Faster Hyperparameter Search on Graphs via Calibrated Dataset Condensation

Anonymous Author(s)
Affiliation
Address
email

Abstract

Dataset condensation aims to reduce the computational cost of training multiple models on a large dataset by condensing the training set into a small synthetic one. State-of-the-art approaches rely on matching the gradients between the real and synthetic data and are recently applied to condense large-scale graphs for node classification tasks. Although dataset condensation may be efficient when we need to train multiple models for hyperparameter optimization, there is no theoretical guarantee on the generalizability of the condensed data, and it can *generalize poorly* across hyperparameters/architectures in practice; while on graphs, we find and prove this overfitting is much more severe. This paper considers a different condensation objective specifically for hyperparameter search. We aim to generate the synthetic dataset so that the validation-performance ranking of different models under different hyperparameters on the condensed and original datasets are comparable. We propose a novel *hyperparameter-calibrated dataset condensation* (HCDC) algorithm, which learns the synthetic validation data by matching the *hyperparameter gradients* computed by implicit differentiation and efficient inverse Hessian approximation. HCDC employs a supernet with differentiable hyperparameters, making it suitable for modeling GNNs with widely different convolution filters. Experiments demonstrate that the proposed framework effectively maintains the validation-performance rankings of GNNs and speeds up hyperparameter/architecture search on graphs.

1 Introduction

Graph neural networks (GNNs) have found remarkable success in tackling a variety of graph-related tasks [Hamilton, 2020]. However, the prevalence of large-scale graphs in real-world contexts, such as social, information, and biological networks [Hu et al., 2020], which frequently scale up to millions of nodes and edges, poses significant computational issues for training GNNs. While training a single model can be expensive, designing deep learning models for new tasks require substantially more computations, as they involve training multiple models on the same dataset many times to verify the design choices, such as architectures and hyperparameters [Elsken et al., 2019]. We ask: *how can we reduce the computational cost for training multiple models on the same dataset, for hyperparameter search/optimization*?

A natural approach is to reduce the training set size through approaches such as graph coreset selection [Baker et al., 2020], graph sparsification [Batson et al., 2013], graph coarsening [Loukas, 2019] and graph sampling [Zeng et al., 2019]. However, these methods are restricted to selecting samples from the given ones, limiting their performance upper-bound. A more effective alternative is to *synthesize* informative samples rather than select from given samples. *Dataset condensation* [Zhao et al., 2020] has emerged as a competent data synthesizing mechanism with promising results. It aims

Submitted to 36th Conference on Neural Information Processing Systems (NeurIPS 2022). Do not distribute.

to learn a small synthetic training set such that a model trained on the synthetic set obtains testing accuracy comparable to that trained on the original training set.

Although *dataset condensation* achieves the state-of-the-art in terms of the performance for neural networks trained on the condensed samples, it is a *unreliable and sub-optimal* solution to our question, the goal of speeding up training for hyperparameter search/optimization: (1) theoretically, dataset condensation learns synthetic samples that minimize the performance drop of a specific model, and there is no performance guarantee when we train other models; and (2) in practice, we rarely compare condensation methods with strong baselines such as various coreset methods, in terms of their ability to preserve the outcome of architecture/hyperparameter optimization. In this paper, we identify the poor generalizability of the condensed data on graphs [Jin et al., 2021] across architectures/hyperparameters, which has been overlooked when applied for image condensation. Not only we observe that graph condensation fails to preserve validation-performance ranking of GNN architectures, but also we identify and prove two dominant effects causing this failure: (1) most GNNs differ from each other by their design of convolution filters, thus the convolution filter used during condensation is a single biased point in "the space of convolution filters"; and (2) the learned adjacency matrix of the synthetic graph easily overfits the condensation objective, thus fails to maintain the characteristics of the original structure and distinguish different architectures.

We aim to *develop a new dataset condensation framework that preserves the outcome of hyperparameter search/optimization on the condensed data*. In addition, to condense the training data, we propose to learn *the validation split* of synthetic data such that the validation-performance ranking of architectures on the condensed and original datasets are comparable. Similar to the standard dataset condensation, this new objective can be written as a bi-level optimization problem. Inspired by the gradient-matching algorithm in [Zhao et al., 2020], if assuming a continuous hyperparameter space or a generic supernet which interpolates all architectures, we find and prove the validation-performance-ranking-preserving goal can be realized by matching the *hyperparameter-gradients* on the synthetic and real validation data. The hyperparameter-gradients (or *hypergradients* for short) can be efficiently computed with constant memory overhead by the implicit function theorem (IFT) and the Neumann series approximation of inverse Hessian [Lorraine et al., 2020].

The proposed *hyperparameter-calibrated dataset condensation* (HCDC) framework assumes *continuous* hyperparameters, which is suitable to model GNNs with different convolution matrices and save the problematic generalizability of graph condensation across GNNs. Although beyond the scope of this paper, HCDC also has the potential to be combined with the supernets in differentiable NAS methods [Liu et al., 2018] to tackle the discrete neural architecture space, which is the primary concern of NAS on image and text data. Experiments demonstrate the effectiveness of the proposed framework in preserving the performance rankings of GNNs. Our distilled graph can be used as proxy data for off-the-shelf graph neural architecture search algorithms to accelerate the search process.

Our contributions can be summarized as follows: **(1)** We formulate a new dataset condensation objective specifically for hyperparameter optimization and propose the *hyperparameter-calibrated dataset condensation* (HCDC) framework, which learns the synthetic validation data by matching the hyperparameter gradients. **(2)** We prove the hardness of generalizing the condensed graph across GNN architectures.**(3)** Experiments demonstrate the effectiveness of HCDC to speed up architecture search on graphs when combined with off-the-shelf graph neural architecture search algorithms.

2 Preliminaries

This paper adopts graph learning notations, but HCDC is generally applicable to other data and tasks. The typical downstream task on graphs is node classification. **Node classification on graph** considers that we are given a graph $\mathcal{T} = (A, X, \mathbf{y})$ with adjacency matrix $A \in \{0,1\}^{n\times n}$, node features $X \in \mathbb{R}^{n\times d}$, node class labels $\mathbf{y} \in [K]^n$, and mutually disjoint node-splits $V_{train} \bigcup V_{val} \bigcup V_{test} = [n]$. Using a GNN model $f_{\theta,\psi} : \mathbb{R}^{n\times n}_{\geq 0} \times \mathbb{R}^{n\times d} \to \mathbb{R}^{n\times K}$, where θ is parameter and ψ is hyperparameter, we aim to find $\theta^{\mathcal{T}} = arg\min_\theta \mathcal{L}^{train}_{\mathcal{T}}(\theta, \psi)$, with cross-entropy loss $\mathcal{L}^{train}_{\mathcal{T}}(\theta, \psi) = \sum_{i\in V_{train}} \ell\big([f_{\theta,\psi}(A, X)]_i, y_i\big)$. The transductive setting can be easily generalized to the inductive setting by assuming only $\{A_{ij} \mid i, j \in V_{train}\}$ and $\{X_i \mid i \in V_{train}\}$ are used during training.

Now, we start from reviewing the *standard dataset condensation* (SDC) and its natural *bilevel optimization* (BL) formulation [Wang et al., 2018].

Standard Dataset Condensation (SDC) aims to find a synthetic graph $\mathcal{S} = (A', X', \mathbf{y}')$ of size $c \ll n$, with (weighted) adjacency matrix $A' \in \mathbb{R}_{\geq 0}^{c \times c}$, node features $X' \in \mathbb{R}^{c \times d}$, node labels $\mathbf{y}' \in [K]^c$, and (possibly) train/validation[1] splits $V'_{train} \bigcup V'_{val} = [c]$. The goal of dataset condensation is to obtain *comparable generalization performance* on real graph by training on the condensed graph, i.e., $\mathcal{L}_{\mathcal{T}}^{test}(\theta^{\mathcal{T}}, \psi) \approx \mathcal{L}_{\mathcal{T}}^{test}(\theta^{\mathcal{S}}, \psi)$ where $\theta^{\mathcal{S}} = arg\min_\theta \mathcal{L}_{\mathcal{S}}^{train}(\theta, \psi)$ is optimized on the synthetic graph. By posing $\theta^{\mathcal{S}}$ as a function of the condensed graph $\mathcal{S}$, dataset condensation can be formulated as a *bilevel optimization* problem,

$$\mathcal{S}^* = \arg\min_{\mathcal{S}} \mathcal{L}_{\mathcal{T}}^{train}(\theta^{\mathcal{S}}(\mathcal{S}), \psi) \quad \text{s.t.} \quad \theta^{\mathcal{S}}(\mathcal{S}) = \arg\min_{\theta} \mathcal{L}_{\mathcal{S}}^{train}(\theta, \psi) \tag{SDC-BL}$$

However, the above problem involves a nested-loop optimization and solving the inner loop for $\theta^{\mathcal{S}}(\mathcal{S})$ at each iteration to recover the gradients for $\mathcal{S}$ requires a computationally expensive procedure: unrolling the recursive computation graph for $\mathcal{S}$ over multiple optimization steps of θ.

Zhao et al. [2020] alleviate the computational issue by its *gradient-matching* (GM) formulation. To start with, assuming neural network $f_{\theta,\psi}$ is a *locally smooth function*, and thus similar weights $\theta^{\mathcal{S}} \approx \theta^{\mathcal{T}}$ imply similar mappings in a local neighborhood and thus generalization performance. Then one can formulate the condensation objective as matching the optimized parameters (which depends on initialization θ_0), i.e., finding $\mathcal{S}^* = \arg\min_{\mathcal{S}} \mathbb{E}_{\theta_0 \sim P_{\theta_0}} \big[D\big(\theta^{\mathcal{S}}(\mathcal{S}, \theta_0), \theta^{\mathcal{T}}(\theta_0)\big) \big]$ s.t. $\theta^{\mathcal{S}}(\mathcal{S}, \theta_0) = \arg\min_\theta \mathcal{L}_{\mathcal{S}}^{train}\big(\theta(\theta_0), \psi\big)$ where $\theta^{\mathcal{T}}(\theta_0) = \arg\min_\theta \mathcal{L}_{\mathcal{T}}^{train}\big(\theta(\theta_0), \psi\big)$ and $D(\cdot, \cdot)$ is a distance function.

The *parameter-matching* problem is still a bilevel optimization but can be simplified with several approximations. Firstly, $\theta^{\mathcal{S}}(\mathcal{S}, \theta_0)$ is approximated by the output of an incomplete gradient-descent optimization, $\theta^{\mathcal{S}}(\mathcal{S}, \theta_0) \approx \theta_{t+1}^{\mathcal{S}} \leftarrow \theta_t^{\mathcal{S}} - \eta \nabla_\theta \mathcal{L}_{\mathcal{S}}^{train}(\theta_t^{\mathcal{S}}, \psi)$. However, the target parameter $\theta^{\mathcal{T}}(\theta_0) = \arg\min_\theta \mathcal{L}_{\mathcal{T}}^{train}\big(\theta(\theta_0), \psi\big)$ may be far away from $\theta_{t+1}^{\mathcal{S}}$. Zhao et al. [2020] propose to match $\theta_{t+1}^{\mathcal{S}}$ with incompletely optimized $\theta_{t+1}^{\mathcal{T}} \leftarrow \theta_t^{\mathcal{T}} - \eta \nabla_\theta \mathcal{L}_{\mathcal{T}}^{train}(\theta_t^{\mathcal{T}}, \psi)$ at each iteration t, and the condensation objective is now $\mathcal{S}^* = \arg\min_{\mathcal{S}} \mathbb{E}_{\theta_0 \sim P_{\theta_0}} \Big[\sum_{t=0}^{T-1} D(\theta_t^{\mathcal{S}}, \theta_t^{\mathcal{T}}) \Big]$.

Starting from the common initialization θ_0 and up to iteration t, if $\theta_t^{\mathcal{S}}$ can always track $\theta_t^{\mathcal{T}}$ by optimizing $\mathcal{S}$, i.e., $\theta_t^{\mathcal{S}} \approx \theta_t^{\mathcal{T}}$. For the one step update, we can replace $D(\theta_{t+1}^{\mathcal{S}}, \theta_{t+1}^{\mathcal{T}})$ by $D\big(\nabla_\theta \mathcal{L}_{\mathcal{S}}^{train}(\theta_t^{\mathcal{S}}, \psi), \nabla_\theta \mathcal{L}_{\mathcal{T}}^{train}(\theta_t^{\mathcal{T}}, \psi)\big) \approx D\big(\nabla_\theta \mathcal{L}_{\mathcal{S}}^{train}(\theta_t^{\mathcal{S}}, \psi), \nabla_\theta \mathcal{L}_{\mathcal{T}}^{train}(\theta_t^{\mathcal{S}}, \psi)\big)$. Repeating this inductive argument, the condensation objective is approximated by matching the gradients at each iteration t,

$$\mathcal{S}^* = \arg\min_{\mathcal{S}} \mathbb{E}_{\theta_0 \sim P_{\theta_0}} \Big[\sum_{t=0}^{T-1} D\big(\nabla_\theta \mathcal{L}_{\mathcal{S}}^{train}(\theta_t^{\mathcal{S}}, \psi), \nabla_\theta \mathcal{L}_{\mathcal{T}}^{train}(\theta_t^{\mathcal{S}}, \psi)\big) \Big] \tag{SDC-GM}$$

We now have a single deep network with parameters θ trained on the condensed graph $\mathcal{S}$. While $\mathcal{S}$ is optimized such that the distance between the gradient vectors of $\mathcal{L}_{\mathcal{T}}^{train}$ and of $\mathcal{L}_{\mathcal{S}}^{train}$ w.r.t. the parameters θ is minimized. Cosine distance $D(\cdot, \cdot) = cos(\cdot, \cdot)$ works well in practice.

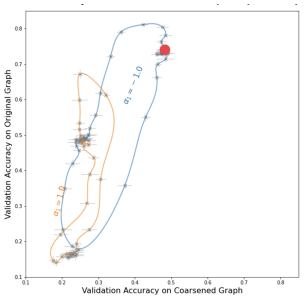
3 Standard Dataset Condensation Is Problematic Across GNNs

Despite its success in preserving the model performance when trained on the condensed dataset, the *gradient-matching* algorithm *naturally overfits* the model $f_{\theta,\psi}$ used during condensation and generalizes poorly to others. There is no guarantee that the condensed synthetic data $\mathcal{S}^*$ which minimizes the objective (Eq. (SDC-GM)) for a specific model $f_{\theta,\psi}$ (marked by its hyperparameter ψ) can generalize well to other models $f_{\theta,\psi'}$ where $\psi' \neq \psi$.

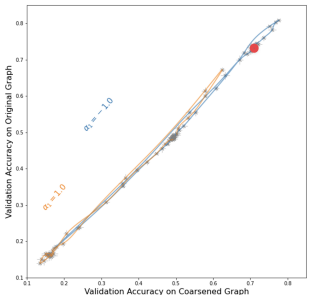
We find this overfitting issue could be much more *severe on graphs*. For the ease of theoretical analysis[2], we consider the simple linear regression problem with *linear convolution* models in this section, $f_{\theta=[\theta_C,\theta_W]}(A, X) = C(A, \theta_C)XW(\theta_W)$, where $C(A; \theta_C)$ is the *convolution matrix* which has the same sizes as the adjacency matrix A and possibly also depends on the parameters $\theta_C \in \mathbb{R}^p$, and $W(\theta_W)$ is the learnable linear weight matrix which is a reshape of the parameters $\theta_W \in \mathbb{R}^d$. The loss is now sum of squares $\mathcal{L}(\theta, \psi) = \|\mathbf{y} - CXW\|_2^2$ (train-split subscript omitted) where the labels $\mathbf{y}$ are continuous.

[1] The validation split of synthetic data is only required by our HCDC; see Eq. (SDC-BL) vs. Eq. (DCHPO).

[2] Assuming convex loss and linear models still reflects the general generalization issue; see Appendix A.2.



(a) $c = 0.2n$



(b) $c = 0.8n$

Figure 1: The manifold of GNNs with convolution filters $C = I + \alpha_1(\hat{L}) + \alpha_2(2\hat{L} - I)$ ($\hat{L}$ defined in ChebNet; see Appendix A.3) projected to the plane of validation accuracy on condensed (x-axis) and original (y-axis) graphs under two ratios c/n on Cora. $C = I + \hat{L}$ (red dot) is a biased point in this model space.

Ratio (c/n)	A' learned	$A' = I_c$
0.05%	61.3 ± 0.5	59.2 ± 1.1
0.25%	64.2 ± 0.4	63.2 ± 0.3

(a) Graph condensation with identity adjacency $A' = I_c$.

C\T	GCN	SGC ($K = 2$)	GIN
GCN	60.3 ± 0.3	59.2 ± 0.7	42.2 ± 4.3
SGC	59.2 ± 1.1	60.5 ± 0.6	39.0 ± 7.1
GIN	—	—	59.1 ± 1.1

(b) Generalization across GNNs.

Table 1: Empirical results on Ogbn-arxiv verifying the two effects (Propositions 2 and 3) that hinders the generalization of condensed graph across GNNs. (a) Adjacency overfitting, (b) Convolution mismatch.

This *linear convolution* model generalizes a wide variety of GNNs [Balcilar et al., 2021, Ding et al., 2021]; see Appendix A.3. For example, the convolution matrix of graph convolution network (GCN) [Kipf and Welling, 2016] is $C(A) = \tilde{D}^{-\frac{1}{2}}\tilde{A}\tilde{D}^{-\frac{1}{2}}$ where $\tilde{A}$ and $\tilde{D}$ are the self-loop-added adjacency and degree matrix. It also generalizes the one-dimensional *convolution neural network* (1D-CNN) (with one channel), where the convolution matrix of kernel size $p = (2K + 1)$ is $C(\theta_C) = \sum_{k=-K}^{k=K} [\theta_C]_k P^k$ and P is the cyclic permutation matrix correspond to a unit shift.

We say the *gradient-matching objective is satisfied on a non-degenerate trajectory*, if there exists a fixed learning trajectory $(\theta_t^{\mathcal{S}})_{t=0}^{T-1}$ which span the whole parameter space, i.e., $\text{span}(\theta_0^{\mathcal{S}}, \ldots, \theta_{T-1}^{\mathcal{S}}) = \mathbb{R}^{p+d}$, such that the *gradient-matching* loss on this trajectory (the objective of Eq. (SDC-GM) without expectation) is 0. The *validity* of *standard dataset condensation* (SDC) can be readily verified; see Proposition 4 in Appendix A.1, where we show if the gradient-matching objective is satisfied on a non-degenerate trajectory, the optimizer on the condensed dataset $\mathcal{S}$ is also optimal on the original dataset $\mathcal{T}$.

However, as on the generalizability of condensed dataset across models, we obtain **contrary results for 1D-CNNs and GNNs**.

Proposition 1 (Successful Generalization of SDC across 1D-CNNs)**.** *Assuming least-square regression with one-dimensional linear convolution* $f^{2K+1}_{\theta=[\theta_C,\theta_W]}(X) = (\sum_{k=-K}^{k=K} [\theta_C]_k P^k) X W(\theta_W)$, *where kernel size is* $(2K + 1), K \geq 0$, *if the gradient-matching objective is satisfied on a non-degenerate trajectory for* f^{2K+1}, *then the condensed dataset* $\mathcal{S}^*$ *still satisfies the gradient-matching objective on any trajectories* $(\theta_t^{\mathcal{S}})_{t=0}^{T-1}$ *for any linear convolution* $f^{2K'+1}$ *with kernel size* $(2K' + 1), K \geq K' \geq 0$.

The intuition behind Proposition 1 is that the 1D-CNN of kernel size $(2K + 1)$ is a "supernet" of the 1D-CNN of kernel size $(2K' + 1)$ if $K' \leq K$, and the condensed dataset via a bigger model can generalize well to smaller ones. This result suggests us to *use a sufficiently large model during condensation*, to enable the generalization of the condensed dataset to a wider range of models.

However, the story for GNNs is vastly different. We find there are *two dominant effects* causing the condensed graph to fail to generalize across GNNs. Firstly, the learned adjacency A' of the synthetic graph $\mathcal{S}$ can easily *overfit* the condensation objective, thus failing to maintain the characteristics of the original structure and distinguish different architectures; see Table 1 for relevant experiments.

Proposition 2 (Condensed Adjacency Overfits SDC Objective)**.** *Assuming least-square regression with a linear GNN,* $f_\theta(A, X) = C(A)XW(\theta)$. *For any synthetic node features* $X' \in \mathbb{R}^{c \times d}$, *there exists a synthetic adjacency matrix* $A' \in \mathbb{R}^{c \times c}_{\geq 0}$ *such that the gradient-matching objective is satisfied on any trajectories.*

Secondly, GNNs differ from each other mostly on the design of convolution $C(A)$, i.e., how the convolution weights C depend on the adjacency information A. The convolution filter $C(A)$ used during

condensation is a single biased point in "the space of convolutions"; see Fig. 1 for a visualization, thus there is a *mismatch* of inductive bias when transferring to a different GNN. These two effects lead to the following hardness results when transferring the condensed graph across GNNs.

Proposition 3 (Failed Generalization of SDC across GNNs)**.** *Assuming least square regression and linear GNN, $f_\theta^C(A,X) = C(A)XW(\theta)$, there always exists a condensed graph $\mathcal{S}^*$, such that the gradient-matching objective is satisfied on any trajectories for f^C. However, if we train a new linear GNN $f_\theta^{\mathfrak{C}}(A,X)$ with convolution matrix $\mathfrak{C}(A')$ on $\mathcal{S}^*$, the relative error between the optimized model parameters of $f^{\mathfrak{C}}$ on the real and condensed graphs is $\|\theta_{\mathfrak{C}}^{\mathcal{S}} - \theta_{\mathfrak{C}}^{\mathcal{T}}\|/\|\theta_{\mathfrak{C}}^{\mathcal{T}}\| \geq \max\{\sigma_{\max}(Q) - 1, 1-\sigma_{\min}(Q)\}$, where $\theta_{\mathfrak{C}}^{\mathcal{T}} = \arg\min_\theta \|\mathbf{y} - f_\theta^{\mathfrak{C}}(A,X)\|_2^2$, $\theta_{\mathfrak{C}}^{\mathcal{S}} = \arg\min_\theta \|\mathbf{y}' - f_\theta^{\mathfrak{C}}(A',X')\|_2^2$, and $Q = \left(X^\top [C(A)]^\top [C(A)]X\right)\left(X^\top [\mathfrak{C}(A)]^\top [\mathfrak{C}(A)]X\right)^{-1}$.*

Proposition 3 provides a effective lower-bound on the relative estimation error of optimal model parameter, when a different convolution filter $\mathfrak{C}(\cdot) \neq C(\cdot)$ is used[3]. According to the spectral characterization of convolution filters of GNNs (Table 1 of [Balcilar et al., 2021]), we can approximately compute the maximum eigenvalue of Q for some GNNs. For example, if we condense with f^C graph isomorphism network (GIN-0) [Xu et al., 2018] but train $f^{\mathfrak{C}}$ GCN on the condensed graph, we have $\|\theta_{\mathfrak{C}}^{\mathcal{S}} - \theta_{\mathfrak{C}}^{\mathcal{T}}\|/\|\theta_{\mathfrak{C}}^{\mathcal{T}}\| \gtrapprox \overline{\text{deg}} + 1$ where $\overline{\text{deg}}$ is the average node degree of the original graph. This large lower bound hints the catastrophic failure when transfer across GIN and GCN; see Table 1.

Although the results above are obtained for least squares loss and linear convolution model, it *still reflects the nature of general non-convex losses and non-linear models*. Since dataset condensation is effectively matching the local minima $\{\theta^{\mathcal{T}}\}$ of the original loss $\mathcal{L}_{\mathcal{T}}^{train}(\theta,\psi)$ with the local minima $\{\theta^{\mathcal{S}}\}$ of the condensed loss $\mathcal{L}_{\mathcal{S}}^{train}(\theta,\psi)$, within the small neighborhoods surrounding the pair of local minima $(\theta^{\mathcal{T}}, \theta^{\mathcal{S}})$, we can approximate the non-convex loss and non-linear model with a convex/linear one respectively. Hence the generalizability issues with convex loss and liner model may hold.

4 Hyperparameter-calibrated Dataset Condensation Objective

Our goal is to develop an optimal and reliable condensation method for architecture/hyperparameter search. Standard Dataset Condensation objective (Eq. (SDC-BL)/Eq. (SDC-GM)) does not accomplish this goal since it does not generalize across GNNs, as proven in Section 3. In this section, we propose a new condensation objective specifically for *preserving the outcome of hyperparameter optimization* (HPO) on the condensed dataset.

HPO Objective HPO finds the optimal hyperparameter $\psi^{\mathcal{T}}$ such that the corresponding model $f_{\theta,\psi^{\mathcal{T}}}$ minimizes the validation loss after training, i.e.,

$$\psi^{\mathcal{T}} = \arg\min_{\psi\in\Psi} \mathcal{L}_{\mathcal{T}}^{val}(\theta^{\mathcal{T}}(\psi),\psi) \quad \text{s.t.} \quad \theta^{\mathcal{T}}(\psi) = \arg\min_{\theta} \mathcal{L}_{\mathcal{T}}^{train}(\theta,\psi) \tag{HPO}$$

We see HPO itself is a bilevel optimization, where the optimal parameter $\theta^{\mathcal{T}}(\psi)$ is posed as a function of the hyperparameter ψ.

Dataset Condensation for HPO Objective If both the train and validation splits are defined on the condensed dataset $\mathcal{S}$, the optimal hyperparameter $\psi^{\mathcal{S}}$ is well-defined. Our goal is to find the synthetic dataset $\mathcal{S}$ such that we can obtain *comparable validation performance* if the hyperparameters are optimized on the condensed dataset, i.e., $\mathcal{L}_{\mathcal{T}}^{val}(\theta^{\mathcal{T}}(\psi^{\mathcal{T}}),\psi^{\mathcal{T}}) \approx \mathcal{L}_{\mathcal{T}}^{val}(\theta^{\mathcal{T}}(\psi^{\mathcal{S}}),\psi^{\mathcal{S}})$. Clearly, this goal looks very similar to the goal of standard dataset condensation, preserving generalization performance $\mathcal{L}_{\mathcal{T}}^{test}(\theta^{\mathcal{T}},\psi) \approx \mathcal{L}_{\mathcal{T}}^{test}(\theta^{\mathcal{S}},\psi)$, which hints us to formulate the new objective as a bilevel optimization problem too,

$$\mathcal{S}^* = \arg\min_{\mathcal{S}} \mathcal{L}_{\mathcal{T}}^{val}\left(\theta^{\mathcal{T}}(\psi^{\mathcal{S}}(\mathcal{S})),\psi^{\mathcal{S}}(\mathcal{S})\right) \quad \text{s.t.} \quad \psi^{\mathcal{S}}(\mathcal{S}) = \arg\min_{\psi\in\Psi} \mathcal{L}_{\mathcal{S}}^{val}\left(\theta^{\mathcal{S}}(\psi),\psi\right) \tag{DCHPO}$$

where $\theta^{\mathcal{T}}(\psi)$ and $\theta^{\mathcal{S}}(\psi)$ are defined following Eq. (HPO).

However, this formulation (Eq. (DCHPO)) is a nested optimization (for dataset condensation) over another nested optimization (for HPO) which necessitates very high order gradients and is challenging to solve. Moreover, we have largely overlooked another important factor of hyperparameter optimization, the search space/feasible set of hyperparameters Ψ.

[3] If $\mathfrak{C}(\cdot) = C(\cdot)$ Proposition 4 guarantees $\theta_C^{\mathcal{S}} = \theta_C^{\mathcal{T}}$ and the lower bound in Proposition 3 is 0.

In contrast to parameter optimization, where the search space is usually assumed to be the continuous and unbounded Euclidean space, the search space of hyperparameters Ψ can be either a discrete set $\Psi = \{\psi_1, \ldots\} = \Psi_1 \times \cdots \times \Psi_p$ (where each hyperparameter vector $\psi \in \Psi$ consists of p discrete hyperparameters of various types, for example, neural network type, width, depth, batch size, etc) or a small bounded set of continuous hyperparameters around its optimum (for example, learning rate, dropout rate, sample weights, etc). Often we face compositions of these discrete- and continuous-natured hyperparameters, and we can either model them all as discrete ones and search by grid search, Bayesian optimization, and reinforcement learning; or relax the discrete search space to a continuous one.

Hyperparameter-Calibration: A Sufficient Alternative to Dataset Condensation for HPO The finiteness/boundedness nature of the search space Ψ cast another challenge to the dataset condensation for HPO. To avoid the complex combinatorial/constrained optimization in Eq. (HPO), we ask: *Is is possible to preserve the outcome of HPO without solving HPO (Eq. (HPO)) directly*? In this spirit, we consider a sufficient alternative condition to preserve the outcome of HPO on Ψ.

Definition 1 (Hyperparameter-Calibration)**.** *Given original dataset $\mathcal{T}$, generic model $f_{\theta,\psi}$, and hyperparameter search space Ψ, we say a condensed dataset $\mathcal{S}$ is hyperparameter-calibrated, if for any $\psi_1 \neq \psi_2 \in \Psi$, it holds that,*

$$\big(\mathcal{L}_{\mathcal{T}}^{val}(\theta^{\mathcal{T}}(\psi_1), \psi_1) - \mathcal{L}_{\mathcal{T}}^{val}(\theta^{\mathcal{T}}(\psi_2), \psi_2)\big)\big(\mathcal{L}_{\mathcal{S}}^{val}(\theta^{\mathcal{S}}(\psi_1), \psi_1) - \mathcal{L}_{\mathcal{S}}^{val}(\theta^{\mathcal{S}}(\psi_2), \psi_2)\big) > 0 \qquad \text{(HC)}$$

that is, changes of validation loss on $\mathcal{T}$ and $\mathcal{S}$ always have the same sign, where $\theta^{\mathcal{T}}(\psi) = \arg\min_\theta \mathcal{L}_{\mathcal{T}}^{train}(\theta, \psi)$ denotes the parameters optimized on the training split of $\mathcal{T}$ with hyperparameter ψ, similar for $\theta^{\mathcal{S}}(\psi)$.

It is clear that if hyperparameter calibration (HC) is satisfied, HPO on the original and condensed datasets yields the same result. Therefore, our mission changes to *how to ensure hyperparameter-calibration for a single pair of hyperparameters* (ψ_1, ψ_2)?

HCDC: Hypergradient-alignment Objective To proceed, we make an important extra assumption that the (possibly discrete) search space Ψ can be extended to a compact and connected set $\Psi' \supset \Psi$, where we can define continuation of the generic model $f_{\theta,\psi}$ on Ψ' so that $f_{\theta,\psi}$ is differentiable anywhere in Ψ'. Such a continual extension naturally exists on graphs (see Section 5) or can be provided by differentiable NAS approaches; see Section 6.

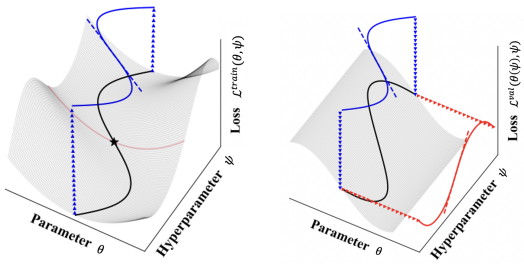
Now, if we consider the special case where ψ_1 is within the neighborhood of ψ_2, i.e., $\psi_1 \in B_r(\psi_2)$ for some $r > 0$, and reparameterize $\psi_1 = \psi + \Delta\psi$, $\psi_2 = \psi$ with $r \geq \|\Delta\psi\|_2 \to 0^+$. The change in validation loss is approximated *up to first-order* by the hyperparameter-gradients (hypergradients for short) $\big(\mathcal{L}_{\mathcal{T}}^{val}(\theta^{\mathcal{T}}(\psi_1), \psi_1) - \mathcal{L}_{\mathcal{T}}^{val}(\theta^{\mathcal{T}}(\psi_2), \psi_2)\big) \approx \nabla_\psi \mathcal{L}_{\mathcal{T}}^{val}(\theta^{\mathcal{T}}(\psi), \psi) \cdot \Delta\psi$. The hyperparameter-calibration condition within this tiny neighborhood $B_r(\psi)$ is then simplified to $\nabla_\psi \mathcal{L}_{\mathcal{T}}^{val}(\theta^{\mathcal{T}}(\psi), \psi) // \nabla_\psi \mathcal{L}_{\mathcal{S}}^{val}(\theta^{\mathcal{S}}(\psi), \psi)$, i.e., the two hypergradient vectors are aligned (i.e., pointing same direction).

Considering the extended search space Ψ' can be covered by the union of many small neighborhoods, we derive the hypergradient-alignment condition: $\nabla_\psi \mathcal{L}_{\mathcal{T}}^{val}(\theta^{\mathcal{T}}(\psi), \psi) // \nabla_\psi \mathcal{L}_{\mathcal{S}}^{val}(\theta^{\mathcal{S}}(\psi), \psi)$ for any $\psi \in \Psi'$. It is not hard to show that the condition above is equivalent to hyperparameter-calibration (Definition 1) on a connected and compact set Ψ'. (1) *Necessity* proved by contradiction. If there exists $\psi_0 \in \Psi'$ such that the two gradient vectors are not aligned at ψ_0, then there exists small perturbation $\Delta\psi_0$ such that $\big(\mathcal{L}_{\mathcal{T}}^{val}(\theta^{\mathcal{T}}(\psi_0 + \Delta\psi_0), \psi_0 + \Delta\psi_0) - \mathcal{L}_{\mathcal{T}}^{val}(\theta^{\mathcal{T}}(\psi_0), \psi_0)\big)$ and $\big(\mathcal{L}_{\mathcal{S}}^{val}(\theta^{\mathcal{S}}(\psi_0 + \Delta\psi_0), \psi_0 + \Delta\psi_0) - \mathcal{L}_{\mathcal{S}}^{val}(\theta^{\mathcal{S}}(\psi_0), \psi_0)\big)$ have different signs. (2) *Sufficiency* proved by integration. For any pair $\psi_1 \neq \psi_2 \in \Psi'$, if we have a continuous path connecting ψ_1 and ψ_2, then integrating hypergradients $\nabla_\psi \mathcal{L}_{\mathcal{T}}^{val}(\theta^{\mathcal{T}}(\psi), \psi)$ and $\nabla_\psi \mathcal{L}_{\mathcal{S}}^{val}(\theta^{\mathcal{S}}(\psi), \psi)$ through the path recovers the hyperparameter-calibration condition.

In this regard, enforcing hypergradient-alignment on Ψ' is sufficient to hyperparameter calibration on Ψ, thus ensuring the outcome of HPO over Ψ is preserved. The hypergradient-alignment objective below realizes hyperparameter-calibrated dataset condensation (HCDC).

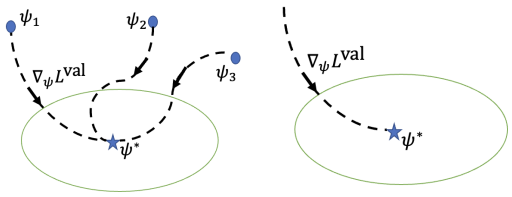
$$\mathcal{S}^* = \arg\min_{\mathcal{S}} \sum_{\psi \in \Psi'} D\big(\nabla_\psi \mathcal{L}_{\mathcal{T}}^{val}(\theta^{\mathcal{T}}(\psi), \psi), \nabla_\psi \mathcal{L}_{\mathcal{S}}^{val}(\theta^{\mathcal{S}}(\psi), \psi)\big) \qquad \text{(HCDC)}$$

where cosine distance $D(\cdot, \cdot) = cos(\cdot, \cdot)$ is used.



(a) train loss (b) validation loss

Figure 2: The parameter-hyperparameter manifolds and IFT. The blue solid line is the best response $\theta^*(\psi)$. The red dashed line is hypergradients $\nabla_\psi \mathcal{L}^{val}(\theta(\psi),\psi)$.



(a) Discrete $\Psi = \{\psi_i\}$ (b) Cont. Ψ around ψ^*

Figure 3: Where to align the hypergradients in HCDC (Eq. (HCDC)); see Section 5 for explanations.

5 Implementation of HCDC

Finally, we work on implementing and simplifying the hyperparameter-calibrated dataset condensation (HCDC) objective and apply it to the graph architecture/hyperparameter search problem.

How is HCDC connected to standard dataset condensation (SDC) (Eq. (SDC-GM))? Theoretically speaking, the objective of HCDC, preserving the outcome of hyperparameter optimization (HPO), is orthogonal to the objective of SDC, preserving generalization performance. Therefore, we can limit the part of the synthetic dataset they optimize to make the two algorithms completely independent. While SDC only learns the training split of $\mathcal{S}$, we restrict HCDC to only optimize the validation split of $\mathcal{S}$ in Eq. (HCDC) and keep the training split fixed[4]. Nevertheless, we need to find the condensed training data $\mathcal{S}^{train}$ before HCDC, and this can be done by all kinds of approaches, from uniform sampling to SDC.

How to compute hypergradients and optimize the hypergradient-alignment loss in Eq. (HCDC)? The efficient computation of hypergradients $\nabla_\psi \mathcal{L}_\mathcal{T}^{val}(\theta^\mathcal{T}(\psi),\psi)$ and $\nabla_\psi \mathcal{L}_\mathcal{S}^{val}(\theta^\mathcal{S}(\psi),\psi)$ uses the implicit function theorem (IFT) (see Section 4 for visualization), $\nabla_\psi \mathcal{L}_\mathcal{T}^{val}(\theta^\mathcal{T}(\psi),\psi) = -\left[\frac{\partial^2 \mathcal{L}_\mathcal{T}^{train}(\theta,\psi)}{\partial\psi\partial\theta^T}\right]\left[\frac{\partial^2 \mathcal{L}_\mathcal{T}^{train}(\theta,\psi)}{\partial\theta\partial\theta^T}\right]^{-1}\nabla_\theta \mathcal{L}_\mathcal{T}^{val}(\theta,\psi) + \nabla_\psi \mathcal{L}_\mathcal{T}^{val}(\theta,\psi)$, where $\nabla_\psi \mathcal{L}_\mathcal{T}^{val}(\theta,\psi)$ is the direct gradient and often identically 0. The first term is the product of inverse training Hessian $\left[\frac{\partial^2 \mathcal{L}_\mathcal{T}^{train}(\theta,\psi)}{\partial\theta\partial\theta^T}\right]^{-1}$, the training mixed partials $\left[\frac{\partial^2 \mathcal{L}_\mathcal{T}^{train}(\theta,\psi)}{\partial\psi\partial\theta^T}\right]$ and the validation gradients $\nabla_\theta \mathcal{L}_\mathcal{T}^{val}(\theta,\psi)$. While the other parts can be computed by back-propagation, the inverse Hessian needs to be approximated. Instead of using conjugate gradient method, Lorraine et al. [2020] propose a stable, tractable and efficient Neumann series approximation, $\left[\frac{\partial^2 \mathcal{L}_\mathcal{T}^{train}(\theta,\psi)}{\partial\theta\partial\theta^T}\right]^{-1} = \lim_{i\to\infty}\sum_{j=0}^{i}\left[I - \frac{\partial^2 \mathcal{L}_\mathcal{T}^{train}(\theta,\psi)}{\partial\theta\partial\theta^T}\right]^j$ with constant memory constraint. To optimize the validation part of $\mathcal{S}$ w.r.t. the cosine hypergradient-matching loss in Eq. (HCDC), note that we only need to take gradients of $\nabla_\theta \mathcal{L}_\mathcal{S}^{val}(\theta,\psi)$ and $\nabla_\psi \mathcal{L}_\mathcal{S}^{val}(\theta,\psi)$ w.r.t. $\mathcal{S}^{val}$. This can be handled by the same back-propagation technique in SDC, where we take gradients of $\nabla_\theta \mathcal{L}_\mathcal{S}^{train}(\theta,\psi)$ w.r.t $\mathcal{S}^{train}$.

Where to align the hypergradients in Eq. (HCDC)? The hypergradient-alignment condition, as a sufficient condition for preserving the outcome of HPO, is often too strong. For a discrete search space Ψ, we can preserve the order of any $\psi_1 \neq \psi_2 \in \Psi$, as long as there exists a continuous path connecting ψ_1 and ψ_2 on which the hypergradients $\nabla_\psi \mathcal{L}_\mathcal{T}^{val}(\theta^\mathcal{T}(\psi),\psi)$ and $\nabla_\psi \mathcal{L}_\mathcal{S}^{val}(\theta^\mathcal{S}(\psi),\psi)$ are aligned. To further avoid the $O(p^2)$ paths, we propose to align the hyperparameters on the p continuous-HPO trajectories. The i-th continuous-HPO trajectory starts from $\psi_{i,0}^\mathcal{S} = \psi_i \in \Psi$ and update through $\psi_{i,t+1}^\mathcal{S} \leftarrow \psi_{i,t}^\mathcal{S} - \eta\nabla_\psi \mathcal{L}_\mathcal{S}^{val}(\theta^\mathcal{S}(\psi_{i,t}^\mathcal{S}),\psi_{i,t}^\mathcal{S})$. All of the p trajectories will approach the optima $\psi^\mathcal{S}$ which form a "connected" path between any pair of hyperparameters $\psi_i \neq \psi_j \in \Psi$. For a continuous search space $\Psi = \Psi'$, since it is often bounded narrowly around the optima $\psi^\mathcal{S}$, we again align the hypergradients along the optimization trajectories $(\psi_{i,t}^\mathcal{S})_{t=0}^{T-1}$ despite that the starting points $\psi_i \in \Psi$ is now randomly sampled; see Section 4.

[4]It is also possible for graph condensation when the train and validation subgraphs are not connected.

What graph architecture/hyperparameter search problem can HCDC solve? We illustrate how to tackle the two types of search spaces: (1) discrete and finite Ψ and (2) continuous and bounded Ψ with two typical examples originating from the problem of searching for the best convolution matrix $C(A)$ on a large graph $\mathcal{T} = (A, X, \mathbf{y})$. **(1)** *Discrete and finite search space* Ψ: often the most important question of architecture search on large graphs is *what design of convolution filter performs best on the given graph*? One may simply train the set of p prior-defined GNNs $\{f^{C_i}_{[\theta_{C_i}, \theta_W]} \mid i = 1, \ldots, p\}$ whose convolution matrices are $\mathcal{C} = \{C_1(A; \theta_{C_1}), \ldots, C_p(A; \theta_{C_p})\}$ and compare their validation performance. We can formulate this problem as HPO, by defining an "interpolated" model $f^{\mathcal{C}}_{[\theta_C, \theta_W], \psi}$ whose convolution matrix is $C(A; \theta_C, \psi) = \varphi_1 C_1(A; \theta_{C_1}) + \cdots + \varphi_p C_p(A; \theta_{C_p})$, where hyperparameters $\psi = [\varphi_1, \ldots, \varphi_p] \in \Psi$ and $\theta_C = [\theta_{C_1}, \ldots, \theta_{C_p}]$. The feasible set $\Psi = \{\psi_1 = \mathbf{e}^p_1, \ldots, \psi_p = \mathbf{e}^p_p\}$ is the set of unit vectors in $\mathbb{R}^p$ and the extended search space can be defined as $\Psi' = [0, 1]^p$. **(2)** *Continuous and bounded search space* Ψ: one may also use a continuous generic formula, e.g., truncated series, to model a wide-range of convolution filters, i.e., $C(A; \psi) = \sum_{i=1}^{p} \varphi_i C_i(A)$, for example in ChebNet [Defferrard et al., 2016] or SGC [Wu et al., 2019] (see Appendix A.3). The only difference to the previous case is that the search space $\Psi = \Psi'$ can be larger than $[0, 1]^p$.

6 Related Work

Graph condensation [Jin et al., 2021] achieves the state-of-the-art performance for preserving GNNs' performance on the simplified graph. However, Jin et al. [2021] only adapt the gradient-matching algorithm of dataset distillation Zhao et al. [2020] to graph data, together with a MLP-based generative model for edges [Anand and Huang, 2018, Simonovsky and Komodakis, 2018], leaving out several major issues on efficiency, performance, and generalizability (discussed in Section 1). Subsequent work aims to apply the more efficient distribution-matching algorithm of dataset distillation [Zhao and Bilen, 2021a, Wang et al., 2022] to graph or speed up gradient-matching graph condensation by reducing the number of gradient-matching-steps [Jin et al., 2022]. While the efficiency issue of graph condensation is mitigated [Jin et al., 2022], the performance degradation on medium- and large-sized graphs still renders graph condensation practically meaningless. Our hyperparameter-calibrated graph distillation is specifically designed for repeated training in architecture search, which is, in contrast, well-motivated.

Implicit differentiation methods apply the implicit function theorem (IFT) to the nested-optimization problems [Ochs et al., 2015, Wang et al., 2019]. The IFT requires inverting the training Hessian with respect to the network weights. Lorraine et al. [2020] approximates the inverse Hessian by the Neumann series, which is a stable alternative to conjugate gradients [Shaban et al., 2019] and successfully scales gradient-based bilevel-optimization to large networks with constant memory constraint. It is shown that unrolling differentiation around locally optimal weights for i steps is equivalent to approximating the Neumann series inverse approximation up to the first i terms.

Differentiable NAS methods, e.g., DARTS [Liu et al., 2018] explore the possibility of transforming the discrete neural architecture space into a continuously differentiable form and further uses gradient optimization to search the neural architecture. DARTS follows a cell-based search space [Zoph et al., 2018] and continuously relaxes the original discrete search strategy. Differentiable NAS techniques have also been applied to graphs to automatically design data-specific GNN architectures [Wang et al., 2021, Huan et al., 2021].

In addition, we summarize graph reduction methods (including graph coreset selection, graph sampling, graph sparsification, and graph coarsening), as well as the more dataset condensation and coreset selection methods beyond graphs in Appendix B.

7 Experiments

In this section we validate the effectiveness of hyperparameter-calibrated dataset condensation (HCDC) when applied to speed up graph architecture/hyperparameter search. In this section, correlation refers to the Spearman's rank correlation coefficient r_s between two rankings of the ordered list of hyperparameters on the original and condensed datasets. Please refer to **??** for more implementation details.

Hyperpara. Method	Cross Validation Correlation	Performance
Random	-0.04	84.0
DC	0.68	82.6
DM	0.76	82.8
Early-Stopping	0.11	84.3
HCDC	0.91	84.7

Table 2: The rank correlation and validation performance on the real dataset of the M-fold cross validation ranked/selected on the condensed dataset.

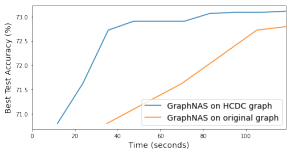


Figure 4: Speed-up to the search process of graph NAS when combined with HCDC on Ogbn-arxiv, best test performance so far vs. time spent.

Synthetic experiments on CIFAR-10. We first consider a synthetically created set of hyperparameters on image dataset, CIFAR-10. Consider the M-fold cross validation, where a fraction of $1/M$ samples are use as the validation split each time. The M-fold cross-validation process can be modeled by a set of M hyperparameters $\{\varphi_i \in \{0,1\} \mid i = 1,\ldots,M\}$, where $\varphi_i = 1$ if and only if the i-th fold is used for validation. The problem of finding the best validation performance among the M results can be modeled as a hyperparameter optimization problem with a discrete search space $|\Psi| = M$. We compare HCDC with the gradient-matching [Zhao et al., 2020] and distribution matching [Zhao and Bilen, 2021a] baselines. We also consider a uniform random sampling baseline, and an early-stopping baseline where we train only $c/n * 500$ epochs but on the original dataset. The results of $M = 20$ and $c/n = 1\%$ is reported in Table 2, where we see HCDC achieves the highest rank correlation.

Finding best convolution filter on (large) graphs. One easy application of HCDC we analyzed in Section 5 is to speed up the selection of the best suited convolution filter design on (large) graphs. Following the method discussed in Section 5, we test HCDC against (1) Random: random uniform sampling of nodes and find their induced subgraph, (2) GCond-X: graph condensation [Jin et al., 2021] but fix the synthetic adjacency to identity, (3) GCond: the graph condensation algorithm in [Jin et al., 2021], and (4) Whole Graph: when the model selection is performed on the original dataset. We use random uniform sampling to find the training synthetic subgraph before we apply HCDC. For the other coreset/condensation methods which does not define the validation split, we random split the train and validation nodes according to the original split ratio. We not only report the Spearman's rank correlation, but also the test performance (on real dataset) of model selected by the condensed dataset. The results are summarized in Table 3.

Speeding up off-the-shelf graph architecture search algorithms. Finally we test HCDC on how much speed-up it can provides to the off-the-shelf graph architecture search methods. We use graph NAS [Gao et al., 2019] on Ogbn-arxiv with a condensation ratio of $c/n = 0.5\%$. The search space of architectures is the same as the set used in Table 3 with a focus on graphs with different convolution filters. We plot the best test performance of searched architecture (so far) versus the time spent for searching (in seconds) in Fig. 4. We see HCDC, as a dataset condensation approach, can further speed up the search process of graph NAS and is orthogonal to the efficient search algorithms like Bayesian optimization or reinforcement learning used by NAS methods.

Dataset	Ratio	Random		GCond-X		GCond		HCDC		Whole Graph
		Corr.	Perf.	Corr.	Perf.	Corr.	Perf.	Corr.	Perf.	Perf. (%)
Cora	0.9%	0.29	81.2	0.14	79.5	0.61	81.9	0.80	83.2	
	1.8%	0.40	81.9	0.21	80.3	0.76	83.2	0.89	83.8	83.8
	3.6%	0.51	82.2	0.22	80.9	0.81	83.2	0.92	83.8	
Citeseer	1.3%	0.38	71.9	0.15	70.8	0.68	71.3	0.90	73.1	
	2.6%	0.56	72.2	0.29	70.8	0.79	71.5	0.93	73.7	73.7
	5.2%	0.71	73.0	0.35	70.2	0.83	71.1	0.97	73.7	
Ogbn-arxiv	0.25%	0.59	70.1	0.39	69.8	0.59	70.3	0.77	71.9	
	0.5%	0.63	70.3	0.44	70.1	0.64	70.5	0.85	72.2	73.4
	1.0%	0.68	70.9	0.47	70.0	0.67	70.1	0.88	72.2	
Reddit	0.1%	0.42	92.1	0.39	90.9	0.53	90.9	0.88	92.1	
	0.2%	0.50	93.1	0.41	90.9	0.61	91.2	0.92	92.7	94.3
	0.4%	0.58	93.1	0.42	91.5	0.66	92.1	0.96	92.7	

Table 3: Spearman's rank correlation and test performance of the convolution filter selected on the condensed graph.

References

William L Hamilton. *Graph Representation Learning*. Morgan & Claypool Publishers, 2020.

Weihua Hu, Matthias Fey, Marinka Zitnik, Yuxiao Dong, Hongyu Ren, Bowen Liu, Michele Catasta, and Jure Leskovec. Open graph benchmark: Datasets for machine learning on graphs. *Advances in neural information processing systems*, 33:22118–22133, 2020.

Thomas Elsken, Jan Hendrik Metzen, and Frank Hutter. Neural architecture search: A survey. *The Journal of Machine Learning Research*, 20(1):1997–2017, 2019.

Daniel Baker, Vladimir Braverman, Lingxiao Huang, Shaofeng H-C Jiang, Robert Krauthgamer, and Xuan Wu. Coresets for clustering in graphs of bounded treewidth. In *International Conference on Machine Learning*, pages 569–579. PMLR, 2020.

Joshua Batson, Daniel A Spielman, Nikhil Srivastava, and Shang-Hua Teng. Spectral sparsification of graphs: theory and algorithms. *Communications of the ACM*, 56(8):87–94, 2013.

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

Hanqing Zeng, Hongkuan Zhou, Ajitesh Srivastava, Rajgopal Kannan, and Viktor Prasanna. Graphsaint: Graph sampling based inductive learning method. In *International Conference on Learning Representations*, 2019.

Bo Zhao, Konda Reddy Mopuri, and Hakan Bilen. Dataset condensation with gradient matching. In *International Conference on Learning Representations*, 2020.

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.

Jonathan Lorraine, Paul Vicol, and David Duvenaud. Optimizing millions of hyperparameters by implicit differentiation. In *International Conference on Artificial Intelligence and Statistics*, pages 1540–1552. PMLR, 2020.

Hanxiao Liu, Karen Simonyan, and Yiming Yang. Darts: Differentiable architecture search. In *International Conference on Learning Representations*, 2018.

Tongzhou Wang, Jun-Yan Zhu, Antonio Torralba, and Alexei A Efros. Dataset distillation. *arXiv preprint arXiv:1811.10959*, 2018.

Muhammet Balcilar, Renton Guillaume, Pierre Héroux, Benoit Gaüzère, Sébastien Adam, and Paul Honeine. Analyzing the expressive power of graph neural networks in a spectral perspective. In *Proceedings of the International Conference on Learning Representations (ICLR)*, 2021.

Mucong Ding, Kezhi Kong, Jingling Li, Chen Zhu, John Dickerson, Furong Huang, and Tom Goldstein. Vq-gnn: A universal framework to scale up graph neural networks using vector quantization. *Advances in Neural Information Processing Systems*, 34:6733–6746, 2021.

Thomas N Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. In *International Conference on Learning Representations*, 2016.

Keyulu Xu, Weihua Hu, Jure Leskovec, and Stefanie Jegelka. How powerful are graph neural networks? In *International Conference on Learning Representations*, 2018.

Michaël Defferrard, Xavier Bresson, and Pierre Vandergheynst. Convolutional neural networks on graphs with fast localized spectral filtering. In *Advances in neural information processing systems*, volume 29, 2016.

Felix Wu, Amauri Souza, Tianyi Zhang, Christopher Fifty, Tao Yu, and Kilian Weinberger. Simplifying graph convolutional networks. In *International conference on machine learning*, pages 6861–6871. PMLR, 2019.

Namrata Anand and Possu Huang. Generative modeling for protein structures. *Advances in neural information processing systems*, 31, 2018.

Martin Simonovsky and Nikos Komodakis. Graphvae: Towards generation of small graphs using variational autoencoders. In *International conference on artificial neural networks*, pages 412–422. Springer, 2018.

Bo Zhao and Hakan Bilen. Dataset condensation with distribution matching. *arXiv preprint arXiv:2110.04181*, 2021a.

Kai Wang, Bo Zhao, Xiangyu Peng, Zheng Zhu, Shuo Yang, Shuo Wang, Guan Huang, Hakan Bilen, Xinchao Wang, and Yang You. Cafe: Learning to condense dataset by aligning features. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 12196–12205, 2022.

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*, pages 720–730, 2022.

Peter Ochs, René Ranftl, Thomas Brox, and Thomas Pock. Bilevel optimization with nonsmooth lower level problems. In *International Conference on Scale Space and Variational Methods in Computer Vision*, pages 654–665. Springer, 2015.

Yuanhao Wang, Guodong Zhang, and Jimmy Ba. On solving minimax optimization locally: A follow-the-ridge approach. In *International Conference on Learning Representations*, 2019.

Amirreza Shaban, Ching-An Cheng, Nathan Hatch, and Byron Boots. Truncated back-propagation for bilevel optimization. In *The 22nd International Conference on Artificial Intelligence and Statistics*, pages 1723–1732. PMLR, 2019.

Barret Zoph, Vijay Vasudevan, Jonathon Shlens, and Quoc V Le. Learning transferable architectures for scalable image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 8697–8710, 2018.

Zhili Wang, Shimin Di, and Lei Chen. Autogel: An automated graph neural network with explicit link information. *Advances in Neural Information Processing Systems*, 34:24509–24522, 2021.

ZHAO Huan, YAO Quanming, and TU Weiwei. Search to aggregate neighborhood for graph neural network. In *2021 IEEE 37th International Conference on Data Engineering (ICDE)*, pages 552–563. IEEE, 2021.

Yang Gao, Hong Yang, Peng Zhang, Chuan Zhou, and Yue Hu. Graphnas: Graph neural architecture search with reinforcement learning. *arXiv preprint arXiv:1904.09981*, 2019.

Will Hamilton, Zhitao Ying, and Jure Leskovec. Inductive representation learning on large graphs. *Advances in neural information processing systems*, 30, 2017.

Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Lio, and Yoshua Bengio. Graph attention networks. In *International Conference on Learning Representations*, 2018.

Johannes Klicpera, Stefan Weißenberger, and Stephan Günnemann. Diffusion improves graph learning. In *Advances in neural information processing systems*. PMLR, 2019.

Yu Rong, Yatao Bian, Tingyang Xu, Weiyang Xie, Ying Wei, Wenbing Huang, and Junzhou Huang. Self-supervised graph transformer on large-scale molecular data. In *Advances in neural information processing systems*, volume 33, 2020.

Omri Puny, Heli Ben-Hamu, and Yaron Lipman. From graph low-rank global attention to 2-fwl approximation. In *International Conference on Machine Learning*. PMLR, 2020.

Jiawei Zhang, Haopeng Zhang, Congying Xia, and Li Sun. Graph-bert: Only attention is needed for learning graph representations. *arXiv preprint arXiv:2001.05140*, 2020.

Ondrej Bohdal, Yongxin Yang, and Timothy Hospedales. Flexible dataset distillation: Learn labels instead of images. *arXiv preprint arXiv:2006.08572*, 2020.

Timothy Nguyen, Zhourong Chen, and Jaehoon Lee. Dataset meta-learning from kernel ridge-regression. In *International Conference on Learning Representations*, 2020.

Timothy Nguyen, Roman Novak, Lechao Xiao, and Jaehoon Lee. Dataset distillation with infinitely wide convolutional networks. *Advances in Neural Information Processing Systems*, 34:5186–5198, 2021.

Felipe Petroski Such, Aditya Rawal, Joel Lehman, Kenneth Stanley, and Jeffrey Clune. Generative teaching networks: Accelerating neural architecture search by learning to generate synthetic training data. In *International Conference on Machine Learning*, pages 9206–9216. PMLR, 2020.

Bo Zhao and Hakan Bilen. Dataset condensation with differentiable siamese augmentation. In *International Conference on Machine Learning*, pages 12674–12685. PMLR, 2021b.

Jang-Hyun Kim, Jinuk Kim, Seong Joon Oh, Sangdoo Yun, Hwanjun Song, Joonhyun Jeong, Jung-Woo Ha, and Hyun Oh Song. Dataset condensation via efficient synthetic-data parameterization. In *International Conference on Machine Learning*, pages 11102–11118. PMLR, 2022.

Yutian Chen, Max Welling, and Alex Smola. Super-samples from kernel herding. In *Proceedings of the Twenty-Sixth Conference on Uncertainty in Artificial Intelligence*, pages 109–116, 2010.

Sylvestre-Alvise Rebuffi, Alexander Kolesnikov, Georg Sperl, and Christoph H Lampert. icarl: Incremental classifier and representation learning. In *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*, pages 2001–2010, 2017.

Rahaf Aljundi, Min Lin, Baptiste Goujaud, and Yoshua Bengio. Gradient based sample selection for online continual learning. *Advances in neural information processing systems*, 32, 2019.

Ozan Sener and Silvio Savarese. Active learning for convolutional neural networks: A core-set approach. In *International Conference on Learning Representations*, 2018.

Zalán Borsos, Mojmir Mutny, and Andreas Krause. Coresets via bilevel optimization for continual learning and streaming. *Advances in Neural Information Processing Systems*, 33:14879–14890, 2020.

Mariya Toneva, Alessandro Sordoni, Remi Tachet des Combes, Adam Trischler, Yoshua Bengio, and Geoffrey J Gordon. An empirical study of example forgetting during deep neural network learning. In *International Conference on Learning Representations*, 2018.

Mansheej Paul, Surya Ganguli, and Gintare Karolina Dziugaite. Deep learning on a data diet: Finding important examples early in training. *Advances in Neural Information Processing Systems*, 34: 20596–20607, 2021.

Suraj Kothawade, Vishal Kaushal, Ganesh Ramakrishnan, Jeff Bilmes, and Rishabh Iyer. Prism: A rich class of parameterized submodular information measures for guided data subset selection. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 36, pages 10238–10246, 2022.

Rishabh Iyer, Ninad Khargoankar, Jeff Bilmes, and Himanshu Asanani. Submodular combinatorial information measures with applications in machine learning. In *Algorithmic Learning Theory*, pages 722–754. PMLR, 2021.

Chengcheng Guo, Bo Zhao, and Yanbing Bai. Deepcore: A comprehensive library for coreset selection in deep learning. *arXiv preprint arXiv:2204.08499*, 2022.

Vladimir Braverman, Shaofeng H-C Jiang, Robert Krauthgamer, and Xuan Wu. Coresets for clustering in excluded-minor graphs and beyond. In *Proceedings of the 2021 ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 2679–2696. SIAM, 2021.

Wei-Lin Chiang, Xuanqing Liu, Si Si, Yang Li, Samy Bengio, and Cho-Jui Hsieh. Cluster-gcn: An efficient algorithm for training deep and large graph convolutional networks. In *Proceedings of the 25th ACM SIGKDD international conference on knowledge discovery & data mining*, pages 257–266, 2019.

Venu Satuluri, Srinivasan Parthasarathy, and Yiye Ruan. Local graph sparsification for scalable clustering. In *Proceedings of the 2011 ACM SIGMOD International Conference on Management of data*, pages 721–732, 2011.

Andreas Loukas and Pierre Vandergheynst. Spectrally approximating large graphs with smaller graphs. In *International Conference on Machine Learning*, pages 3237–3246. PMLR, 2018.

Zengfeng Huang, Shengzhong Zhang, Chong Xi, Tang Liu, and Min Zhou. Scaling up graph neural networks via graph coarsening. In *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining*, pages 675–684, 2021.

Chen Cai, Dingkang Wang, and Yusu Wang. Graph coarsening with neural networks. In *International Conference on Learning Representations*, 2020.

Yoshua Bengio. Gradient-based optimization of hyperparameters. *Neural computation*, 12(8): 1889–1900, 2000.

Jan Larsen, Lars Kai Hansen, Claus Svarer, and M Ohlsson. Design and regularization of neural networks: the optimal use of a validation set. In *Neural Networks for Signal Processing VI. Proceedings of the 1996 IEEE Signal Processing Society Workshop*, pages 62–71. IEEE, 1996.

Jelena Luketina, Mathias Berglund, Klaus Greff, and Tapani Raiko. Scalable gradient-based tuning of continuous regularization hyperparameters. In *International conference on machine learning*, pages 2952–2960. PMLR, 2016.

Fabian Pedregosa. Hyperparameter optimization with approximate gradient. In *International conference on machine learning*, pages 737–746. PMLR, 2016.

James Martens and Roger Grosse. Optimizing neural networks with kronecker-factored approximate curvature. In *International conference on machine learning*, pages 2408–2417. PMLR, 2015.

Liam Li and Ameet Talwalkar. Random search and reproducibility for neural architecture search. In *Uncertainty in artificial intelligence*, pages 367–377. PMLR, 2020.

Arber Zela, Thomas Elsken, Tonmoy Saikia, Yassine Marrakchi, Thomas Brox, and Frank Hutter. Understanding and robustifying differentiable architecture search. In *International Conference on Learning Representations*, 2019.

Sirui Xie, Hehui Zheng, Chunxiao Liu, and Liang Lin. Snas: stochastic neural architecture search. In *International Conference on Learning Representations*, 2018.

Eric Jang, Shixiang Gu, and Ben Poole. Categorical reparameterization with gumbel-softmax. In *International Conference on Learning Representations*, 2017.

Chris J Maddison, Andriy Mnih, and Yee Whye Teh. The concrete distribution: A continuous relaxation of discrete random variables. In *International Conference on Learning Representations*, 2017.

Supplementary Material

A Proofs and Extended Theoretical Results

Recall we consider the linear regression problem with linear convolution model $f_{\theta=[\theta_C,\theta_W]}(A,X) = C(A,\theta_C)XW(\theta_W)$, where $C(A;\theta_C)$ is the *convolution matrix* which has the same sizes as the adjacency matrix A and possibly also depends on the parameters $\theta_C \in \mathbb{R}^p$, and $W(\theta_W)$ is the learnable linear weight matrix which is a reshape of the parameters $\theta_W \in \mathbb{R}^d$. The loss is now sum of squares $\mathcal{L}(\theta,\psi) = \|\mathbf{y} - CXW\|_2^2$ (train-split subscript omitted) where the labels $\mathbf{y}$ are continuous.

Recall that we say the gradient-matching objective is satisfied for a non-degenerate trajectory, if there exists a fixed learning trajectory $(\theta_t^{\mathcal{S}})_{t=0}^{T-1}$ which span the whole parameter space, i.e., $\text{span}(\theta_0^{\mathcal{S}},\ldots,\theta_{T-1}^{\mathcal{S}}) = \mathbb{R}^{p+d}$, such that the *gradient-matching* loss on this trajectory (objective of Eq. (SDC-GM) without expectation) is 0.

A.1 Validity of Standard Dataset Condensation

Proposition 4. *(Validity of SDC) Assuming least square regression with linear convolution model, if gradient-matching objective is satisfied for a non-degenerate trajectory, then the optimizer on the condensed dataset $\mathcal{S}$, i.e., $\theta^{\mathcal{S}} = \arg\min_\theta \mathcal{L}_{\mathcal{S}}(\theta,\psi)$ is also optimal for the original dataset, i.e., $\mathcal{L}_{\mathcal{T}}(\theta^{\mathcal{S}},\psi) = \min_\theta \mathcal{L}_{\mathcal{T}}(\theta,\psi)$.*

Proof. If the gradient-matching objective is satisfied for a non-degenerate trajectory, one can derive $(X^\top C^\top CX) = (X'^\top C'^\top C'X')$ and $X^\top C^\top \mathbf{y} = X'^\top C'^\top \mathbf{y}'$. These directly least to that the optimizer $\theta_W^{\mathcal{S}} = \theta_W^{\mathcal{T}}$ and thus $\mathcal{L}_{\mathcal{T}}(\theta^{\mathcal{S}},\psi) = \min_\theta \mathcal{L}_{\mathcal{T}}(\theta,\psi)$. □

A.2 Generalization Issue of Standard Dataset Condensation

Proof of Proposition 1: If the gradient-matching objective is satisfied for a non-degenerate trajectory, similarly, one can derive $(X^\top P^{k\top} P^k X) = (X'^\top P^{k\top} P^k X')$ and $X^\top P^{k\top}\mathbf{y} = X'^\top P^{k\top}\mathbf{y}'$ for any $k = -K,\ldots,0,\ldots,K$. Thus for a 1D-CNN with smaller $K' \le K$, the equations above readily lead to the gradient-matching objective for the new model. □

Proof of Proposition 2: This proposition easily follow form the fact that the gradient-matching objective is minimized for any trajectory if the optimized parameter matches $\theta^{\mathcal{S}} = (X'^\top C'^\top C'X')X'^\top C'^\top \mathbf{y}' = (X^\top C^\top CX)X^\top C^\top \mathbf{y}$. From which we see we only require the product $C'X'$ to match CX. □

Proof of Proposition 3: The lower bound follows from the optimized parameter $\theta_W^{\mathcal{S}}$ and $\theta_W^{\mathcal{T}}$ with the new GNN with convolution filter $\mathfrak{C}$ and the inequality $\|AB\|_F \le \sigma_{max}(A)\|B\|_F$. □

Although the results above are obtained for least squares loss and linear convolution model, it still reflects the nature of general non-convex losses and non-linear models. Since dataset condensation is effectively matching the local minima $\{\theta^{\mathcal{T}}\}$ of the original loss $\mathcal{L}_{\mathcal{T}}^{train}(\theta,\psi)$ with the local minima $\{\theta^{\mathcal{S}}\}$ of the condensed loss $\mathcal{L}_{\mathcal{S}}^{train}(\theta,\psi)$, within the small neighborhoods surrounding the pair of local minima $(\theta^{\mathcal{T}},\theta^{\mathcal{S}})$, we can approximate the non-convex loss and non-linear model with a convex/linear one respectively. And hence the generalizability issues with convex loss and liner model may hold.

A.3 Convolution Filters of GNNs

The convolution formulation of many popular GNNs [Balcilar et al., 2021] is summarized in Table 4.

B Extended Related Work

B.1 Dataset Condensation and Coreset Selection

Firstly, we review the two main approaches to reducing the training set size while preserving model performance.

Model Name	Design Idea	Conv. Matrix Type	# of Conv.	Convolution Matrix
GCN[1] [Kipf and Welling, 2016]	Spatial Conv.	Fixed	1	$C = \widetilde{D}^{-1/2}\widetilde{A}\widetilde{D}^{-1/2}$
SAGE-Mean[2] [Hamilton et al., 2017]	Message Passing	Fixed	2	$\begin{cases} C^{(1)} = I_n \\ C^{(2)} = D^{-1}A \end{cases}$
GAT[3] [Veličković et al., 2018]	Self-Attention	Learnable	# of heads	$\begin{cases} \mathfrak{C}^{(s)} = A + I_n \text{ and} \\ h^{(s)}_{\boldsymbol{a}^{(l,s)}}(X^{(l)}_{i,:}, X^{(l)}_{j,:}) = \exp\big(\text{LeakyReLU}(\\ \quad (X^{(l)}_{i,:}W^{(l,s)} \parallel X^{(l)}_{j,:}W^{(l,s)}) \cdot \boldsymbol{a}^{(l,s)})\big) \end{cases}$
GIN[1] [Xu et al., 2018]	WL-Test	Fixed + Learnable	2	$\begin{cases} C^{(1)} = A \\ \mathfrak{C}^{(2)} = I_n \text{ and } h^{(2)}_{\epsilon^{(l)}} = 1 + \epsilon^{(l)} \end{cases}$
SGC!![2] [Defferrard et al., 2016]	Spectral Conv.	Learnable	order of poly.	$\begin{cases} \mathfrak{C}^{(1)} = I_n, \mathfrak{C}^{(2)} = 2L/\lambda_{\max} - I_n, \\ \mathfrak{C}^{(s)} = 2\mathfrak{C}^{(2)}\mathfrak{C}^{(s-1)} - \mathfrak{C}^{(s-2)} \\ \text{and } h^{(s)}_{\theta^{(s)}} = \theta^{(s)} \end{cases}$
ChebNet[2] [Defferrard et al., 2016]	Spectral Conv.	Learnable	order of poly.	$\begin{cases} \mathfrak{C}^{(1)} = I_n, \mathfrak{C}^{(2)} = 2L/\lambda_{\max} - I_n, \\ \mathfrak{C}^{(s)} = 2\mathfrak{C}^{(2)}\mathfrak{C}^{(s-1)} - \mathfrak{C}^{(s-2)} \\ \text{and } h^{(s)}_{\theta^{(s)}} = \theta^{(s)} \end{cases}$
GDC[3] [Klicpera et al., 2019]	Diffusion	Fixed	1	$C = S$
Graph Transformers[4] [Rong et al., 2020]	Self-Attention	Learnable	# of heads	$\begin{cases} \mathfrak{C}^{(s)}_{i,j} = 1 \text{ and } h^{(s)}_{(W^{(l,s)}_Q, W^{(l,s)}_K)}(X^{(l)}_{i,:}, X^{(l)}_{j,:}) \\ = \exp\big(\frac{1}{\sqrt{d_{k,l}}}(X^{(l)}_{i,:}W^{(l,s)}_Q)(X^{(l)}_{j,:}W^{(l,s)}_K)^\intercal\big) \end{cases}$

[1] Where $\widetilde{A} = A + I_n$, $\widetilde{D} = D + I_n$. [2] $C^{(2)}$ represents mean aggregator. Weight matrix in [Hamilton et al., 2017] is $W^{(l)} = W^{(l,1)} \parallel W^{(l,2)}$. [3] Need row-wise normalization. $C^{(l,s)}_{i,j}$ is non-zero if and only if $A_{i,j} = 1$, thus GAT follows direct-neighbor aggregation. [4] The weight matrices of the two convolution supports are the same, $W^{(l,1)} = W^{(l,2)}$. [5] Where normalized Laplacian $L = I_n - D^{-1/2}AD^{-1/2}$ and $\lambda_{\max}$ is its largest eigenvalue, which can be approximated as 2 for a large graph. [6] Where S is the diffusion matrix $S = \sum_{k=0}^{\infty}\theta_k\boldsymbol{T}^k$, for example, decaying weights $\theta_k = e^{-t}\frac{t^k}{k!}$ and transition matrix $\boldsymbol{T} = \widetilde{D}^{-1/2}\widetilde{A}\widetilde{D}^{-1/2}$. [7] Need row-wise normalization. Only describes the global self-attention layer, where $W^{(l,s)}_Q, W^{(l,s)}_Q \in \mathbb{R}^{f_l, d_{k,l}}$ are weight matrices which compute the queries and keys vectors. In contrast to GAT, all entries of $\mathfrak{C}^{(l,s)}_{i,j}$ are non-zero. Different design of Graph Transformers [Puny et al., 2020, Rong et al., 2020, Zhang et al., 2020] use graph adjacency information in different ways, and is not characterized here, see the original papers for details.

Table 4: Summary of GNNs formulated as generalized graph convolution.

Dataset condensation (or distillation) is first proposed in [Wang et al., 2018] as a learning-to-learn problem by formulating the network parameters as a function of synthetic data and learning them through the network parameters to minimize the training loss over the original data. However, the nested-loop optimization precludes it scaling up to large-scale in-the-wild datasets. Zhao et al. [2020] alleviate this issue by enforcing the gradients of the synthetic samples w.r.t. the network weights to approach those of the original data, which successfully alleviates the expensive unrolling of the computational graph. Based on the meta-learning formulation in [Wang et al., 2018], Bohdal et al. [2020] and Nguyen et al. [2020, 2021] propose to simplify the inner-loop optimization of a classification model by training with ridge regression which has a closed-form solution, while Such et al. [2020] model the synthetic data using a generative network. To improve the data efficiency of synthetic samples in gradient-matching algorithm, Zhao and Bilen [2021b] apply differentiable Siamese augmentation, and Kim et al. [2022] introduce efficient synthetic-data parametrization. Recently, a new distribution-matching framework [Zhao and Bilen, 2021a] proposes to match the hidden features rather than the gradients for fast optimization, but may suffer from performance degradation compared to gradient-matching [Zhao and Bilen, 2021a], where Kim et al. [2022] provide some interpretation.

Coreset selection methods choose samples that are important for training based on heuristic criteria, for example, minimizing the distance between coreset and whole-dataset centers [Chen et al., 2010, Rebuffi et al., 2017], maximizing the diversity of selected samples in the gradient space [Aljundi et al., 2019], discovering cluster centers [Sener and Savarese, 2018], and choosing samples with the largest negative implicit gradient [Borsos et al., 2020]. *Forgetting* [Toneva et al., 2018] measures the forgetfulness of trained samples and drops those that are not easy to forget. *GraNd* [Paul et al., 2021] selects the training samples that contribute most to the training loss in the first few epochs. *Prism* [Kothawade et al., 2022] select samples to maximize submodular set-functions which are combinatorial generalizations of entropy measures [Iyer et al., 2021]. Recent benchmark [Guo et al., 2022] of a variety of coreset selection methods for image classification indicates *Forgetting*, *GraNd*, and *Prism* are among the best performing corset methods but still evidently underperform the dataset

condensation baselines. Although coreset selection can be very efficient, most of the methods above suffer from three major limitations: (1) their performance is upper-bounded by the information in the selected samples; (2) most of them do not directly optimize the synthetic samples to preserve the model performance; and (3) most of them select samples incrementally and greedily, which are short-sighted.

B.2 Graph Reduction

Secondly, we summarize the traditional graph reduction method for graph neural network training.

Graph coreset selection is a non-trivial generalization of the above method coreset methods given the non-*iid* nature of graph nodes and the non-linearity nature of GNNs. The very few off-the-shelf graph coreset algorithms are designed for graph clustering [Baker et al., 2020, Braverman et al., 2021] and are not optimal for the training of GNNs.

Graph sampling methods [Chiang et al., 2019, Zeng et al., 2019] can be as simple as uniformly sampling a set of nodes and finding their induced subgraph, which is understood as a graph-counterpart of uniform sampling of *iid* samples. However, most of the present graph sampling algorithms (e.g., ClusterGCN [Chiang et al., 2019] and GraphSAINT [Zeng et al., 2019]) are designed for sampling multiple subgraphs (mini-batches), which forms a cover of the original graph for training GNNs with memory constraint. Therefore those graph mini-batch sampling algorithms are effectively graph partitioning algorithms and not optimized to find just one representative subgraph.

Graph sparsification [Batson et al., 2013, Satuluri et al., 2011] and **graph coarsening** [Loukas and Vandergheynst, 2018, Loukas, 2019, Huang et al., 2021, Cai et al., 2020] algorithms are usually designed to preserve specific graph properties like graph spectrum and graph clustering. Such objectives are often not aligned with the optimization of downstream GNNs and are shown to be sub-optimal in preserving the information to train GNNs well [Jin et al., 2021].

B.3 Other Related Areas

Lastly, we list several important relevant areas.

Implicit differentiation methods apply the implicit function theorem (IFT) to the nested-optimization problems [Ochs et al., 2015, Wang et al., 2019]. The IFT requires inverting the training Hessian with respect to the network weights, where early work either computes the inverse explicitly [Bengio, 2000, Larsen et al., 1996] or approximates it as the identity matrix [Luketina et al., 2016]. Conjugate gradient (CG) is applied to invert the Hessian approximately [Pedregosa, 2016], but is difficult to scale to deep networks. Several methods have been proposed to efficiently approximate Hessian inverse, for example, 1-step unrolled differentiation [Luketina et al., 2016], Fisher information matrix [Larsen et al., 1996], NN-structure aided Kronecker-factored inversion [Martens and Grosse, 2015]. Lorraine et al. [2020] use the Neumann inverse approximation, which is a stable alternative to CG [Shaban et al., 2019] and successfully scale gradient-based bilevel-optimization to large networks with constant memory constraint. It is shown that unrolling differentiation around locally optimal weights for i steps is equivalent to approximating the Neumann series inverse approximation up to the first i terms.

Differentiable NAS methods, e.g., DARTS [Liu et al., 2018] explore the possibility of transforming the discrete neural architecture space into a continuously differentiable form and further uses gradient optimization to search the neural architecture. DARTS follows a cell-based search space [Zoph et al., 2018] and continuously relaxes the original discrete search strategy. Despite its simplicity, several work cast double on the effectiveness of DARTS [Li and Talwalkar, 2020, Zela et al., 2019]. SNAS [Xie et al., 2018] points out that DARTS suffers from the unbounded bias issue towards its objective, and it remodels the NAS and leverages the Gumbel-softmax trick [Jang et al., 2017, Maddison et al., 2017] to learn the exact architecture parameter. Differentiable NAS techniques have also been applied to graphs to automatically design data-specific GNN architectures [Wang et al., 2021, Huan et al., 2021].