Graph Distillation with Eigenbasis Matching

Yang Liu [*1] **Deyu Bo** [*1] **Chuan Shi** [1]

Abstract

The increasing amount of graph data places requirements on the efficient training of graph neural networks (GNNs). The emerging graph distillation (GD) tackles this challenge by distilling a small synthetic graph to replace the real large graph, ensuring GNNs trained on real and synthetic graphs exhibit comparable performance. However, existing methods rely on GNN-related information as supervision, including gradients, representations, and trajectories, which have two limitations. First, GNNs can affect the spectrum (*i.e.*, eigenvalues) of the real graph, causing *spectrum bias* in the synthetic graph. Second, the variety of GNN architectures leads to the creation of different synthetic graphs, requiring *traversal* to obtain optimal performance. To tackle these issues, we propose Graph Distillation with Eigenbasis Matching (GDEM), which aligns the eigenbasis and node features of real and synthetic graphs. Meanwhile, it directly replicates the spectrum of the real graph and thus prevents the influence of GNNs. Moreover, we design a discrimination constraint to balance the effectiveness and generalization of GDEM. Theoretically, the synthetic graphs distilled by GDEM are restricted spectral approximations of the real graphs. Extensive experiments demonstrate that GDEM outperforms state-of-the-art GD methods with powerful cross-architecture generalization ability and significant distillation efficiency. Our code is available at https://github.com/liuyang-tian/GDEM.

1. Introduction

Graph neural networks (GNNs) are proven effective in a variety of graph-related tasks (Kipf & Welling, 2017; Velickovic et al., 2018). However, the non-Euclidean nature of graph structure presents challenges to the efficiency and scalability of GNNs (Hamilton et al., 2017). To accelerate training, one data-centric approach is to summarize the large-scale graph into a much smaller one. Traditional methods primarily involve sparsification (Spielman & Srivastava, 2011; Yu et al., 2022) and coarsening (Loukas, 2019; Kumar et al., 2023). However, these methods are typically designed to optimize some heuristic metrics, *e.g.*, spectral similarity (Loukas, 2019) and pair-wise distance (Ahmed et al., 2020), which may be irrelevant to downstream tasks, leading to sub-optimal performance.

Recently, graph distillation (GD), *a.k.a.*, graph condensation, has attracted considerable attention in graph reduction due to its remarkable compression ratio and lossless performance (Gao et al., 2024). Generally, GD aims to synthesize a small graph wherein GNNs trained on it exhibit comparable performance to those trained on the real large graph. To this end, existing methods are designed to optimize the synthetic graphs by matching some GNN-related information, such as gradients (Jin et al., 2022b;a), representations (Liu et al., 2022a), and training trajectories (Zheng et al., 2023), between the real and synthetic graphs. As a result, the synthetic graph aligns its distribution with the real graph and also incorporates information from downstream tasks.

Despite the considerable progress, existing GD methods require pre-selecting a specific GNN as the distillation model, introducing two limitations: (1) GNNs used for distillation affect the real spectrum, leading to spectrum bias in the synthetic graph, *i.e.*, a few eigenvalues dominate the data distribution. Figure 1 illustrates the total variation (TV) (Gutman & Zhou, 2006) of the real and synthetic graphs. Notably, TV reflects the smoothness of the signal over a graph. A small value of TV indicates a low-frequency distribution, and vice versa. We can observe that the values of TV in the synthetic graph distilled by a low-pass filter consistently appear lower than those in the real graph, while the opposite holds for the high-pass filter, thus verifying the existence of spectrum bias (See Section 3 for a theoretical analysis). (2) The optimal performance is obtained by traversing various GNN architectures, resulting in non-negligible computational costs. Table 1 presents the cross-architecture results of GCOND (Jin et al., 2022b) across six well-known GNNs, including GCN (Kipf & Welling, 2017), SGC (Wu et al., 2019), PPNP (Klicpera et al., 2019), ChebyNet (Def-

[*]Equal contribution [1]Department of Computer Science, Beijing University of Posts and Telecommunication, Beijing, China. Correspondence to: Chuan Shi <shichuan@bupt.edu.cn>.

Proceedings of the 41st International Conference on Machine Learning, Vienna, Austria. PMLR 235, 2024. Copyright 2024 by the author(s).

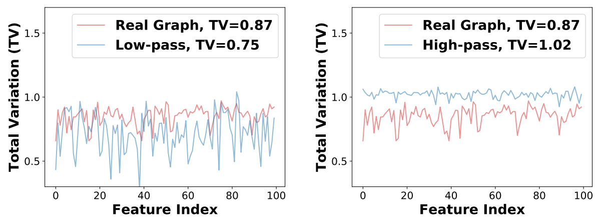


Figure 1. Data distribution of the real and synthetic graphs in Pubmed dataset, where the average TV of the real graph is 0.87. **Left**: Synthetic graph distilled by a low-pass filter has a lower value of TV (0.75). **Right**: Synthetic graph distilled by a high-pass filter has a higher value of TV (1.02). For clarity, only the first 100-dimensional features are visualized. Best viewed in color.

Table 1. Cross-architecture performance (%) of GCOND with various distillation (D) and evaluation (E) GNNs in Pubmed dataset. **Bold** indicates the best in each column.

D \ E	GCN	SGC	PPNP	Cheb.	Bern.	GPR.
GCN	74.57	71.70	75.53	70.13	68.40	71.73
SGC	**77.72**	**77.60**	77.34	76.03	74.42	76.52
PPNP	72.70	70.40	77.46	73.38	70.56	74.02
Cheb.	73.60	70.62	75.10	**77.30**	77.62	78.10
Bern.	67.68	73.76	74.30	77.20	**78.12**	**78.28**
GPR.	76.04	72.20	**77.94**	75.92	77.12	77.96

ferrard et al., 2016), BernNet (He et al., 2021), and GPR-GNN (Chien et al., 2021). It can be seen that the evaluation performance of different GNNs varies greatly. As a result, existing GD methods need to distill and traverse various GNN architectures to obtain optimal performance, which significantly improves the time overhead. See Appendix A.1 for the definition of TV and Appendix A.2 for more experimental details.

Once the weaknesses of existing methods are identified, it is natural to ask: *How to distill graphs without being affected by different GNNs?* To answer this question, we propose Graph Distillation with Eigenbasis Matching (GDEM). Specifically, GDEM decomposes the graph structure into eigenvalues and eigenbasis. During distillation, GDEM matches the eigenbasis and node features of real and synthetic graphs, which equally preserves the information of different frequencies, thus addressing the spectrum bias. Additionally, a discrimination loss is jointly optimized to improve the performance of GDEM and balance its effectiveness and generalization. Upon completing the matching, GDEM leverages the real graph spectrum and synthetic eigenbasis to construct a complete synthetic graph, which prevents the spectrum from being affected by GNNs and ensures the uniqueness of the synthetic graph, thus avoiding the traversal requirement and improving the distillation efficiency.

The contributions of our paper are as follows. (1) We systematically analyze the limitations of existing distillation methods, including spectrum bias and traversal requirement. (2) We propose GDEM, a novel graph distillation framework, which mitigates the dependence on GNNs by matching the eigenbasis instead of the entire graph structure. Additionally, it is theoretically demonstrated that GDEM preserves essential spectral similarity during distillation. (3) Extensive experiments on seven graph datasets validate the superiority of GDEM over state-of-the-art GD methods in terms of effectiveness, generalization, and efficiency.

2. Preliminary

Before describing our framework in detail, we first introduce some notations and concepts used in this paper. Specifically, we focus on the node classification task, where the goal is to predict the labels of the nodes in a graph. Assume that there is a graph $\mathcal{G} = (\mathcal{V}, \mathcal{E}, \mathbf{X})$, where $\mathcal{V}$ is the set of nodes with $|\mathcal{V}| = N$, $\mathcal{E}$ indicates the set of edges, and $\mathbf{X} \in \mathbb{R}^{N \times d}$ is the node feature matrix. The adjacency matrix of $\mathcal{G}$ is defined as $\mathbf{A} \in \{0,1\}^{N \times N}$, where $A_{ij} = 1$ if there is an edge between nodes i and j, and $A_{ij} = 0$ otherwise. The corresponding normalized Laplacian matrix is defined as $\mathbf{L} = \mathbf{I}_N - \mathbf{D}^{-\frac{1}{2}}\mathbf{A}\mathbf{D}^{-\frac{1}{2}}$, where $\mathbf{I}_N$ is an identity matrix and $\mathbf{D}$ is the degree matrix with $D_{ii} = \sum_j A_{ij}$ for $i \in \mathcal{V}$ and $D_{ij} = 0$ for $i \neq j$. Without loss of generality, we assume that $\mathcal{G}$ is undirected and all the nodes are connected.

Eigenbasis and Eigenvalue. The normalized graph Laplacian can be decomposed as $\mathbf{L} = \mathbf{U}\mathbf{\Lambda}\mathbf{U}^\top = \sum_{i=1}^{N} \lambda_i \mathbf{u}_i \mathbf{u}_i^\top$, where $\mathbf{\Lambda} = \mathrm{diag}(\{\lambda_i\}_{i=1}^{N})$ are the eigenvalues and $\mathbf{U} = [\mathbf{u}_1, \cdots, \mathbf{u}_N] \in \mathbb{R}^{N \times N}$ is the eigenbasis, consisting of a set of eigenvectors. Each eigenvector $\mathbf{u}_i \in \mathbb{R}^N$ has a corresponding eigenvalue λ_i, such that $\mathbf{L}\mathbf{u}_i = \lambda_i \mathbf{u}_i$. Without loss of generality, we assume $0 \leq \lambda_1 \leq \cdots \leq \lambda_N \leq 2$.

Graph Distillation. GD aims to distill a small synthetic graph $\mathcal{G}' = (\mathcal{V}', \mathcal{E}', \mathbf{X}')$, where $|\mathcal{V}'| = N' \ll N$ and $\mathbf{X}' \in \mathbb{R}^{N' \times d}$, from the real large graph $\mathcal{G}$. Meanwhile, GNNs trained on $\mathcal{G}$ and $\mathcal{G}'$ will have comparable performance, thus accelerating the training of GNNs. Existing frameworks can be divided into three categories: gradient matching, distribution matching, and trajectory matching. See Appendix C for more detailed descriptions.

3. Spectrum Bias in Gradient Matching

In this section, we give a detailed analysis of the objective of gradient matching in graph data, which motivates the design of our method. We start with a vanilla example, which adopts a one-layer GCN as the distillation model and simplifies the objective of GNNs into the MSE loss:

$$\mathcal{L} = \frac{1}{2}\left\|\mathbf{A}\mathbf{X}\mathbf{W} - \mathbf{Y}\right\|_F^2, \tag{1}$$

where $\mathbf{W}$ is the model parameter. The gradients on the real and synthetic graphs are calculated as follows:

$$\begin{aligned} \nabla_{\mathbf{W}} &= (\mathbf{A}\mathbf{X})^T (\mathbf{A}\mathbf{X}\mathbf{W} - \mathbf{Y}), \\ \nabla'_{\mathbf{W}} &= (\mathbf{A}'\mathbf{X}')^T (\mathbf{A}'\mathbf{X}'\mathbf{W} - \mathbf{Y}'). \end{aligned} \tag{2}$$

Assume that the objective of gradient matching is the MSE loss between two gradients, *i.e.*, $\mathcal{L}_{GM} = \|\nabla_{\mathbf{W}} - \nabla'_{\mathbf{W}}\|_F^2$. To further characterize its properties, we analyze the following upper-bound of $\mathcal{L}_{GM}$:

$$\begin{aligned} \mathcal{L}_{GM} \leq & \|\mathbf{W}\|_F^2 \|\mathbf{X}^\top \mathbf{A}^2 \mathbf{X} - \mathbf{X}'^\top \mathbf{A}'^2 \mathbf{X}'\|_F^2 \\ & + \|\mathbf{X}^\top \mathbf{A}\mathbf{Y} - \mathbf{X}'^\top \mathbf{A}'\mathbf{Y}'\|_F^2, \end{aligned} \tag{3}$$

where $\mathbf{X}^\top \mathbf{A}^2 \mathbf{X}$ and $\mathbf{X}^\top \mathbf{A}\mathbf{Y}$ are two target distributions in the real graph, which are used to supervise the update of the synthetic graph. However, both of them will be dominated by a few eigenvalues, resulting in spectrum bias.

Lemma 3.1. *The target distribution of GCN is dominated by the smallest eigenvalue after stacking multiple layers.*

Proof. The target distribution can be reformulated as:

$$\mathbf{X}^\top \mathbf{A}^{2t} \mathbf{X} = \sum_{i=1}^{N} (1 - \lambda_i)^{2t} \mathbf{X}^\top \mathbf{u}_i \mathbf{u}_i^\top \mathbf{X}, \tag{4}$$

where t is the number of layers. When t goes to infinity, only the smallest eigenvalue $\lambda_0 = 0$ preserves its coefficient $(1 - \lambda_0)^{2t} = 1$ and other coefficients tend to 0. Hence, the target distribution $\mathbf{X}^\top \mathbf{A}^{2t} \mathbf{X}$ is dominated by $\mathbf{X}^\top \mathbf{u}_0 \mathbf{u}_0^\top \mathbf{X}$. The same analysis can be applied for $\mathbf{X}^\top \mathbf{A}^t \mathbf{Y}$. □

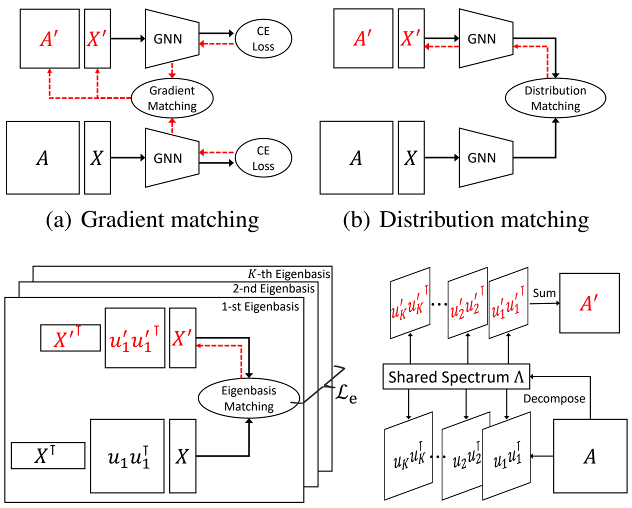
Lemma 3.2. *Suppose the distillation GNN has an analytic filtering function $g(\cdot)$. Then the target distributions will be dominated by the eigenvalues whose filtered values are greater than 1,* i.e., $g(\lambda_i) \geq 1$.

Proof. The objective function of distillation GNN is $\mathcal{L} = \frac{1}{2} \|g(\mathbf{L})\mathbf{X}\mathbf{W} - \mathbf{Y}\|_F^2$. Then the target distributions become $\mathbf{X}^\top g(\mathbf{L})^{2t} \mathbf{X}$ and $\mathbf{X}^\top g(\mathbf{L})^t \mathbf{Y}$ as g is analytic. Therefore, the filtered eigenvalues with values $g(\lambda_i) \geq 1$ retain their coefficients and dominate the target distributions. □

Lemmas 3.1 and 3.2 state that leveraging the information of GNNs in distillation will introduce a spectral bias in the target distributions. As a result, the synthetic graph can only match part of the data distribution of the real graph, leaving its structural information incomplete.

4. The Proposed Method: GDEM

In this section, we introduce the proposed method GDEM. Compared with previous methods, *e.g.*, gradient matching (Figure 2(a)) and distribution matching (Figure 2(b)), GDEM, illustrated in 2(c), does not rely on specific GNNs, whose distillation process can be divided into two steps: (1) Matching the eigenbasis and node features between the real and synthetic graphs. (2) Constructing the synthetic graph by using the synthesized eigenbasis and real spectrum.



(a) Gradient matching (b) Distribution matching

(c) Eigenbasis matching

Figure 2. Comparison between different graph distillation methods, where the red characters represent the synthetic data, the solid black lines, and red dotted lines indicate the forward and backward passes, respectively.

4.1. Eigenbasis Matching

The eigenbasis of a graph represents its crucial structural information. For example, eigenvectors corresponding to smaller eigenvalues reflect the global community structure, while eigenvectors corresponding to larger eigenvalues encode local details (Bo et al., 2021). Generally, the number of eigenvectors is the same as the number of nodes in a graph, suggesting that we cannot preserve all the real eigenbasis in the synthetic graph. Therefore, GDEM is designed to match eigenvectors with the K_1 smallest and the K_2 largest eigenvalues, where K_1 and K_2 are hyper-parameters, and $K_1 + K_2 = K \leq N'$. This approach has been proven effective in both graph coarsening (Jin et al., 2020) and spectral GNNs (Bo et al., 2023). We initialize a matrix $\mathbf{U}'_K = [\mathbf{u}'_1, \cdots, \mathbf{u}'_{N'}] \in \mathbb{R}^{N' \times K}$ to match the principal eigenbasis of the real graph, denoted as $\mathbf{U}_K = [\mathbf{u}_1, \cdots, \mathbf{u}_{K_1}, \mathbf{u}_{N-K_2}, \cdots, \mathbf{u}_N] \in \mathbb{R}^{N \times K}$.

To eliminate the influence of GNNs, GDEM does not use the spectrum information during distillation. Therefore, the first term in Equation 3 becomes:

$$\mathcal{L}_e = \sum_{k=1}^{K} \left\| \mathbf{X}^\top \mathbf{u}_k \mathbf{u}_k^\top \mathbf{X} - \mathbf{X}'^\top \mathbf{u}'_k \mathbf{u}'^\top_k \mathbf{X}' \right\|_F^2, \tag{5}$$

where $\mathbf{u}_k \mathbf{u}_k^\top$ and $\mathbf{u}'_k \mathbf{u}'^\top_k$ are the subspaces induced by the k-th eigenvector in the real and synthetic graphs.

Additionally, as the basis of graph Fourier transform, eigenvectors are naturally normalized and orthogonal to each other. However, directly optimizing $\mathbf{U}'_K$ via gradient descent cannot preserve this property. Therefore, an additional regularization is used to constrain the representation space:

$$\mathcal{L}_o = \left\| \mathbf{U}_K'^{\top} \mathbf{U}_K' - \mathbf{I}_K \right\|_F^2 . \tag{6}$$

See Appendix A.3 for more implementation details.

4.2. Discrimination Constraint

In practice, we find that eigenbasis matching improves the cross-architecture generalization of GDEM but contributes less to the performance of node classification as it only preserves the global distribution, *i.e.*, $\mathbf{X}^\top \mathbf{u}\mathbf{u}^\top \mathbf{X}$, without considering the information of downstream tasks. Therefore, we need to approximate the second term in Equation 3. Interestingly, we find that $\mathbf{X}^\top \mathbf{A}\mathbf{Y} \in \mathbb{R}^{d\times C}$ indicates the category-level representations, which assigns each category a d-dimensional representation. However, the MSE loss only emphasizes the intra-class similarity between the real and synthetic graphs and ignores the inter-class dissimilarity.

Based on this discovery, we design a discrimination constraint to effectively preserve the category-level information, which can also be treated as a class-aware regularization technique (Zhao et al., 2023; Wang et al., 2022). Specifically, we first learn the category-level representations of the real and synthetic graphs:

$$\mathbf{H} = \mathbf{Y}^\top \mathbf{A}\mathbf{X}, \quad \mathbf{H}' = \mathbf{Y}'^\top \sum_{k=1}^{K} (1-\lambda_k)\mathbf{u}_k' \mathbf{u}_k'^\top \mathbf{X}', \tag{7}$$

where λ_k is the k-th eigenvalue of the real graph Laplacian. We then constrain the cosine similarity between $\mathbf{H}$ and $\mathbf{H}'$:

$$\mathcal{L}_d = \sum_{i=1}^{C} \left(1 - \frac{\mathbf{H}_i^\top \cdot \mathbf{H}_i'}{||\mathbf{H}_i||\,||\mathbf{H}_i'||}\right) + \sum_{\substack{i,j=1\\ i\neq j}}^{C} \frac{\mathbf{H}_i^\top \cdot \mathbf{H}_j'}{||\mathbf{H}_i||\,||\mathbf{H}_j'||}. \tag{8}$$

Note that the discrimination constraint introduces the spectrum information in the distillation process, which conflicts with the eigenbasis matching. However, we find that adjusting the weights of eigenbasis matching and the discrimination constraint can balance the performance and generalization of GDEM. Ablation studies can be seen in Section 6.5.

4.3. Final Objective and Synthetic Graph Construction

In summary, the overall loss function of GDEM is formulated as the weighted sum of three regularization terms:

$$\mathcal{L}_{total} = \alpha\mathcal{L}_e + \beta\mathcal{L}_d + \gamma\mathcal{L}_o, \tag{9}$$

where α, β, and γ are the hyperparameters. The pseudo-code of GDEM is presented in Algorithm 1.

Upon minimizing the total loss function, the outputs of GDEM are the eigenbasis and node features of the synthetic graph. However, the data remains incomplete due to the absence of the graph spectrum. Essentially, the graph spectrum encodes the global shape of a graph (Martinkus et al., 2022). Ideally, if the synthetic graph preserves the distribution of the real graph, they should have similar spectrums. Therefore, we directly replicate the real spectrum for the synthetic graph to construct its Laplacian matrix or adjacency matrix:

$$\mathbf{L}' = \sum_{k=1}^{K} \lambda_k \mathbf{u}_k' \mathbf{u}_k'^\top, \quad \mathbf{A}' = \sum_{k=1}^{K} (1-\lambda_k)\mathbf{u}_k' \mathbf{u}_k'^\top. \tag{10}$$

Algorithm 1 GDEM for Graph Distillation

```
Input: Real graph $\mathcal{G} = (\mathbf{A}, \mathbf{X}, \mathbf{Y})$ with eigenvalues $\{\lambda_i\}_{i=1}^K$ and eigenbasis $\mathbf{U}_K$
Init: Synthetic graph $\mathcal{G}'$ with eigenbasis $\mathbf{U}_K'$, node features $\mathbf{X}'$, and labels $\mathbf{Y}'$
for $t = 1$ to $T$ do
    Compute $\mathcal{L}_e$, $\mathcal{L}_o$, and $\mathcal{L}_d$ via Eqs. 5, 6, and 8
    Compute $\mathcal{L}_{total} = \alpha\mathcal{L}_e + \beta\mathcal{L}_d + \gamma\mathcal{L}_o$
    if $t\%(\tau_1 + \tau_2) < \tau_1$ then
        Update $\mathbf{U}_K' \leftarrow \mathbf{U}_K' - \eta_1 \nabla_{\mathbf{U}_K'} \mathcal{L}_{total}$
    else
        Update $\mathbf{X}' \leftarrow \mathbf{X}' - \eta_2 \nabla_{\mathbf{X}'} \mathcal{L}_{total}$
    end if
end for
Compute $\mathbf{A}' = \sum_{k=1}^{K} (1-\lambda_k)\mathbf{u}_k' \mathbf{u}_k'^\top$
Return: $\mathbf{A}'$, $\mathbf{X}'$
```

4.4. Discussion

Complexity. The complexity of decomposition is $\mathcal{O}(N^3)$. However, given that we only utilize the K smallest or largest eigenvalues, the complexity reduces to $\mathcal{O}(KN^2)$. Additionally, $\mathbf{u}_k^\top \mathbf{X}$ in Equation 5 and $\mathbf{H}$ in Equation 8 cost $\mathcal{O}(KNd)$ and $\mathcal{O}(Ed)$ in pre-processing. During distillation, the complexity of $\mathcal{L}_e$, $\mathcal{L}_d$ and $\mathcal{L}_o$ are $\mathcal{O}(KN'd + Kd^2)$, $\mathcal{O}(KN'd' + Cd^2)$, and $\mathcal{O}(KN'^2)$, respectively.

Relation to Message Passing. Message-passing (MP) is the most popular paradigm for GNNs. Although GDEM does not explicitly perform message-passing during distillation, eigenbasis matching already encodes the information of neighbors as most MP operators rely on the combination of the out product of eigenvectors, *e.g.*, $\mathbf{L} = \sum_{i=1}^{N} \lambda_i \mathbf{u}_i \mathbf{u}_i^\top$. Therefore, GDEM not only inherits the expressive power of MP but also addresses the weaknesses of the previous distillation methods.

Limitations. Hereby we discuss the limitations of GDEM. (1) The decomposition of the real graph introduces additional computational costs for distillation. (2) In scenarios with extremely high compression rates, the synthetic graphs can only match a limited number of real eigenbasis, resulting in performance degradation.

5. Theoretical Analysis

In this section, we give a theoretical analysis of GDEM and prove that it preserves the restricted spectral similarity.

Definition 5.1. (Spectral Similarity (Spielman & Srivastava, 2011)) Let $\mathbf{A}, \mathbf{B} \in \mathbb{R}^{N \times N}$ be two square matrices. Matrix $\mathbf{B}$ is considered a spectral approximation of $\mathbf{A}$ if there exists a positive constant ϵ, such that for any vector $\mathbf{x} \in \mathbb{R}^N$, the following inequality holds:

$$(1-\epsilon)\mathbf{x}^\top \mathbf{A}\mathbf{x} < \mathbf{x}^\top \mathbf{B}\mathbf{x} < (1+\epsilon)\mathbf{x}^\top \mathbf{A}\mathbf{x}.$$

However, it is impossible to satisfy this condition for all $\mathbf{x} \in \mathbb{R}^N$ (Loukas, 2019). Therefore, we only consider a restricted version of spectral similarity in the feature space.

Definition 5.2. (Restricted Spectral Similarity, RSS [1]) The synthetic graph Laplacian $\mathbf{L}'$ preserves RSS of the real graph Laplacian $\mathbf{L}$, if there exists an $\epsilon > 0$ such that:

$$(1-\epsilon)\mathbf{x}^\top \mathbf{L}\mathbf{x} < \mathbf{x}'^\top \mathbf{L}'\mathbf{x}' < (1+\epsilon)\mathbf{x}^\top \mathbf{L}\mathbf{x} \quad \forall \mathbf{x}, \mathbf{x}' \in \mathbf{X}, \mathbf{X}'.$$

Proposition 5.3. *The synthetic graph distilled by GDEM is a restricted ϵ-spectral approximation of the real graph.*

Proof. We first characterize the spectral similarity of node features in the real and synthetic graphs, respectively. Notably, here we use the principal K eigenvalues and eigenvectors as a truncated representation of the real graph

$$\mathbf{x}^\top \mathbf{L}\mathbf{x} = \mathbf{x}^\top \sum_{k=1}^{N} \lambda_k \mathbf{u}_k \mathbf{u}_k^\top \mathbf{x} \approx \sum_{k=1}^{K} \lambda_k \mathbf{x}^\top \mathbf{u}_k \mathbf{u}_k^\top \mathbf{x}, \quad (11)$$

$$\begin{aligned}\mathbf{x}'^\top \mathbf{L}'\mathbf{x}' &= \mathbf{x}'^\top (\sum_{k=1}^{N'} \lambda_k \mathbf{u}'_k \mathbf{u}'^\top_k + \tilde{\mathbf{U}}\mathbf{\Lambda}\tilde{\mathbf{U}}^\top)\mathbf{x}' \\ &\approx \sum_{k=1}^{K} \lambda_k \mathbf{x}'^\top \mathbf{u}'_k \mathbf{u}'^\top_k \mathbf{x}' + \Delta, \quad (12)\end{aligned}$$

where $\Delta = \mathbf{x}'^\top \tilde{\mathbf{U}}\mathbf{\Lambda}\tilde{\mathbf{U}}^\top \mathbf{x}'$ and $\tilde{\mathbf{U}}$ represents the non-orthogonal terms of the eigenbasis $\mathbf{U}'_K$, which means that $\mathbf{U}'_K + \tilde{\mathbf{U}}$ is strictly orthogonal.

Combining Equations 11 and 12, we have

$$\begin{aligned}&\left|\mathbf{x}^\top \mathbf{L}\mathbf{x} - \mathbf{x}'^\top \mathbf{L}'\mathbf{x}'\right| \\ \approx &\left|\sum_{k=1}^{K} \lambda_k \mathbf{x}^\top \mathbf{u}_k \mathbf{u}_k^\top \mathbf{x} - \sum_{k=1}^{K} \lambda_k \mathbf{x}'^\top \mathbf{u}'_k \mathbf{u}'^\top_k \mathbf{x}' - \Delta\right| \quad (13) \\ \leq &\underbrace{\sum_{k=1}^{K} \lambda_k \left|\mathbf{x}^\top \mathbf{u}_k \mathbf{u}_k^\top \mathbf{x} - \mathbf{x}'^\top \mathbf{u}'_k \mathbf{u}'^\top_k \mathbf{x}'\right|}_{\mathcal{L}_e} + \underbrace{|\Delta|}_{\mathcal{L}_o}.\end{aligned}$$

The above inequality shows that the objective of eigenbasis matching is the upper bound of the spectral discrepancy between the real and synthetic graphs. Optimizing $\mathcal{L}_e$ and $\mathcal{L}_o$ makes the bound tighter and preserves the spectral similarity of the real graph. The synthetic graph is a restricted ϵ-spectral approximation of the real graph with $\epsilon = \sum_{k=1}^{K} \lambda_k \left|\mathbf{x}^\top \mathbf{u}_k \mathbf{u}_k^\top \mathbf{x} - \mathbf{x}'^\top \mathbf{u}'_k \mathbf{u}'^\top_k \mathbf{x}'\right| + |\Delta|$. □

[1] RSS defined in this paper is different from Loukas (2019), which limits the signal $\mathbf{x}$ in the eigenvector space.

6. Experiments

In this section, we conduct experiments on a variety of graph datasets to validate the effectiveness, generalization, and efficiency of the proposed GDEM.

6.1. Experimental Setup

Datasets. To evaluate the effectiveness of our GDEM, we select seven representative graph datasets, including five homophilic graphs, *i.e.*, Citeseer, Pubmed (Kipf & Welling, 2017), Ogbn-arxiv (Hu et al., 2020), Filckr (Zeng et al., 2020), and Reddit (Hamilton et al., 2017), and two heterophilic graphs, *i.e.*, Squirrel (Rozemberczki et al., 2021) and Gamers (Lim et al., 2021).

Baselines. We benchmark our model against several competitive baselines, which can be divided into two categories: (1) Traditional graph reduction methods, including three coreset methods, *i.e.*, Random, Herding, and K-Center (Welling, 2009; Sener & Savarese, 2018), and one coarsening method (Loukas, 2019). (2) Graph distillation methods, including two gradient matching methods, *i.e.*, GCOND (Jin et al., 2022b) and SGDD (Yang et al., 2023), and one trajectory matching method, *i.e.*, SFGC (Zheng et al., 2023). See Appendix A.6 for more details.

Evaluation Protocol. To fairly evaluate the quality of synthetic graphs, we perform the following two steps for all methods: (1) **Distillation step**, where we apply the distillation methods in the training set of the real graphs. (2) **Evaluation step**, where we train GNNs on the synthetic graph from scratch and then evaluate their performance on the test set of real graphs. In the node classification experiment (Section 6.2), we follow the settings of the original papers (Jin et al., 2022b; Zheng et al., 2023; Yang et al., 2023). In the generalization experiment (Section 6.3), we use six representative GNNs, including three spatial GNNs, *i.e.*, GCN, SGC, and PPNP, and three spectral GNNs, *i.e.*, ChebyNet, BernNet, and GPR-GNN. See Appendix A.7 for more detailed description.

Settings and Hyperparameters. To eliminate randomness, in the distillation step, we run the distillation methods 10 times and yield 10 synthetic graphs. Moreover, we set $K_1 + K_2 = N'$. To reduce the tuning complexity, we treat $r_k = \{0.8, 0.85, 0.9, 0.95, 1.0\}$ as a hyperparameter and set $K_1 = r_k N'$, $K_2 = (1 - r_k)N'$ for eigenbasis matching. In the evaluation step, spatial GNNs have two aggregation layers and the polynomial order of spectral GNNs is set to 10. For more details, see Appendix A.8.

Table 2. Node classification performance of different distillation methods, mean accuracy (%) ± standard deviation. **Bold** indicates the best performance and underline means the runner-up.

Dataset	Ratio (r)	Traditional Methods				Graph Distillation Methods				Whole Dataset
		Random ($\mathbf{A}', \mathbf{X}'$)	Coarsening ($\mathbf{A}', \mathbf{X}'$)	Herding ($\mathbf{A}', \mathbf{X}'$)	K-Center ($\mathbf{A}', \mathbf{X}'$)	GCOND ($\mathbf{A}', \mathbf{X}'$)	SFGC ($\mathbf{X}'$)	SGDD ($\mathbf{A}', \mathbf{X}'$)	GDEM ($\mathbf{U}', \mathbf{X}'$)	
Citeseer	0.90%	54.4±4.4	52.2±0.4	57.1±1.5	52.4±2.8	70.5±1.2	71.4±0.5	69.5±0.4	**72.3±0.3**	71.7±0.1
	1.80%	64.2±1.7	59.0±0.5	66.7±1.0	64.3±1.0	70.6±0.9	72.4±0.4	70.2±0.8	**72.6±0.6**	
	3.60%	69.1±0.1	65.3±0.5	69.0±0.1	69.1±0.1	69.8±1.4	70.6±0.7	70.3±1.7	**72.6±0.5**	
Pubmed	0.08%	69.4±0.2	18.1±0.1	76.7±0.7	64.5±2.7	76.5±0.2	76.4±1.2	77.1±0.5	**77.7±0.7**	79.3±0.2
	0.15%	73.3±0.7	28.7±4.1	76.2±0.5	69.4±0.7	77.1±0.5	77.5±0.4	78.0±0.3	**78.4±1.8**	
	0.30%	77.8±0.3	42.8±4.1	78.0±0.5	**78.2±0.4**	77.9±0.4	77.9±0.3	77.5±0.5	**78.2±0.8**	
Ogbn-arxiv	0.05%	47.1±3.9	35.4±0.3	52.4±1.8	47.2±3.0	59.2±1.1	**65.5±0.7**	60.8±1.3	63.7±0.8	71.4±0.1
	0.25%	57.3±1.1	43.5±0.2	58.6±1.2	56.8±0.8	63.2±0.3	**66.1±0.4**	65.8±1.2	63.8±0.6	
	0.50%	60.0±0.9	50.4±0.1	60.4±0.8	60.3±0.4	64.0±0.4	**66.8±0.4**	66.3±0.7	64.1±0.3	
Flickr	0.10%	41.8±2.0	41.9±0.2	42.5±1.8	42.0±0.7	46.5±0.4	46.6±0.2	46.9±0.1	**49.9±0.8**	47.2±0.1
	0.50%	44.0±0.4	44.5±0.1	43.9±0.9	43.2±0.1	47.1±0.1	47.0±0.1	47.1±0.3	**49.4±1.3**	
	1.00%	44.6±0.2	44.6±0.1	44.4±0.6	44.1±0.4	47.1±0.1	47.1±0.1	47.1±0.1	**49.9±0.6**	
Reddit	0.05%	46.1±4.4	40.9±0.5	53.1±2.5	46.6±2.3	88.0±1.8	89.7±0.2	91.8±1.9	**92.9±0.3**	93.9±0.0
	0.10%	58.0±2.2	42.8±0.8	62.7±1.0	53.0±3.3	89.6±0.7	90.0±0.3	91.0±1.6	**93.1±0.2**	
	0.50%	66.3±1.9	47.4±0.9	71.0±1.6	58.5±2.1	90.1±0.5	89.9±0.4	91.6±1.8	**93.2±0.4**	
Squirrel	0.60%	22.4±1.6	20.9±1.1	21.3±1.1	21.8±0.3	27.0±1.3	24.0±0.4	24.1±2.3	**28.4±2.0**	33.0±0.4
	1.20%	25.0±0.2	21.1±0.4	21.4±2.1	22.8±0.9	25.7±2.3	26.9±2.5	24.7±2.5	**28.2±2.4**	
	2.50%	26.9±1.4	21.5±0.3	22.4±1.6	22.9±1.7	25.3±0.8	26.1±0.8	25.8±1.8	**27.8±1.6**	
Gamers	0.05%	56.6±1.8	56.1±0.1	56.7±1.7	52.5±4.2	58.5±1.5	58.2±1.1	57.5±1.8	**59.3±1.9**	62.6±0.0
	0.25%	60.5±1.0	56.9±3.0	57.5±2.0	57.2±2.3	58.9±1.8	58.8±0.5	57.7±1.0	**60.8±0.4**	
	0.50%	60.0±0.5	57.1±0.4	58.6±1.3	57.8±1.7	58.5±1.9	59.9±0.3	58.4±1.7	**61.2±0.3**	

6.2. Node Classification

The node classification performance is reported in Table 2, in which we have the following observations:

First, the GD methods consistently outperform the traditional methods, including coreset and coarsening. The reasons are two-fold: On the one hand, GD methods can leverage the powerful representation learning ability of GNNs to synthesize the graph data. On the other hand, the distillation process involves the downstream task information. In contrast, the traditional methods can only leverage the structural information.

Second, GDEM achieves state-of-the-art performance in 6 out of 7 graph datasets, demonstrating its effectiveness in preserving the distribution of real graphs. Existing GD methods heavily rely on the information of GNNs to distill synthetic graphs. However, the results of GDEM reveal that matching eigenbasis can also yield good synthetic graphs. Furthermore, some results of GDEM are better than those on the entire dataset, which may be due to the use of high-frequency information.

Third, GDEM performs slightly worse on Ogbn-arxiv but achieves promising results on other large-scale graphs. We conjecture this is because, under the compression ratios of 0.05% - 0.50%, there are only hundreds of eigenvectors for eigenbasis matching, which is not enough to cover all the useful subspaces in Ogbn-arxiv. See Appendix A.9 for further experimental verification.

6.3. Cross-architecture Generalization

We evaluate the generalization ability of the synthetic graphs distilled by four different GD methods, including GCOND, SFGC, SGDD, and GDEM. In particular, each synthetic graph is evaluated by six GNNs, and the average accuracy and variance of the evaluation results are shown in Table 3.

First, GDEM stands out by exhibiting the highest average accuracy across datasets except for Ogbn-arxiv, indicating that the synthetic graphs distilled by GDEM can consistently benefit a variety of GNNs. Moreover, GDEM significantly reduces the performance gap between different GNNs. For example, the variance of GCOND is 2-6 times higher than that of GDEM. On the other hand, SGDD broadcasts the structural information to synthetic graphs and exhibits better generalization ability than GCOND, implying that preserving graph structures can improve the generalization of synthetic graphs. SFGC proposes structure-free distillation. However, this strategy may lead to restricted application scenarios due to the lack of explicit graph structures.

Table 3. Generalization of different distillation methods across GNNs. ↑ means higher the better and ↓ means lower the better. Avg., Std., and Impro. indicate average accuracy, standard deviation, and absolute performance improvement.

Dataset (Ratio)	**Methods**	**Spatial GNNs**			**Spectral GNNs**			**Avg.** (↑)	**Std.** (↓)	**Impro.** (↑)
		GCN	SGC	PPNP	ChebyNet	BernNet	GPR-GNN			
Citeseer ($r = 1.80\%$)	GCOND	70.5	70.3	69.6	68.3	63.1	67.2	68.17	2.54	(+) 4.21
	SFGC	71.6	71.8	70.5	71.8	71.1	71.7	71.42	**0.47**	(+) 0.96
	SGDD	70.2	71.3	69.2	70.5	64.7	69.7	69.27	2.14	(+) 3.11
	GDEM	72.6	72.1	72.6	71.4	72.6	73.0	**72.38**	0.51	-
Pubmed ($r = 0.15\%$)	GCOND	77.7	77.6	77.3	76.0	74.4	76.5	76.58	1.15	(+) 1.34
	SFGC	77.5	77.4	77.6	77.3	76.4	78.6	77.47	**0.64**	(+) 0.45
	SGDD	78.0	76.6	78.7	76.9	75.5	77.0	77.12	1.02	(+) 0.80
	GDEM	78.4	76.1	78.1	78.1	78.2	78.6	**77.92**	0.83	-
Ogbn-arxiv ($r = 0.25\%$)	GCOND	63.2	63.7	63.4	54.9	55.0	60.5	60.12	3.80	(+) 2.90
	SFGC	65.1	64.8	63.9	60.7	63.8	64.9	**63.87**	1.50	(-) 0.85
	SGDD	65.8	64.0	63.6	56.4	62.0	64.0	62.63	3.00	(+) 0.39
	GDEM	63.8	62.9	63.5	62.4	61.9	63.6	63.02	**0.69**	-
Flickr ($r = 0.50\%$)	GCOND	47.1	46.1	45.9	42.8	44.3	46.4	45.43	1.45	(+) 3.90
	SFGC	47.1	42.5	40.7	45.4	45.7	46.4	44.63	2.27	(+) 4.70
	SGDD	47.1	46.5	44.3	45.3	46.0	46.8	46.00	0.96	(+) 3.33
	GDEM	49.4	50.3	49.4	48.3	49.6	49.0	**49.33**	**0.60**	-
Reddit ($r = 0.10\%$)	GCOND	89.4	89.6	87.8	75.5	67.1	78.8	81.37	8.35	(+) 10.10
	SFGC	89.7	89.5	88.3	82.8	87.8	85.4	87.25	2.44	(+) 4.22
	SGDD	91.0	89.4	89.2	78.4	72.4	81.4	83.63	6.80	(+) 7.84
	GDEM	93.1	90.0	92.6	90.0	92.7	90.4	**91.47**	**1.35**	-
Squirrel ($r = 1.20\%$)	GCOND	25.7	27.2	23.2	23.3	26.0	26.6	25.33	1.55	(+) 1.89
	SFGC	26.9	24.2	27.2	25.3	25.5	26.6	25.95	**1.04**	(+) 1.27
	SGDD	24.7	27.2	22.4	24.5	24.7	27.3	25.13	1.69	(+) 2.09
	GDEM	28.2	28.0	25.4	26.1	28.2	27.4	**27.22**	1.09	-
Gamers ($r = 0.25\%$)	GCOND	58.9	54.2	60.1	60.3	59.1	59.3	58.65	2.05	(+) 1.57
	SFGC	58.8	55.0	56.3	57.2	57.5	59.8	57.43	1.57	(+) 2.79
	SGDD	57.7	54.6	56.0	57.3	58.8	58.6	57.17	1.47	(+) 3.05
	GDEM	60.8	59.5	61.0	59.9	59.8	60.3	**60.22**	**0.54**	-

Table 4. Optimal performance of different methods.

Evaluation	GCN	SGC	PPNP	Cheb.	Bern.	GPR.
GCOND	77.7	**77.6**	77.9	77.3	**78.2**	78.3
SGDD	78.0	76.6	**78.7**	77.5	78.0	78.3
GDEM	**78.4**	76.1	78.1	**78.1**	**78.2**	**78.6**

Table 5. Time overhead (s) of different methods.

Distillation	GCN	SGC	PPNP	Cheb.	Bern.	GPR.	Overall
GCOND	1.99	1.36	1.52	3.89	56.94	3.05	68.75
SGDD	2.95	2.18	2.33	4.95	58.07	4.28	74.76
GDEM	-	-	-	-	-	-	**1.79**

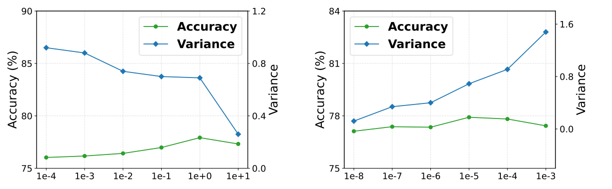
6.4. Optimal Performance and Time Overhead

We compare the optimal performance and time overhead of different GD methods by traversing various GNN architectures in the Pubmed dataset. Since GDEM does not use GNNs during distillation, we remove the inner- and outer-loop of GCOND and SGDD when calculating the time overhead for a fair comparison. Therefore, the running time is faster than the results in Yang et al. (2023).

In Table 4, we can find that both GCOND and SGDD improve their performance by traversing different GNNs compared to the results in Table 3. However, this strategy also introduces additional computation costs in the distillation stage. As shown in Table 5, the complexity of GCOND and SGDD is related to the complexity of distillation GNNs. Notably, when choosing GNNs with high complexity, *e.g.*, BernNet, their time overhead will increase significantly. On the other hand, GDEM still exhibits remarkable performance compared to the traversal results of GCOND and SGDD. More importantly, the complexity of GDEM will not be affected by GNNs, which eliminates the traversal requirement of previous methods. As a result, the overall time overhead of GDEM is significantly smaller than GCOND and SGDD, which validates the efficiency of GDEM. See Appendix A.2 for more generalization results of GCOND and SGDD.

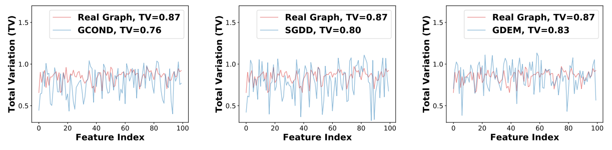
Table 6. Ablation studies on Pubmed / Gamers.

Pubmed	GCN (↑)	GPR. (↑)	Avg. (↑)	Var. (↓)
GDEM	78.4 / 60.8	78.6 / 60.3	77.92 / 60.22	0.69 / 0.29
w/o $\mathcal{L}_e$	76.1 / 56.5	76.9 / 59.8	76.13 / 58.93	1.18 / 2.39
w/o $\mathcal{L}_o$	77.9 / 59.0	76.4 / 58.9	77.07 / 58.85	2.15 / 2.34
w/o $\mathcal{L}_d$	76.7 / 59.9	77.2 / 60.3	76.77 / 59.78	0.21 / 0.13



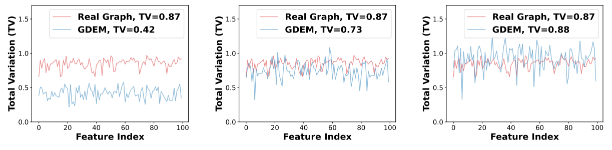
(a) Analysis of α with β=10^{-5} (b) Analysis of β with α=1.0

Figure 3. Influence of $\mathcal{L}_e$ and $\mathcal{L}_d$ in GDEM.



(a) GCOND (b) SGDD (c) GDEM

Figure 4. TVs of synthetic graphs distilled by different methods.



(a) Epoch: 50 (b) Epoch: 100 (c) Epoch: 200

Figure 5. TVs of synthetic graphs at different epochs (GDEM).

6.5. Ablation Study

We perform ablation studies in the Pubmed and Gamers datasets to verify the effectiveness of different regularization terms, *i.e.*, $\mathcal{L}_e$, $\mathcal{L}_o$, and $\mathcal{L}_d$.

Model Analysis. Table 6 shows the roles of different regularization terms. First, all of them contribute to both the effectiveness and generalization of GDEM. Specifically, $\mathcal{L}_e$ and $\mathcal{L}_o$ primarily govern the generalization ability of GDEM, as the variance of GNNs increases significantly when removing either of them. Second, we observe that $\mathcal{L}_d$ hurts the generalization of GDEM. The reason is that the discrimination constraint uses the information of the graph spectrum and introduces the low-frequency preference. But it also improves the performance of GDEM. Therefore, GDEM needs to carefully balance these two loss functions.

Parameters Analysis. We conduct an additional parameter analysis to further demonstrate the influence of $\mathcal{L}_e$ and $\mathcal{L}_d$, as illustrated in Figure 3. Specifically, we observe that with the increase in α, the variance of GDEM gradually decreases. However, a higher value of α also leads to performance degeneration. On the other hand, increasing the value of β will continue to increase the variance of GDEM but the accuracy decreases when β surpasses a specific threshold.

6.6. Visualization

We visualize the data distribution of synthetic graphs for a better understanding of our model. Specifically, Figure 4 illustrates the synthetic graphs distilled by GCOND, SGDD, and GDEM, from which we can observe that the value of TV in GDEM is the closest to the real graph. SGDD is closer to the distribution of the real graph than GCOND, implying that SGDD can better preserve the structural information. However, the performance is still not as good as GDEM, which validates the effectiveness of eigenbasis matching.

Besides, we also visualize the synthetic graphs distilled by GDEM at different epochs in Figure 5. We can find that with the optimization of GDEM, the value of TV in the synthetic graphs is approaching the real graph (0.42 $\rightarrow$ 0.73 $\rightarrow$ 0.88), which validates Proposition 5.3 that GDEM can preserve the spectral similarity of the real graph.

7. Related Work

Graph Neural Networks aim to design effective convolution operators to exploit the node features and topology structure information adequately. GNNs have achieved great success in graph learning and play a vital role in diverse real-world applications (Quan et al., 2023; Yang et al., 2017). Existing methods are roughly divided into spatial and spectral approaches. Spatial GNNs focus on neighbor aggregation strategies in the vertical domain (Kipf & Welling, 2017; Velickovic et al., 2018; Hamilton et al., 2017). Spectral GNNs aim to design filters in the spectral domain to extract certain frequencies for the downstream tasks (Chien et al., 2021; Defferrard et al., 2016; Bo et al., 2023; He et al., 2022; 2021).

Dataset Distillation (DD) has shown great potential in reducing data redundancy and accelerating model training (Sachdeva & McAuley, 2023; Lei & Tao, 2023; Geng et al., 2023; Yu et al., 2023). DD aims to generate small yet informative synthetic training data by matching the model gradient (Zhao et al., 2021; Liu et al., 2022b), data distribution (Zhao & Bilen, 2023; Wang et al., 2022), and training trajectory (Cazenavette et al., 2022; Guo et al., 2023) between the real and synthetic data. As a result, models trained on the real and synthetic data will have comparable performance.

DD has been widely used for graph data, including node-level tasks, *e.g.*, GCond (Jin et al., 2022b), SFGC (Zheng et al., 2023), GCDM (Liu et al., 2022a) and MCond (Gao et al., 2023), and graph-level tasks, *e.g.*, DosCond (Jin et al., 2022a) and KIDD (Xu et al., 2023). GCond is the first GD method based on gradient matching, which needs to optimize GNNs during the distillation procedure, resulting in inefficient computation. DosCond further provides one-step gradient matching to approximate gradient matching, thereby avoiding the bi-level optimization. GCDM proposes distribution matching for GD, which views the receptive fields of a graph as its distribution. Additionally, SFGC proposes structure-free GD to compress the structural information into the node features. KIDD utilizes the kernel ridge regression to further reduce the computational cost. However, all these methods do not consider the influence of GNNs, resulting in spectrum bias and traversal requirement.

8. Conclusion

In this paper, we propose eigenbasis matching for graph distillation, which only aligns the eigenbasis and node features of the real and synthetic graphs, thereby alleviating the spectrum bias and traversal requirement of the previous methods. Theoretically, GDEM preserves the restricted spectral similarity of the real graphs. Extensive experiments on both homophilic and heterophilic graphs validate the effectiveness, generalization, and efficiency of the proposed method. A promising future work is to explore eigenbasis matching without the need for explicit eigenvalue decomposition.

Acknowledgements

This work is supported in part by the National Natural Science Foundation of China (No. U20B2045, 62192784, U22B2038, 62002029, 62172052).

Impact Statement

This paper presents work aiming to advance the field of efficient graph learning and will save social resources by diminishing computation and storage energy consumption. There are many potential societal consequences of our work, none of which we feel must be specifically highlighted here.

References

Ahmed, A. R., Bodwin, G., Sahneh, F. D., Hamm, K., Jebelli, M. J. L., Kobourov, S. G., and Spence, R. Graph spanners: A tutorial review. *Comput. Sci. Rev.*, 37:100253, 2020.

Bo, D., Wang, X., Shi, C., and Shen, H. Beyond low-frequency information in graph convolutional networks. In *AAAI*, pp. 3950–3957. AAAI Press, 2021.

Bo, D., Shi, C., Wang, L., and Liao, R. Specformer: Spectral graph neural networks meet transformers. In *ICLR*, 2023.

Cazenavette, G., Wang, T., Torralba, A., Efros, A. A., and Zhu, J. Dataset distillation by matching training trajectories. In *CVPR*, pp. 10708–10717. IEEE, 2022.

Chien, E., Peng, J., Li, P., and Milenkovic, O. Adaptive universal generalized pagerank graph neural network. In *ICLR*. OpenReview.net, 2021.

Defferrard, M., Bresson, X., and Vandergheynst, P. Convolutional neural networks on graphs with fast localized spectral filtering. In *NIPS*, pp. 3837–3845, 2016.

Gao, X., Chen, T., Zang, Y., Zhang, W., Nguyen, Q. V. H., Zheng, K., and Yin, H. Graph condensation for inductive node representation learning. *ArXiv*, abs/2307.15967, 2023.

Gao, X., Yu, J., Jiang, W., Chen, T., Zhang, W., and Yin, H. Graph condensation: A survey. *ArXiv*, abs/2401.11720, 2024.

Geng, J., Chen, Z., Wang, Y., Woisetschlaeger, H., Schimmler, S., Mayer, R., Zhao, Z., and Rong, C. A survey on dataset distillation: Approaches, applications and future directions. In *IJCAI*, pp. 6610–6618. ijcai.org, 2023.

Guo, Z., Wang, K., Cazenavette, G., Li, H., Zhang, K., and You, Y. Towards lossless dataset distillation via difficulty-aligned trajectory matching. *arXiv preprint arXiv:2310.05773*, 2023.

Gutman, I. and Zhou, B. Laplacian energy of a graph. *Linear Algebra and its applications*, 414(1):29–37, 2006.

Hamilton, W. L., Ying, Z., and Leskovec, J. Inductive representation learning on large graphs. In *NIPS*, pp. 1024–1034, 2017.

He, M., Wei, Z., Huang, Z., and Xu, H. Bernnet: Learning arbitrary graph spectral filters via bernstein approximation. In *NeurIPS*, pp. 14239–14251, 2021.

He, M., Wei, Z., and Wen, J. Convolutional neural networks on graphs with chebyshev approximation, revisited. In *NeurIPS*, 2022.

Hu, W., Fey, M., Zitnik, M., Dong, Y., Ren, H., Liu, B., Catasta, M., and Leskovec, J. Open graph benchmark: Datasets for machine learning on graphs. In *NeurIPS*, 2020.

Jin, W., Tang, X., Jiang, H., Li, Z., Zhang, D., Tang, J., and Yin, B. Condensing graphs via one-step gradient matching. In *KDD*, pp. 720–730, 2022a.

Jin, W., Zhao, L., Zhang, S., Liu, Y., Tang, J., and Shah, N. Graph condensation for graph neural networks. In *ICLR*, 2022b.

Jin, Y., Loukas, A., and JáJá, J. F. Graph coarsening with preserved spectral properties. In *AISTATS*, volume 108, pp. 4452–4462. PMLR, 2020.

Kipf, T. N. and Welling, M. Semi-supervised classification with graph convolutional networks. In *ICLR*. OpenReview.net, 2017.

Klicpera, J., Bojchevski, A., and Günnemann, S. Predict then propagate: Graph neural networks meet personalized pagerank. In *ICLR*. OpenReview.net, 2019.

Kumar, M., Sharma, A., Saxena, S., and Kumar, S. Featured graph coarsening with similarity guarantees. In *ICML*, volume 202 of *Proceedings of Machine Learning Research*, pp. 17953–17975. PMLR, 2023.

Lei, S. and Tao, D. A comprehensive survey of dataset distillation. *ArXiv*, abs/2301.05603, 2023.

Lim, D., Hohne, F., Li, X., Huang, S. L., Gupta, V., Bhalerao, O., and Lim, S. Large scale learning on non-homophilous graphs: New benchmarks and strong simple methods. In *NeurIPS*, pp. 20887–20902, 2021.

Liu, M., Li, S., Chen, X., and Song, L. Graph condensation via receptive field distribution matching. *ArXiv*, abs/2206.13697, 2022a.

Liu, S., Wang, K., Yang, X., Ye, J., and Wang, X. Dataset distillation via factorization. In *NeurIPS*, 2022b.

Loukas, A. Graph reduction with spectral and cut guarantees. *J. Mach. Learn. Res.*, 20:116:1–116:42, 2019.

Martinkus, K., Loukas, A., Perraudin, N., and Wattenhofer, R. SPECTRE: spectral conditioning helps to overcome the expressivity limits of one-shot graph generators. In *ICML*, volume 162, pp. 15159–15179. PMLR, 2022.

Quan, Y., Ding, J., Gao, C., Yi, L., Jin, D., and Li, Y. Robust preference-guided denoising for graph based social recommendation. In *WWW*, pp. 1097–1108. ACM, 2023.

Rozemberczki, B., Allen, C., and Sarkar, R. Multi-scale attributed node embedding. *J. Complex Networks*, 9(2), 2021.

Sachdeva, N. and McAuley, J. Data distillation: A survey. *ArXiv*, abs/2301.04272, 2023.

Sener, O. and Savarese, S. Active learning for convolutional neural networks: A core-set approach. In *ICLR*. OpenReview.net, 2018.

Spielman, D. A. and Srivastava, N. Graph sparsification by effective resistances. *SIAM J. Comput.*, 40(6):1913–1926, 2011.

Velickovic, P., Cucurull, G., Casanova, A., Romero, A., Liò, P., and Bengio, Y. Graph attention networks. In *ICLR*, 2018.

Wang, K., Zhao, B., Peng, X., Zhu, Z., Yang, S., Wang, S., Huang, G., Bilen, H., Wang, X., and You, Y. CAFE: learning to condense dataset by aligning features. In *CVPR*, pp. 12186–12195. IEEE, 2022.

Welling, M. Herding dynamical weights to learn. In *ICML*, volume 382, pp. 1121–1128. ACM, 2009.

Wu, F., Jr., A. H. S., Zhang, T., Fifty, C., Yu, T., and Weinberger, K. Q. Simplifying graph convolutional networks. In *ICML*, volume 97, pp. 6861–6871. PMLR, 2019.

Xu, Z., Chen, Y., Pan, M., Chen, H., Das, M., Yang, H., and Tong, H. Kernel ridge regression-based graph dataset distillation. In *KDD*, pp. 2850–2861, 2023.

Yang, B., Wang, K., Sun, Q., Ji, C., Fu, X., Tang, H., You, Y., and Li, J. Does graph distillation see like vision dataset counterpart? In *NeurIPS*, 2023.

Yang, C., Sun, M., Zhao, W. X., Liu, Z., and Chang, E. Y. A neural network approach to jointly modeling social networks and mobile trajectories. *ACM Transactions on Information Systems (TOIS)*, 35(4):1–28, 2017.

Yu, R., Liu, S., and Wang, X. Dataset distillation: A comprehensive review. *ArXiv*, abs/2301.07014, 2023.

Yu, S., Alesiani, F., Yin, W., Jenssen, R., and Príncipe, J. C. Principle of relevant information for graph sparsification. In *UAI*, volume 180 of *Proceedings of Machine Learning Research*, pp. 2331–2341. PMLR, 2022.

Zeng, H., Zhou, H., Srivastava, A., Kannan, R., and Prasanna, V. K. Graphsaint: Graph sampling based inductive learning method. In *ICLR*. OpenReview.net, 2020.

Zhao, B. and Bilen, H. Dataset condensation with distribution matching. In *WACV*, pp. 6503–6512. IEEE, 2023.

Zhao, B., Mopuri, K. R., and Bilen, H. Dataset condensation with gradient matching. In *ICLR*. OpenReview.net, 2021.

Zhao, G., Li, G., Qin, Y., and Yu, Y. Improved distribution matching for dataset condensation. In *CVPR*, pp. 7856–7865. IEEE, 2023.

Zheng, X., Zhang, M., Chen, C., Nguyen, Q. V. H., Zhu, X., and Pan, S. Structure-free graph condensation: From large-scale graphs to condensed graph-free data. *ArXiv*, abs/2306.02664, 2023.

A. Experimental Details

A.1. Visualization of Synthetic Graphs

Distillation Details with Low-pass and High-pass Filters. We use GCOND to distill two synthetic graphs on Pubmed by replacing SGC with a low-pass filter $\mathcal{F}_L = \mathbf{AXW}$ and a high-pass filter $\mathcal{F}_H = \mathbf{LXW}$, respectively.

Visualization Details Once we generate the synthetic graphs, we calculate the value of total variation (TV) for each dimension. TV is a widely used metric to represent the distribution, i.e., smoothness, of a signal on the graph:

$$\mathbf{x}^\top \mathbf{L}\mathbf{x} = \sum\nolimits_{(i,j)\in\mathcal{E}} (x_i - x_j)^2 = \sum_{i=1}^{n} \lambda_i \mathbf{x}^\top \mathbf{u}_i \mathbf{u}_i^\top \mathbf{x}. \tag{14}$$

Note that the edge number of synthetic graphs and the original graph is different, so we normalize node features and laplacian matrix first:

$$\begin{aligned} \hat{\mathbf{x}}_i &= \frac{\mathbf{x}_i}{\|\mathbf{x}_i\|}, \\ \hat{\mathbf{L}} &= \mathbf{I}_N - \mathbf{D}^{-\frac{1}{2}}\mathbf{A}\mathbf{D}^{-\frac{1}{2}}, \end{aligned} \tag{15}$$

where $\mathbf{x}_i$ is the i-th dimension node feature. Then we substitute $\hat{\mathbf{x}}_i$ and $\hat{\mathbf{L}}$ into Equation 14 calculating the TV of the graph. Additionally, we report the average TV of all dimensions as reported in the legend of the visualization figures.

A.2. Cross-architecture Performance of GCOND and SGDD

To verify the cross-architecture performance of the GCOND and SGDD, we generate six synthetic graphs on Pubmed under a 0.15% compression ratio, using six GNNs for the distillation procedure. Then we train these GNNs on the six synthetic graphs and evaluate their performance. Experimental settings are as follows.

Distillation Step. For spatial GNNs, *i.e.*, GCN, SGC, and PPNP, we set the aggregation layers to 2. For GCN, we use 256 hidden units for each convolutional layer. For spectral GNNs, *i.e.*, ChebyNet, BernNet, and GPR-GNN, we set the polynomial order to 10. The linear feature transformation layers of all GNNs are set to 1. For hyper-parameters tuning, we select training epochs from {400, 500, 600}, learning rates of node feature and topology structure from {0.0001, 0.0005, 0.001, 0.005, 0.05}, outer loop from {25, 20, 15, 10}, and inner loop from {15, 10, 5, 1}.

Evaluation Step. For spatial GNNs, we use two aggregation layers. For spectral GNNs, we set the polynomial order to 10. The hidden units of convolutional layers and linear feature transformation layers are both set to 256. We train each GNN for 2000 epochs and select the model parameters with the best performance on validation sets for evaluation.

Table 7. GCOND with various distillation (D) and evaluation (E) GNNs in Pubmed dataset.

D \ E	GCN	SGC	PPNP	Cheb.	Bern.	GPR.
GCN	74.57	71.70	75.53	70.13	68.40	71.73
SGC	**77.72**	**77.60**	77.34	76.03	74.42	76.52
PPNP	72.70	70.40	77.46	73.38	70.56	74.02
Cheb.	73.60	70.62	75.10	**77.30**	77.62	78.10
Bern.	67.68	73.76	74.30	77.20	**78.12**	**78.28**
GPR.	76.04	72.20	**77.94**	75.92	77.12	77.96
Optimal	77.72	77.60	77.94	77.30	78.12	78.28

Table 8. SGDD with various distillation (D) and evaluation (E) GNNs in Pubmed dataset.

D \ E	GCN	SGC	PPNP	Cheb.	Bern.	GPR.
GCN	76.92	70.10	74.64	74.98	76.66	75.18
SGC	**78.04**	**76.60**	**78.72**	76.90	75.45	77.02
PPNP	76.44	74.34	76.28	73.70	74.94	75.98
Cheb.	77.42	73.66	75.40	**77.50**	**77.96**	77.12
Bern.	70.64	71.22	74.88	76.38	76.16	77.84
GPR.	63.76	61.24	76.32	71.40	71.70	**78.30**
Optimal	78.04	76.60	78.72	77.50	77.96	78.30

A.3. Implementation Details of GDEM

Predefined Labels $\mathbf{Y}'$ of Synthetic Graphs. The labels $\mathbf{Y}'$ are predefined one-hot vectors, indicating the category to which the nodes belong. Specifically, given N_l labeled nodes in the real graph, we set the number of nodes of category c in the synthetic graph as $N_c' = N_c \times \frac{N'}{N_l}$, where N_c is the number of nodes with label c. The setting will make the label distribution of the synthetic graph consistent with the real graph.

Initialization of Synthetic Graphs. Different from previous GD methods that directly learn the adjacency matrix of the synthetic graph, GDEM aims to generate its eigenbasis. To ensure that the initialized eigenbasis is valid, we first use the stochastic block model (SBM) to randomly generate the adjacency matrix of the synthetic graph $\mathbf{A}' \in \{0,1\}^{N' \times N'}$, and then decompose it to produce the top-K eigenvectors as the initialized eigenbasis $\mathbf{U}'_K \in \mathbb{R}^{N' \times K}$. Moreover, to initialize the synthetic node features $\mathbf{X}' \in \mathbb{R}^{N' \times d}$, we first train an MLP $\rho(\cdot)$ in the real node features. Then we freeze the well-trained MLP and feed the synthetic node features into it to minimize the classification objective. This process can be formulated as:

$$\min_{\mathbf{X}'} \sum_{i=1}^{n'} -y'_i \log \rho(\mathbf{x}'_i, \theta^*), \text{ s.t. } \theta^* = \arg\min_{\theta} \sum_{i=1}^{n} -y_i \log \rho(\mathbf{x}_i, \theta) \tag{16}$$

where θ indicates the parameters of MLP.

A.4. Complexity of Different Methods

We analyze the complexity of different methods and give the final complexity in Table 9. We use E to present the number of edges. For simplicity, we use d to denote both feature dimension and hidden units of GNNs. t is the number of GNN layers and r is the number of sampled neighbors per node. θ_t denotes the model parameters of the GNNs. For SFGC, M is the number of training trajectories and S is the length of each trajectory.

Complexity of GDEM.

(1) Pre-processing: The complexity of decomposition is $\mathcal{O}(KN^2)$. It's noteworthy that the decomposition is performed once per graph and can be repeatedly used for subsequent training, inference, and hyperparameter tuning. Therefore, the time overhead of decomposition should be amortized by the entire experiment rather than simply summarized them. Additionally, we pre-process $\mathbf{u}_k^\top \mathbf{X}$ in Equation 5 and H in Equation 8, which cost $\mathcal{O}(KNd)$ and $\mathcal{O}(Ed)$.
(1) Complexity of $\mathcal{L}_e$: $\mathcal{O}(KN'd + Kd^2)$.
(2) Complexity of $\mathcal{L}_d$: The complexity of calculating H' is $\mathcal{O}(KN'd')$. The calculation of cosine similarity costs $\mathcal{O}(Cd^2)$.
(3) Complexity of $\mathcal{L}_o$: $\mathcal{O}(KN'^2)$.
The final complexity can be simplified as $\mathcal{O}(KN^2 + KNd + Ed) + \mathcal{O}(KN'^2 + KN'd + (K+C)d^2)$.

Complexity of GCOND.

(1) Pre-processing: GCOND doesn't need special pre-processing.
(2) Inference for A': $\mathcal{O}(N'^2 d^2)$.
(3) Forward process of SGC on the original graph: $\mathcal{O}(r^t N d^2)$. That on the synthetic graph: $\mathcal{O}(tN'^2 d + tN'd)$.
(4) Calculation of second-order derivatives in backward propagation: $\mathcal{O}(|\theta_t| + |A'| + |X'|)$.
The final complexity can be simplified as $\mathcal{O}(r^t N d^2) + \mathcal{O}(N'^2 d^2)$.

Complexity of SGDD.

(1) Pre-processing: SGDD doesn't need special pre-processing.
(2) Inference for A': $\mathcal{O}(N'^2 d^2)$.
(3) Forward process of SGC on the original graph: $\mathcal{O}(r^t N d^2)$. That on the synthetic graph: $\mathcal{O}(tN'^2 d + tN'd)$.
(4) Calculation of second-order derivatives in backward propagation: $\mathcal{O}(|\theta_t| + |A'| + |X'|)$.
(5) Structure optimization term: $\mathcal{O}(N'^2 k + NN'^2)$.
The final complexity can be simplified as $\mathcal{O}(r^t N d^2) + \mathcal{O}(N'^2 N)$.

Complexity of SFGC.

(1) Pre-processing: $\mathcal{O}(MS(tEd + tNd^2))$. Note that MS is usually very large, so it cannot be omitted.
(2) Forward process of GCN on the synthetic graph: $\mathcal{O}(tN'd^2 + tN'd)$. Note that SFGC pre-trains the trajectories on GCN, so there is no need to calculate the forward process on the original graph.
(3) Backward propagation: SFGC uses a MTT(Cazenavette et al., 2022) method, which results in bi-level optimization(Yu et al., 2023) for the backward.
The final complexity can be simplified as $\mathcal{O}(MS(tEd + tNd^2)) + \mathcal{O}(tN'd^2)$.

Table 9. Complexity of different distillation methods.

Method	Pre-processing	Training
GCOND	-	$\mathcal{O}(r^L N d^2) + \mathcal{O}(N'^2 d^2)$
SGDD	-	$\mathcal{O}(r^L N d^2) + \mathcal{O}(N'^2 N)$
SFGC	$\mathcal{O}(MS(LEd + LNd^2))$	$\mathcal{O}(LN'd^2)$
GDEM	$\mathcal{O}(KN^2 + KNd + Ed)$	$\mathcal{O}(KN'^2 + KN'd + (K+C)d^2)$

A.5. Statistics of Datasets

In the experiments, we use seven graph datasets to validate the effectiveness of GDEM. For homophilic graphs, we use the public data splits. For heterophilic graphs, we use the splitting with training/validation/test sets accounting for 2.5%/2.5/%95% on Squirrel, and 50%/25%25% on Gamers. The detailed statistical information of each dataset is shown in Table 10.

Table 10. Statistics of datasets.

Dataset	**Nodes**	**Edges**	**Classes**	**Features**	**Training/Validation/Test**	**Edge hom.**	**LCC**
Citeseer	3,327	4,732	6	3,703	120/500/1000	0.74	2,120
Pubmed	19,717	44,338	3	500	60/500/1,000	0.80	19,717
Ogbn-arxiv	169,343	1,166,243	40	128	90,941/29,799/48,603	0.66	169,343
Flickr	89,250	899,756	7	500	44,625/22,312/22,313	0.33	89,250
Reddit	232,965	57,307,946	41	602	153,932/23,699/55,334	0.78	231,371
Squirrel	5,201	396,846	5	2,089	130/130/4,941	0.22	5,201
Gamers	168,114	13,595,114	2	7	84,056/42,028/42,030	0.55	168,114

A.6. Baselines

For a fair comparison of performance, we adopt the results of baselines reported in their papers, which are evaluated through meticulous experimental design and careful hyperparameter tuning. The experimental details are as follows: (1) GCOND employs a 2-layer SGC for distillation and a 2-layer GCN with 256 hidden units for evaluation.
(2) SGDD employs a 2-layer SGC for distillation and a 2-layer GCN with 256 hidden units for evaluation.
(3) SFGC employs 2-layer GCNs with 256 hidden units both for distillation and evaluation.

A.7. Evaluation Details

Performance Evaluation. For comparison with baselines, we report the performance of GDEM evaluated with a 2-layer GCN with 256 hidden units. Specifically, we generate 10 synthetic graphs with different seeds on the original graph. Then we train the GCN using these 10 synthetic graphs and report the average results of the best performance evaluated on test sets of the original graph.

Generalization Evaluation. For generalization evaluation, we train 6 GNNs using the synthetic graphs generated by different distillation methods. For SGC, GCN, and APPNP, we use 2-layer aggregations. For ChebyNet, we set the convolution layers to 2 with propagation steps from $\{2, 3, 5\}$. For BernNet and GPRGNN, we set the polynomial order to 10. The hidden units of both convolution layers and linear feature transformation are 256.

A.8. Hyperparamters

Hyperparameter details are listed in Table 11. τ_1 and τ_2 are steps for alternating updates of node features and eigenvectors. α, β, and γ denote the weights in Equation 9. lr_feat and lr_eigenvecs are the learning rates of node features and eigenvectors, respectively.

Table 11. Hyper-parameters of GDEM.

Dataset	Ratio	epochs	K_1	K_2	τ_1	τ_2	α	β	γ	lr_feat	lr_eigenvecs
Citeseer	0.90%	500	30	0	5	1	1.0	1e-05	1.0	0.0001	0.01
	1.80%	1500	48	12	10	15	0.05	1e-05	0.5	0.0005	0.0005
	3.60%	500	114	6	1	10	0.01	1e-06	0.1	0.001	0.0001
Pubmed	0.08%	1000	15	0	15	5	0.0001	1e-07	0.01	0.0001	0.0005
	0.15%	1500	30	0	5	5	1.0	1e-05	0.01	0.0005	0.01
	0.30%	1500	57	3	20	1	0.01	1e-07	0.5	0.001	0.0001
Ogbn-arxiv	0.05%	500	86	4	1	5	0.0001	1e-02	0.01	0.0005	0.0005
	0.25%	2000	409	45	10	5	0.01	1e-04	0.01	0.0001	0.0001
	0.50%	1000	773	136	1	5	0.001	1e-04	1.0	0.0001	0.005
Flickr	0.10%	2000	44	0	5	10	0.01	1e-07	0.05	0.0001	0.05
	0.50%	2000	223	0	5	10	0.01	1e-07	0.05	0.0001	0.05
	1.00%	2000	446	0	5	10	0.01	1e-07	0.05	0.0001	0.05
Reddit	0.05%	1000	76	0	20	5	1.0	1e-06	0.01	0.0001	0.0001
	0.10%	500	153	0	15	10	0.5	1e-06	05	0.0005	0.005
	0.50%	1000	693	76	5	5	1.0	1e-06	0.5	0.0005	0.0001
Squirrel	0.60%	1000	31	1	5	1	1.0	1e-07	0.01	0.0001	0.005
	1.20%	500	62	3	10	5	1.0	1e-07	0.01	0.0001	0.0001
	2.05%	2000	104	26	5	1	0.0001	1e-05	0.05	0.0001	0.01
Gamers	0.05%	2000	80	4	15	1	0.0001	1e-07	0.05	0.0001	0.01
	0.25%	2000	420	0	20	20	0.0001	1e-07	0.05	0.0001	0.005
	0.50%	500	756	84	15	1	0.0001	1e-07	0.05	0.0001	0.0001

Table 12. The node classification performance of Ogbn-arxiv and Reddit on various truncated graph structures.

Dataset	$K = 500$	$K = 1000$	$K = 3000$	$K = 5000$	Full Graph
Reddit	92.41±0.49	93.45±0.48	93.94±0.41	94.07±0.37	94.51±0.24
Ogbn-arxiv	61.87±0.89	64.65±1.20	67.32±1.11	69.22±0.93	70.02±1.19

A.9. Analysis of the Worse Performance on Obgn-arxiv

To investigate the reason why GDEM performs slightly worse on Obgn-arxiv but achieves promising results on other large-scale graphs, we evaluate the number of useful eigenbasis in both Ogbn-arxiv and Reddit. Specifically, we first truncate the graph structures of Ogbn-arxiv and Reddit by:

$$\mathbf{A}' = \sum_{k=1}^{K_1} \lambda_k \mathbf{u}_k \mathbf{u}_k^\top + \sum_{k=N-K_2+1}^{N} \lambda_k \mathbf{u}_k \mathbf{u}_k^\top \tag{17}$$

where $K_1 = r_k K$ and $K_2 = (1 - r_k)K$. We then gradually increase the value of K and train a 2-layer SGC on each truncated graph structure. The results are shown in Table 12.

We can observe that in Reddit, only 1,000 eigenvectors are enough to match the performance of the full graph (93.45 / 94.51 $\approx$ 98.9%), while in Ogbn-arxiv, a large number of eigenvectors (5,000) is required to approximate the full graph (69.22 / 70.02 $\approx$ 98.9%). Thus, we speculate that the structure information of Ogbn-arxiv is more widely distributed in the eigenbasis, making it challenging for GDEM to compress the entire distribution in synthetic data with an extremely small compression rate.

B. Theoretical Analysis of RSS for Gradient Matching

We further theoretically analyze whether the gradient matching method can preserve the restricted spectral similarity. Given x' and L' learned by the gradient matching method, we have:

$$\begin{aligned}
&\left|\mathbf{x}^\top \mathbf{L}\mathbf{x} - \mathbf{x}'^\top \mathbf{L}'\mathbf{x}'\right| \\
=&\left|\sum_{k=0}^{K} \lambda_k \mathbf{x}^\top \mathbf{u}_k \mathbf{u}_k^\top \mathbf{x} - \sum_{k=0}^{K} \lambda_k' \mathbf{x}'^\top \mathbf{u}_k' \mathbf{u}_k'^\top \mathbf{x}'\right| \\
=&\left|\left(\sum_{k=0}^{K} \lambda_k \mathbf{x}^\top \mathbf{u}_k \mathbf{u}_k^\top \mathbf{x} - \sum_{k=0}^{K} \lambda_k \mathbf{x}'^\top \mathbf{u}_k' \mathbf{u}_k'^\top \mathbf{x}'\right) + \left(\sum_{k=0}^{K} \lambda_k \mathbf{x}'^\top \mathbf{u}_k' \mathbf{u}_k'^\top \mathbf{x}' - \sum_{k=0}^{K} \lambda_k' \mathbf{x}'^\top \mathbf{u}_k' \mathbf{u}_k'^\top \mathbf{x}'\right)\right| \\
\leqslant& \sum_{k=0}^{K} \lambda_k \left|\mathbf{x}^\top \mathbf{u}_k \mathbf{u}_k^\top \mathbf{x} - \mathbf{x}'^\top \mathbf{u}_k' \mathbf{u}_k'^\top \mathbf{x}'\right| + \sum_{k=0}^{K} \left|\lambda_k - \lambda_k'\right| \left(\mathbf{x}'^\top \mathbf{u}_k' \mathbf{u}_k'^\top \mathbf{x}'\right)
\end{aligned} \tag{18}$$

Combining with Lemma 3.1, when the number of GCN layers goes to infinity, the objective optimization based on gradient matching is dominated by $\left|\mathbf{x}^\top \mathbf{u}_0 \mathbf{u}_0^\top \mathbf{x} - \mathbf{x}'^\top \mathbf{u}_0' \mathbf{u}_0'^\top \mathbf{x}'\right|$, while paying less attention to the optimization of $\left|\mathbf{x}^\top \mathbf{u}_k \mathbf{u}_k^\top \mathbf{x} - \mathbf{x}'^\top \mathbf{u}_k' \mathbf{u}_k'^\top \mathbf{x}'\right|$, when $k \neq 0$. Thus, gradient matching fails to constrain the first term of the upper bound of RSS. Moreover, gradient matching introduces spectrum bias causing $\lambda_k' \neq \lambda_k$, thus failing to constrain the second term of the upper bound. In summary, the gradient matching method is unable to preserve the restricted spectral similarity.

C. Graph Distiilation

Gradient Matching (Jin et al., 2022b;a) generates the synthetic graph and node features by minimizing the differences between model gradients on $\mathcal{G}$ and $\mathcal{G}'$, which can be formulated as:

$$\min_{\mathbf{A}',\mathbf{X}'} \mathop{\mathbb{E}}_{\theta\sim P_\theta} \left[D\left(\nabla_\theta \mathcal{L}\left(\Phi_\theta\left(\mathbf{A}',\mathbf{X}'\right),\mathbf{Y}'\right), \nabla_\theta \mathcal{L}\left(\Phi_\theta\left(\mathbf{A},\mathbf{X}\right),\mathbf{Y}\right)\right)\right], \tag{19}$$

where Φ_θ is the condensation GNNs with parameters θ, ∇_θ indicates the model gradients, D is a metric to measure their differences, and $\mathcal{L}$ is the loss function. For clarity, we omit the subscript that indicates the training data.

Distribution Matching (Liu et al., 2022a) aims to align the distributions of node representations in each GNN layer to generate the synthetic graph, which can be expressed as:

$$\min_{\mathbf{A}',\mathbf{X}'} \mathop{\mathbb{E}}_{\theta\sim P_\theta} \left[\sum_{t=1}^{L} D\left(\Phi_\theta^t\left(\mathbf{A}',\mathbf{X}'\right), \Phi_\theta^t\left(\mathbf{A},\mathbf{X}\right)\right)\right], \tag{20}$$

where Φ_θ^t is the t-th layer in GNNs.

Trajectory Matching (Zheng et al., 2023) aligns the long-term GNN learning behaviors between the original graph and the synthetic graph:

$$\min_{\mathbf{A}',\mathbf{X}'} \mathop{\mathbb{E}}_{\theta_t^{*,i}\sim P_{\Theta\mathcal{T}}} \left[\mathcal{L}_{\text{meta-tt}}\left(\theta_t^*|_{t=t_0}^{p}, \tilde{\theta}_t|_{t=t_0}^{q}\right)\right]. \tag{21}$$

where $\theta_t^*|_{t=t_0}^{p}$ and $\tilde{\theta}_t|_{t=t_0}^{q}$ is the parameters of $\text{GNN}_\mathcal{T}$ and $\text{GNN}_\mathcal{S}$, $\mathcal{L}_{\text{meta-tt}}$ calculates certain parameter training intervals within $\left[\theta_{t_0}^{*,i}, \theta_{t_0+p}^{*,i}\right]$ and $\left[\tilde{\theta}_{t_0}, \tilde{\theta}_{t_0+q}\right]$.

D. General Settings

Optimizer. We use the Adam optimizer for all experiments.

Environment. The environment in which we run experiments is:

- Linux version: 5.15.0-91-generic
- Operating system: Ubuntu 22.04.3 LTS
- CPU information: Intel(R) Xeon(R) Platinum 8358 CPU @ 2.60GHz
- GPU information: NVIDIA A800 80GB PCIe

Resources. The address and licenses of all datasets are as follows:

- Citeseer: https://github.com/kimiyoung/planetoid (MIT License)
- Pubmed: https://github.com/kimiyoung/planetoid (MIT License)
- Ogbn-arxiv: https://github.com/snap-stanford/ogb (MIT License)
- Flickr: https://github.com/GraphSAINT/GraphSAINT (MIT License)
- Reddit: https://github.com/williamleif/GraphSAGE (MIT License)
- Squirrel: https://github.com/benedekrozemberczki/MUSAE (GPL-3.0 license)
- Gamers: https://github.com/benedekrozemberczki/datasets (MIT License)