datasets in Cora, Citeseer, Pubmed, Flickr, Ogbn-arxiv, and Reddit at 1%, and 3% sizes. Bonsai is faster on all architectures. The gap between Bonsai and the baselines increase with increase in distilled dataset size. This is more pronounced in larger datasets like Flickr, Ogbn-arxiv, and Reddit because the number of edges is $O(|V|^2)$ for the baselines and forward pass through Gnns is usually $O(|E|)$. Additionally, the distilled dataset created by GCond and Gdem for Reddit at 3% size cannot be used to train basic PyTorch Geometric models for Gat and Gin, resulting in out-of-memory (OOM) errors because the backward pass graph maintained by PyTorch is larger.

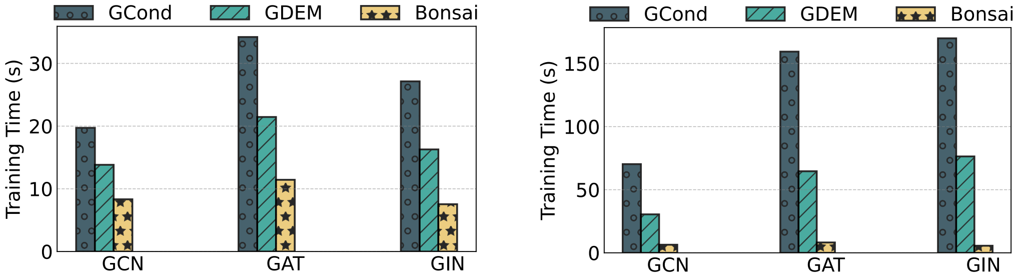
B.6 Carbon Emissions



(a) Cora (b) Citeseer

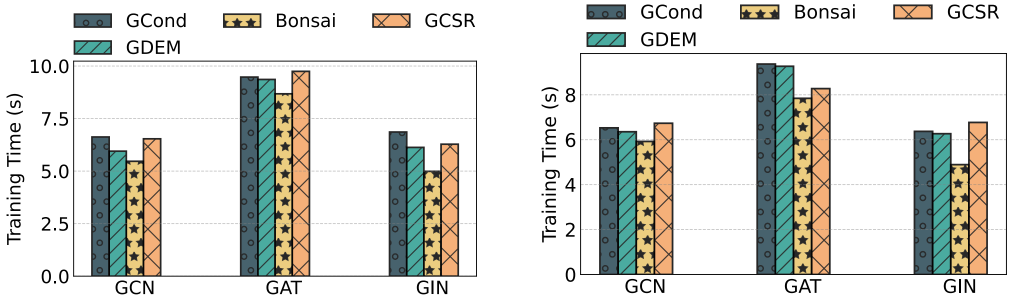


(c) Pubmed (d) Flickr

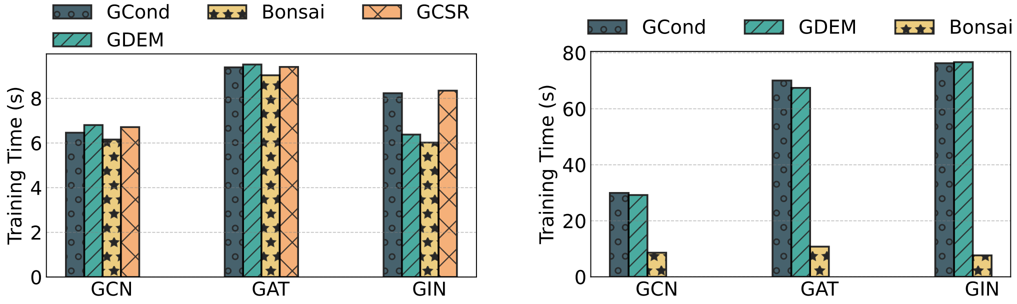


(e) Ogbn-arxiv (f) Reddit

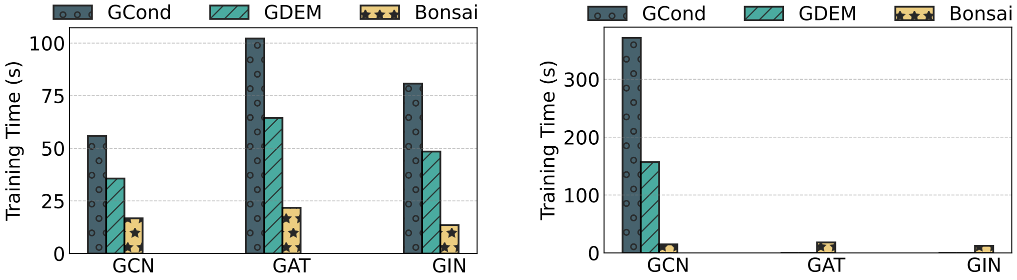
Figure 8: Comparing time required to train models on datasets distilled by GCOND, GDEM, BONSAI, and GCSR at 1% of Cora, Citeseer, Pubmed, Flickr, Ogbn-arxiv, and Reddit. Note that there is no distilled dataset for Flickr, Ogbn-arxiv, Reddit output by GCSR due to OOT.



(a) Cora (b) Citeseer



(c) Pubmed (d) Flickr



(e) Ogbn-arxiv (f) Reddit

Figure 9: Comparing time required to train models on datasets distilled by GCOND, GDEM, BONSAI, and GCSR at 3% of Cora, Citeseer, Pubmed, Flickr, Ogbn-arxiv, and Reddit. Note that there is no distilled dataset for Flickr, Ogbn-arxiv, Reddit output by GCSR due to OOT; and distilled datasets produced by GCOND and GDEM for Reddit cannot train on standard 2-layered PyTorch Geometric GAT and GIN due to OOM.

Table 9: Estimated CO_2 emissions from distillation of various methods in seconds at 0.5%. Emissions that are higher than training on the full dataset itself are highlighted in red. The least emission from distillation for each dataset is highlighted in green. CO_2 emissions are computed as 10.8kg per 100 hours for Nvidia A100 GPU and 4.32kg per 100 hours for 10 CPUs of Intel Xeon Gold 6248 (Lacoste et al., 2019).

Dataset	**GCOND**	**GDEM**	**GCSR**	**EXGC**	**GC-SNTK**	**GEOM**	BONSAI	**Full**
Cora	82.14	3.15	157.80	1.05	2.46	389.88	0.03	0.75
Citeseer	81.36	5.01	199.08	1.03	3.72	472.89	0.03	0.75
Pubmed	77.01	15.90	39.57	3.45	3.51	$\geq$540.00	0.3	1.53
Flickr	58.05	102.15	523.35	7.30	18.36	$\geq$540.00	1.42	5.40
Ogbn-arxiv	434.22	17.07	$\geq$540.00	46.49	366.54	$\geq$540.00	4.18	15.74
Reddit	903.36	602.94	$\geq$540.00	207.10	876.33	$\geq$540.00	17.34	72.77