SizeShiftReg: a Regularization Method for Improving Size-Generalization in Graph Neural Networks

Davide Buffelli
Department of Information Engineering
University of Padova
Padova, Italy 35131
davide.buffelli@unipd.it

Pietro Liò
Department of Computer Science and Technology
University of Cambridge
Cambridge, United Kingdom CB3 0FD
pietro.lio@cl.cam.ac.uk

Fabio Vandin
Department of Information Engineering
University of Padova
Padova, Italy 35131
fabio.vandin@unipd.it

Abstract

In the past few years, graph neural networks (GNNs) have become the *de facto* model of choice for graph classification. While, from the theoretical viewpoint, most GNNs can operate on graphs of any size, it is empirically observed that their classification performance degrades when they are applied on graphs with sizes that differ from those in the training data. Previous works have tried to tackle this issue in graph classification by providing the model with inductive biases derived from assumptions on the generative process of the graphs, or by requiring access to graphs from the test domain. The first strategy is tied to the quality of the assumptions made for the generative process, and requires the use of specific models designed after the explicit definition of the generative process of the data, leaving open the question of how to improve the performance of *generic* GNN models in general settings. On the other hand, the second strategy can be applied to any GNN, but requires access to information that is not always easy to obtain. In this work we consider the scenario in which we only have access to the training data, and we propose a regularization strategy that can be applied to any GNN to improve its generalization capabilities from smaller to larger graphs without requiring access to the test data. Our regularization is based on the idea of simulating a shift in the size of the training graphs using coarsening techniques, and enforcing the model to be robust to such a shift. Experimental results on standard datasets show that popular GNN models, trained on the 50% smallest graphs in the dataset and tested on the 10% largest graphs, obtain performance improvements of up to 30% when trained with our regularization strategy.

1 Introduction

Graph structured data are found in a large variety of domains, ranging from social networks, to molecules and transportation networks. When dealing with graph data in machine learning settings, it is common to resort to graph representation learning [29]. Graph representation learning is the task of learning to generate vector representations of the nodes, or of entire graphs, that encode structural and feature-related information that can be used for downstream tasks. In this area, Graph Neural

36th Conference on Neural Information Processing Systems (NeurIPS 2022).

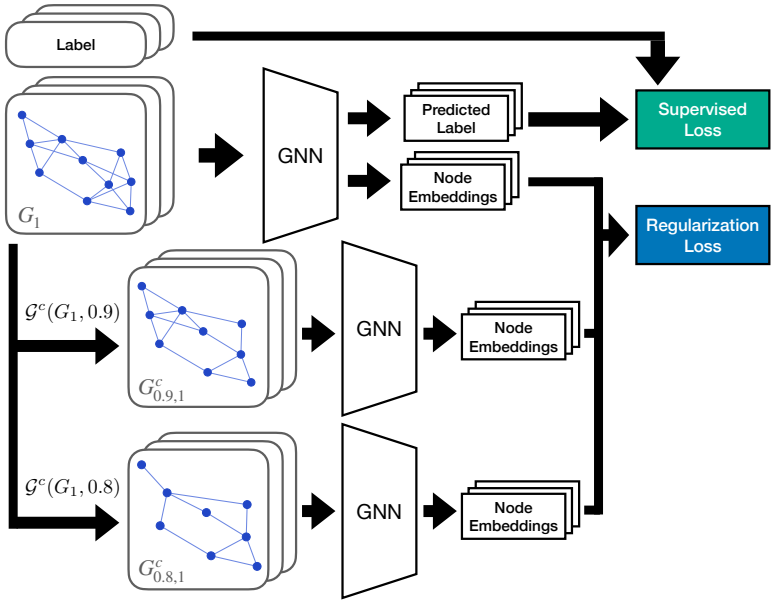


Figure 1: Overview of our method: given a training set of labelled graphs, we simulate a size shift by applying a coarsening function $\mathcal{G}^c$ and we regularize the model to be robust towards this shift. The regularization enforces the *distribution* of node embeddings generated by the model for the original graph and its coarsened versions to be similar.

Networks (GNNs) are a deep learning model for graph representation learning that has become the *de facto* standard for many tasks, including graph classification, which is the focus of this work.

GNNs are designed to be able to work on graphs of any size. However, empirical results show that they struggle at generalizing to graphs of sizes that differ from those in the training data [33, 25, 6, 68]. Obtaining good size-generalization performance from smaller to larger graphs is crucial for several reasons. **First**, it is common for graphs in the same domain to vary in size. For example, social networks can range from tens to millions of nodes, and molecular graphs can go from few atoms, to large compounds with thousands of atoms. **Second**, in many scenarios, obtaining labels for smaller graphs is cheaper than for larger graphs. For example, recently deep learning is being used in combinatorial optimization problems [5], and GNNs are heavily used in this area as many problems can be formulated as graph classification instances [53, 24, 11]. Combinatorial optimization often deals with NP-hard problems, and obtaining ground truth labels can be extremely expensive for large scale graphs. **Third**, training on smaller graphs requires less computational resources.

Two approaches have been proposed in literature to tackle the poor size-generalization of GNNs. The first approach makes assumptions on the causal graph describing the generative process behind the graphs in the dataset, and then designs a model that is tailored towards such causal graph [6]. This strategy requires an explicit definition of the generative process behind the data, which, in the case of Bevilacqua et al. [6], leads to a model requiring the computation of induced homomorphism densities over all connected k-vertex subgraphs in order to make a prediction. While this strategy shows great results on synthetic graphs produced by the assumed generative process, its benefits are reduced on real-world graphs, for which the underlying generative process is unknown. The second approach assumes access to graphs coming from different domains, or from the test distribution, and applies domain adaptation techniques to transfer knowledge across domains [68, 17]. These methods do not require ad-hoc models, but require knowledge about each graph's domain, or access to the test distribution, both of which are not always available or easily obtainable in advance.

In this work we consider the graph classification scenario in which we only have access to the training data (as in the first approach described above). However, in contrast to prior work, we do not try to improve the size-generalization capabilities (from smaller to larger graphs) of GNNs by handcrafting models on the base of specific knowledge or assumptions, but we study a general form of regularization that can be applied to any GNN, and that aims at letting the model *learn* to

be robust to size-shifts. The idea behind our regularization is to simulate a shift in the size of the training graphs, and to regularize the model to be robust towards this shift. The size-shift simulation is obtained using graph coarsening techniques, while the regularization is performed by minimizing the discrepancy between the distributions of the node embeddings generated by the GNN model for the original training graphs and their coarsened versions (an overview is shown in Figure 1).

In the experimental section we evaluate GNN models on popular benchmark datasets using a particular train/test split designed to test for size-generalizaiton. In particular, following [68, 6], we train on the 50% smallest graphs and test on the 10% largest. This split leads to an average size of the test graphs which is 3 to 9 times larger than the average size of the training graphs. Our results show that our regularization method applied to standard GNN models (GCN [36], PNA [15], GIN [65]) leads to an average performance improvement on the test set of up to 30% across datasets. Furthermore, these performance improvements show that standard GNN models trained with our regularization can achieve better size-generalization performance than more expensive models. The latter more expensive models were designed after making explicit assumptions on the generative process behind the graphs in the dataset [6], while our method relies on *learning* to be robust towards size-shifts.

Our contributions. Our contributions are threefold. (1) We propose a regularization strategy that can be applied to any GNN to improve its size-generalization capabilities on the task of graph classification. (2) We analyze the impact that our strategy has on the embeddings generated by a GNN. (3) We test our strategy on popular GNN models, and compare against previously proposed methods, showing that "standard" GNN models augmented with our regularization strategy achieve comparable or better size-generalization performance than more complex models designed after explicit assumptions on the generative process behind the data.

2 Preliminaries

Notation. Let $G = (V, E)$ be a graph, where V is the set of nodes with size n, and $E \subseteq V \times V$ is the set of edges. For a node $v \in V$, we use $\mathcal{N}_v$ to indicate the set of its one-hop neighbours (i.e., the nodes connected to v by an edge). We represent an *attributed* graph G with a tuple $(\mathbf{A}, \mathbf{X})$, where $\mathbf{A} \in \{0,1\}^{n \times n}$ is the adjacency matrix (i.e., the element in position i, j is equal to 1 if there is an edge between nodes i and j, and zero otherwise), and $\mathbf{X} \in \mathbb{R}^{n \times d}$ is the feature matrix, where the i-th row, $\mathbf{X}_i$, contains the d-dimensional feature vector for node i.

2.1 Graph Neural Networks

Most popular Graph Neural Networks (GNNs) [12, 64] models follow the *message-passing* paradigm [26]. Let $\mathbf{H}^{(\ell)} \in \mathbb{R}^{n \times d'}$ be the matrix containing the node representations at layer ℓ (i.e., the representation for node i is contained in the i-th row $\mathbf{H}_i^{(\ell)}$), with $\mathbf{H}^{(0)} = \mathbf{X}$. Given an aggregation function Φ, which is permutation invariant, and a learnable update function Ψ (usually a neural network), a message passing layer updates the representation of every node v as follows:

$$\mathbf{H}_v^{(\ell+1)} = \Psi(\mathbf{H}_v^{(\ell)}, m_v^{(\ell)}), \text{ with } m_v^{(\ell)} = \Phi(\{\mathbf{H}_u^{(\ell)} \; \forall u \in \mathcal{N}_v\}) \tag{1}$$

where m_v represents the aggregated message received by node v from its neighbours. After L message-passing layers, the final node embeddings $\mathbf{H}^{(L)}$ are used to perform a given task (e.g., they are the input to a network performing the task), and the whole system is trained end-to-end.

2.2 Graph Coarsening

Given a graph $G = (V, E)$, a coarsened version $G^c = (V^c, E^c)$ is obtained by partitioning V and grouping together the nodes that belong to the same subset of the partition. More formally, given a partition $P = \{S_1, S_2, \ldots, S_{n\prime}\}$ where $S_i \subseteq V$, $S_i \cap S_j = \emptyset$, $\bigcup_i S_i = V$, we associate a node s_i to each subset S_i and let $V^c = \{s_1, s_2, \ldots, s_{n\prime}\}$ and $E^c = \{(s_i, s_j) \mid$ there is a node in S_i connected to at least one node in $S_j\}$. The goal of graph coarsening is to generate a coarsened version of a graph (i.e., $n\prime < n$) such that specific properties are preserved. For instance, Loukas and Vandergheynst [45] aim at preserving the principal eigenvalues and eigenspaces, while Jin et al. [32] aim at minimizing a distance function that measures the changes in the structure and connectivity that occur in the coarsening process. Jin et al. [32] further prove that their algorithm bounds the spectral distance [35] between the original and a "lifted" version of the coarsened graph.

2.3 Central Moment Discrepancy

The Central Moment Discrepancy (CMD) [70] is a metric used to measure the discrepancy between the distribution of high-dimensional random variables. Let p and q be two probability distributions with support in the interval $[a, b]^d$ and let c_k be the k-th order moment, then the CMD between p and q is defined as:

$$CMD(p, q) = \frac{1}{|b-a|} \|\mathbb{E}(p) - \mathbb{E}(q)\|_2 + \sum_{k=2}^{\infty} \frac{1}{|b-a|^k} \|c_k(p) - c_k(q)\|_2 . \tag{2}$$

At a practical level, we follow Zhu et al. [72], and limit the number of considered moments to 5. To align with the definition of CMD it is possible to treat node embeddings as realizations of *bounded* high dimensional distributions by applying bounded activation functions like tanh or sigmoid at the embeddings layer (as done in [70, 72]).

3 Our Method

The high level idea behind our method is to "simulate" a size-shift in the training dataset, and to regularize the GNN to be robust to this shift (an overview is shown in Figure 1). While previous works rely on models that incorporate domain-specific inductive biases and/or are tied to assumptions on the generative process of the data, our method aims at *learning* to be robust towards size-shifts.

Simulating the shift. Let $\mathcal{G}^c : (G, r) \rightarrow G^c_r$ be a *coarsening function*, that takes as input a graph G with n nodes and a ratio $r \in (0, 1)$, and returns a coarsened version of the graph, G^c_r, which has $\lfloor r \times n \rfloor$ nodes. Our method is not bound to a specific coarsening function $\mathcal{G}^c$, any existing method can be used, e.g. [45, 43, 32]. Our method then proceeds in the following way: given a dataset of graphs $\mathcal{D} = \{G_1, G_2, \ldots, G_\ell\}$, a coarsening function $\mathcal{G}^c$, and a set of k coarsening ratios $C = \{r_1, r_2, \ldots, r_k\}$, we create a coarsened dataset $\mathcal{D}_{r_j}$ for each ratio by applying the coarsening function to each graph in the dataset: $\mathcal{D}_{r_j} = \{G^c_{r_j,1} = \mathcal{G}^c(G_1, r_j), G^c_{r_j,2} = \mathcal{G}^c(G_2, r_j), \cdots, G^c_{r_j,\ell} = \mathcal{G}^c(G_\ell, r_j)\}, \forall j = 1, 2, \ldots, k$. When the graph is attributed, we obtain the features for each node of the coarsened version of the graph by aggregating the features of the corresponding nodes in the original graph using simple aggregations (e.g., mean, max, sum), as commonly done in readout functions for GNNs [64].

Regularizing the GNN. Given the graph dataset $\mathcal{D}$, its coarsened versions $\mathcal{D}_{r_1}, \mathcal{D}_{r_2}, \cdots, \mathcal{D}_{r_k}$, and a GNN f_θ, where θ are the parameters, we aim at regularizing the GNN so that the distribution of the node embeddings generated by the model for the graphs in the original dataset and their coarsened version is similar. In other words, for a given graph, we want the model to generate a *distribution* of node embeddings that is robust across different coarsened versions of the graph. In more detail, we regularize a GNN by minimizing the CMD [70] between the distribution of the node embeddings generated by the GNN for the original graph and the distribution of the node embeddings generated by the GNN for the coarsened version(s) of the graph. We choose CMD as it has proven to be successful and stable as a regularization term for non-linear models [42, 72]. Formally, let $\mathcal{L}$ be the supervised loss function that is used to train the model (e.g. cross entropy for classification), and let λ be a term measuring the strength of the regularization. Our optimization problem is

$$\underset{\theta}{\operatorname{argmin}}\, \mathcal{L} + \lambda \mathcal{L}_{\text{size}}, \text{ where } \mathcal{L}_{\text{size}} = \sum_{j=1}^{k} \sum_{i=1}^{\ell} \text{CMD}(f_\theta(G_i), f_\theta(G^c_{r_j,i})). \tag{3}$$

Our method is model-agnostic, and can hence be applied to any GNN.

Pseudocode and practical aspect. Algorithm 1 presents the pseudocode for computing our regularization loss for a batch of graphs during training. The pseudocode is presented in an extended manner for clarity, but at a practical level, the coarsened versions of the training graphs are pre-computed before training, and the computation of the loss is done in a vectored manner so it is computed concurrently for all graphs in the batch in one pass (i.e., we do not iterate through the graphs in the batch). At a practical level, in our experiments we notice that training a model with our regularization introduces a 50% overhead in the running time for a training epoch w.r.t. training the same model without regularization.

3.1 Limitations

Similarly to previous works [6, 68] we are assuming that there are some properties that determine the label of a graph and that do not depend on the size of the graph. In a scenario in which small graphs do not carry information that is relevant for solving the task on larger graphs, we do not expect our regularization to provide substantial benefits, and the best option would be to include larger graphs in the training set. In our experiments, the ratio between the average size of the graphs in the test set and the average size of the graphs in the training set is between 3 and 9. While our method shows significant performance improvements in this setting, it may show lower performance improvements when this ratio reaches much higher values.

Algorithm 1 Computing Regularization Loss for an Input Batch during Training

Require: Coarsening ratios C, coarsening function $\mathcal{G}^c(\text{graph}, \text{ratio})$
Input: Batch B of size n_b, GNN model f_θ
  coarsened_batches $\leftarrow \{\}$
  for r in C **do** ▷ Create a new batch of coarsened graphs for each ratio
    Batch $B_r \leftarrow []$
    for G in B **do**
      $G_r^c \leftarrow \mathcal{G}^c(G, r)$
      B_r.add(G_r^c)
    end for
    coarsened_batches $\leftarrow$ coarsened_batches $\cup B_r$
  end for
  $\ell \leftarrow 0$ ▷ Initialize loss
  for B_r in coarsened_batches **do**
    for i in $\{1, 2, \ldots, n_b\}$ **do**
      embs_og $= f_\theta(B[i])$ ▷ Compute node embeddings for original graph
      embs_coarse $= f_\theta(B_r[i])$ ▷ Compute node embeddings for coarsened version of graph
      $\ell \leftarrow \ell +$ CMD(embs_og, embs_coarse) ▷ Compute CMD between node embeddings
    end for
  end for
  Return ℓ

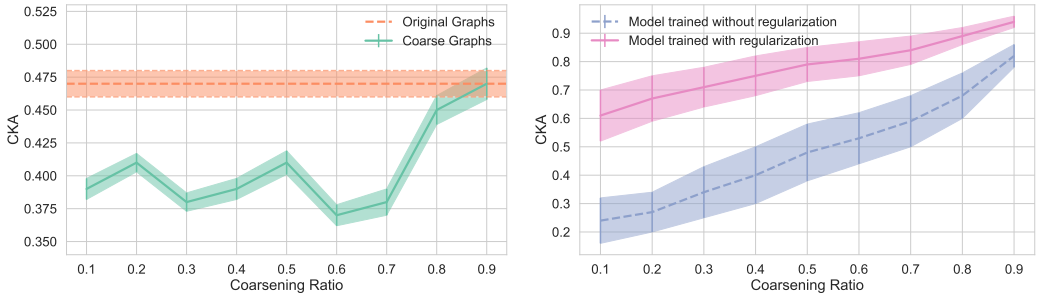
4 Analysis of Node Embeddings

Before evaluating how our regularization impacts the size-generalization performance of a model, we analyze the effects that our regularization has on the embeddings generated by the model. In order to do this, we consider two identical GIN [65] models and we train one with our regularization and one without. We then use these models to generate node embeddings and use the Central Kernel Alignment (CKA) [38] to study the generated representations, similarly to [34]. CKA takes as input two matrices $A \in \mathbb{R}^{m \times d'}, B \in \mathbb{R}^{m \times d''}$ of representations and provides a value between 0 and 1 quantifying how aligned the representations are (allowing for $d' \neq d''$). CKA quantifies the similarity of representations learned by (possibly) different models and gives us a way to study the effects of our regularization. We report results averaged over 10 different random seeds. We show only results for the DD dataset for space limitations, but the same trends are observed for the other datasets (see Appendix).

Table 1: Average CKA values between the node embeddings generated by two models, one trained with and one without our regularization, across the graphs in a dataset and their coarsened versions.

Dataset	NCI1	NCI109	PROTEINS	DD
Original	0.43 ± 0.06	0.58 ± 0.10	0.45 ± 0.06	0.47 ± 0.01
Coarsened	0.12 ± 0.06	0.38 ± 0.13	0.34 ± 0.08	0.40 ± 0.01

First, we ask if a model trained with our regularization and a model trained without our regularization produce similar node embeddings. To compare the node embeddings *between* the two models we obtain a representation for each graph by concatenating the node embeddings for that graph. We then compute the CKA between the representations generated by the model trained with regularization and the representations from model trained without regularization. We do this for the original graphs and the coarsened versions of the graphs, and report the average CKA. Results shown in Table 1 highlight that there is low alignment between the embeddings generated by a model trained with regularization and one trained without regularization (the average CKA across datasets is 0.48 for the original graphs). This is even more apparent for the coarsened graphs, showing that our regularization



(a) (b)

Figure 2: (a) Average CKA values between the node embeddings from two models, one trained with our regularization and one without, across the original graphs in the DD dataset and their coarsened versions for different ratios. The models produce representations which are not aligned (values are far from 1.0), specially for coarsened graphs. (b) Average CKA values between the node embeddings generated by the same model for the original graphs in DD and their coarsened versions. A model trained with our regularization produces representations of the coarsened graphs that are more aligned to the representations produced for the original graphs (i.e. it learns to be robust to size shifts).

is impacting the way a model generates embeddings, especially for size-shifted versions of the graphs. To analyze the trend more in detail, we also plot the CKA values across coarsening ratios in Figure 2 (a). Here we notice that the CKA between the embeddings generated by a model trained with our regularization and a model trained without, decreases significantly with ratios lower than 0.8. This shows that, while the alignment between the representations learned by the models trained with and without our regularization is low even for a coarsening ratio of 0.9 (average CKA value is 0.47), it becomes 15% lower (on average) for ratios smaller than 0.7, indicating that when the size shift is strong, there is a stronger misalignment between models trained with and without regularization.

Second, we ask if a model trained with our regularization is in fact being more robust to size shifts with respect to a model trained without our regularization. To answer this question we analyze the alignment between the representations generated by the same model, trained with or without regularization, on original graphs and coarsened versions of the graphs. In Figure 2 (b) we plot the average CKA between the node embeddings for the original graphs and the node embeddings for the coarsened versions of the graphs, both generated by the same model, for different ratios. The plot shows that a model trained with our regularization is learning to be more robust to size shifts, as its representation for coarsened versions of the graphs are much more aligned to the representations for the original graphs compared to a model trained without regularization. This is even more apparent for low coarsening ratios (i.e., when the shift is stronger), indicating that our regularization is in fact making the model more robust to size shifts.

5 Evaluation

Datasets. Following prior works [6, 68], we evaluate on datasets from the TUDataset library [48] with a train/test split explicitly designed to test for size-generalization. In particular, the training split contains the graphs with size smaller or equal than the 50-th percentile, and the test split contains graphs with a size larger or equal than the 90-th percentile. With this split, the average size of the test graphs is 3 to 9 times larger than the average size of the training graphs (in more detail, it is 3 for NCI1 and NCI109, 9 for PROTEINS, and 5 for DD). We use 10% of the training graphs as a validation set. More details on the datasets can be found in the Appendix.

Models. As in Bevilacqua et al. [6], we consider three standard GNN models: GCN [36], GIN [65], and PNA [15], a more expressive GNN: RPGNN [49], and two graph kernels: the Graphlet Counting kernel (GC Kernel) [58], and the WL Kernel [59]. We further apply the Invariant Risk Minimization (IRM) [1] penalty to the standard GNN models, and compare against the E-Invariant models ($\Gamma_{\text{1-hot}}, \Gamma_{\text{GIN}}, \Gamma_{\text{RPGIN}}$) introduced in Bevilacqua et al. [6]. For IRM, we follow [6], and

Table 2: Average Matthews Correlation Coefficient (MCC) for standard GNN models over the size-generalization test set. The models have been trained with (✓) and without (✗) our regularization method. The right-most column shows the average improvement brought by our regularization.

Dataset	**NCI1**		**NCI109**		**PROTEINS**		**DD**		Avg.
Reg.	✗	✓	✗	✓	✗	✓	✗	✓	Impr.
PNA	0.19 ± 0.08	0.22 ± 0.07	0.23 ± 0.07	0.24 ± 0.07	0.22 ± 0.12	0.33 ± 0.09	0.23 ± 0.09	0.27 ± 0.08	**+22%**
GCN	0.17 ± 0.06	0.25 ± 0.06	0.15 ± 0.06	0.19 ± 0.06	0.21 ± 0.10	0.29 ± 0.13	0.24 ± 0.07	0.26 ± 0.07	**+30%**
GIN	0.19 ± 0.06	0.23 ± 0.08	0.18 ± 0.05	0.20 ± 0.05	0.25 ± 0.07	0.36 ± 0.11	0.23 ± 0.09	0.25 ± 0.09	**+21%**

Table 3: Comparison of average Matthews Correlation Coefficient (MCC) over the size-generalization test data between the standard GNN models trained with our proposed regularization and previously proposed methods. Highlighted are the **first**, **second**, and **third** best performing models per dataset.

Dataset	**NCI1**	**NCI109**	**PROTEINS**	**DD**
PNA (reg) [ours]	0.22 ± 0.07	$\mathbf{0.24 \pm 0.07}$	$\mathbf{0.33 \pm 0.09}$	$\mathbf{0.27 \pm 0.08}$
GCN (reg) [ours]	$\mathbf{0.25 \pm 0.06}$	0.19 ± 0.06	0.29 ± 0.13	$\mathbf{0.26 \pm 0.07}$
GIN (reg) [ours]	0.23 ± 0.08	0.20 ± 0.05	$\mathbf{0.36 \pm 0.11}$	0.25 ± 0.09
PNA + IRM [1]	0.17 ± 0.07	0.20 ± 0.07	0.21 ± 0.12	0.24 ± 0.08
GCN + IRM [1]	0.22 ± 0.06	0.20 ± 0.06	0.23 ± 0.16	0.23 ± 0.08
GIN + IRM [1]	0.18 ± 0.06	0.15 ± 0.04	0.24 ± 0.08	0.21 ± 0.10
WL kernel [59]	$\mathbf{0.39 \pm 0.00}$	$\mathbf{0.21 \pm 0.00}$	0.00 ± 0.00	0.00 ± 0.00
GC kernel [58]	0.02 ± 0.00	0.01 ± 0.00	$\mathbf{0.29 \pm 0.00}$	0.00 ± 0.00
RPGIN [49]	0.18 ± 0.06	0.16 ± 0.04	0.22 ± 0.08	0.13 ± 0.04
$\Gamma_{\text{1-hot}}$ [6]	0.15 ± 0.05	$\mathbf{0.22 \pm 0.06}$	0.18 ± 0.08	0.22 ± 0.09
Γ_{GIN} [6]	0.24 ± 0.05	0.16 ± 0.07	0.28 ± 0.10	$\mathbf{0.27 \pm 0.05}$
Γ_{RPGIN} [6]	$\mathbf{0.26 \pm 0.05}$	0.19 ± 0.06	0.26 ± 0.07	0.20 ± 0.05

artificially create two environments: one with graphs with size smaller than the median size of the graphs in the training set, and one with graphs with size larger than the median size in the training set.

Hyperparameters and Evaluation Protocol. For the standard GNN models we adopt the same hyperparameters of Bevilacqua et al. [6], which were obtained from a tuning procedure involving number of layers, learning rate, batch size, hidden layers dimension, and regularization terms. All models are trained with early stopping (i.e., taking the weights at the epoch with best performance on the validation set). We do not modify the original hyperparameters when we apply our regularization. We tune the regularization coefficient λ and the coarse ratios C for GIN on the PROTEINS dataset (we found $\lambda = 0.1$ and $C = \{0.8, 0.9\}$ to be the best on the validation set), and we apply these settings to *all models and datasets* when using our regularization, to show that our method can work without extensive (and expensive) hyperparameter tuning. We use the SGC coarsening algorithm [32] to obtain the coarsened versions of the graphs (chosen for its theoretical properties). The only hyperparameter that is tuned on a per-dataset basis is the aggregation strategy used to obtain the features for the nodes in the coarsened versions of the graphs (in particular we take the best performing between 'sum', 'max', and 'mean'). For all other models we use the hyperparameters introduced by their respective papers. Bevilacqua et al. [6] presented results by averaging over 10 runs (each time with a different random seed), but given the high variance observed, we re-run the experiments for all the models and present the results averaged over 50 runs. More details on the hyperparameters can be found in the Appendix and source code is publicly available[1].

5.1 Results

Table 2 shows how the three considered standard GNN models perform on the test set, containing the 10% largest graphs in the dataset, when trained with and without our regularization on the 50% smallest graphs in the dataset. As this split leads to an imbalanced dataset, we follow previous works and report the results in terms of Matthews correlation coefficient (MCC), which has been shown

[1] `https://github.com/DavideBuffelli/SizeShiftReg`

to be more reliable in imbalanced settings with respect to other common metrics [14]. MCC gives a value between -1 and 1, where -1 indicates perfect disagreement, 0 is the value for a random guesser, and 1 indicates perfect agreement between the predictions and the true labels. The results show that the use of the proposed regularization is always beneficial to the performance of the models, and that it leads to an average improvement across datasets of **21 to 30%**.

In Table 3, we compare the standard GNNs trained with our regularization strategy against more expressive models (RPGNN), graph kernels, models trained with the IRM strategy, and the E-invariant models. We notice that on 3 out of 4 datasets, a "standard" GNN trained with our regularization obtains the best performance on the test set composed of graphs with size larger than those present in the training set. Furthermore on all datasets there is at least one model trained with our regularization in the top-3 best performing models. We also confirm previous results [6, 17] showing that IRM is not effective in the graph domain, and that using theoretically more expressive models, like RPGNN, does not necessarily lead to good size-generalization performance. Graph kernels are highly dataset-dependent, and, while they can perform well for some datasets, in many cases they fail to perform better than a random classifier. Perhaps more surprisingly, on 3 out of 4 datasets there is at least one "standard" GNN trained with our regularization that performs comparably or better than the best E-Invariant model. In fact, E-invariant models are tied to the assumed causal model for the generation of graphs, which is not guaranteed to hold reliably for real-world datasets for which we do not have this kind of information. Futhermore, E-invariant models require the computation of induced homomorphism densities over all possible connected k-vertex subgraphs both at training time and at test time. Our method instead tries to *learn* to be robust to size-shifts, does not require any additional computation at inference time, and yet can lead even simple models like GCN to have size-generalization performance comparable to E-invariant models.

Table 4: Average MCC results on the Deezer dataset for models trained with (✓) and without (✗) our regularization.

Dataset	**Deezer**	
Reg.	✗	✓
PNA	0.59 ± 0.06	0.64 ± 0.07
GCN	0.49 ± 0.10	0.59 ± 0.06
GIN	0.55 ± 0.08	0.61 ± 0.07

Finally, in addition to the benchmarks considered by Bevilacqua et al. [6], we also experiment on a dataset from a different domain. In particular, we consider a social network dataset obtained from Deezer [55]. We follow the same evaluation strategy as before: train on the 50% smallest and test on the 10% largest, and we apply our regularization with $\lambda = 0.1$ and $C = \{0.8, 0.9\}$ without any additional hyperparameter tuning. Results shown in Table 4 confirm the effectiveness of our method.

5.2 Ablation Study

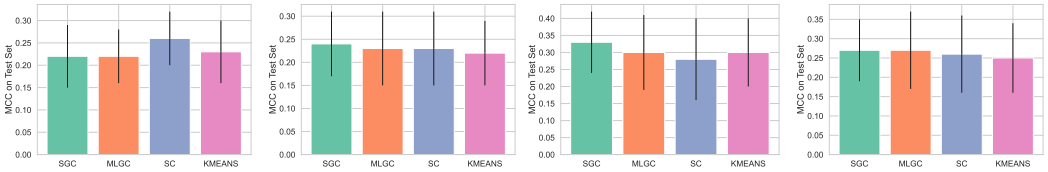
In this section we use a PNA model to study the contribution and importance of the different components of our regularization. We choose PNA because of its performance in the previous analysis, but similar conclusions are observed for GIN and GCN (as we report in the Appendix).

Changing Size of Coarsened Graphs. In Table 5 we train a PNA model with our regularization strategy using different coarsening ratios C. We notice an overall trend in which the performance decreases as the coarsening ratio decreases. This follows intuitively as very low coarsening ratios may lead to uninformative graphs. Furthermore we notice that using all ratios (from 0.1 to 0.9) is usually not effective and setting $C = \{0.8, 0.9\}$ seems a good default option, leading to the best performance on 3 out of 4 datasets.

Table 5: Average MCC results on the size-generalization test set for a PNA model trained with our regularization strategy using different coarsening ratios.

Datasets Ratio(s)	**NCI1**	**NCI109**	**PROTEINS**	**DD**
0.1	0.07 ± 0.11	0.19 ± 0.08	0.12 ± 0.15	0.07 ± 0.14
0.2	0.11 ± 0.12	0.20 ± 0.08	0.18 ± 0.16	0.20 ± 0.11
0.3	0.15 ± 0.09	0.22 ± 0.07	0.17 ± 0.16	0.22 ± 0.10
0.4	0.18 ± 0.08	0.22 ± 0.08	0.23 ± 0.14	0.26 ± 0.11
0.5	0.22 ± 0.08	0.24 ± 0.07	0.29 ± 0.12	0.22 ± 0.09
0.6	0.23 ± 0.08	0.23 ± 0.07	0.26 ± 0.09	0.27 ± 0.10
0.7	0.22 ± 0.07	0.24 ± 0.06	0.30 ± 0.14	0.24 ± 0.06
0.8	0.24 ± 0.06	0.23 ± 0.07	0.32 ± 0.11	0.25 ± 0.09
0.9	0.19 ± 0.06	0.21 ± 0.08	0.28 ± 0.12	0.25 ± 0.10
$\{0.1, 0.9\}$	0.12 ± 0.10	0.20 ± 0.06	0.14 ± 0.16	0.14 ± 0.14
$\{0.5, 0.9\}$	0.23 ± 0.08	0.22 ± 0.07	0.32 ± 0.12	0.27 ± 0.08
$\{0.8, 0.9\}$	0.22 ± 0.07	0.24 ± 0.07	0.33 ± 0.09	0.27 ± 0.08
$\{0.3, 0.7\}$	0.22 ± 0.09	0.21 ± 0.08	0.24 ± 0.11	0.25 ± 0.09
ALL	0.18 ± 0.09	0.18 ± 0.09	0.17 ± 0.14	0.23 ± 0.10

Changing Coarsening Method. In Figure 3 we show how the performance of a PNA model change when it is regularized with our method using graphs coming from different coarsening functions. We consider Spectral Graph Coarsen-



(a) NCI1 (b) NCI109 (c) PROTEINS (d) DD

Figure 3: Average MCC on the test set of a PNA model trained with our regularization strategy using different coarsening methods: Spectral Graph Coarsening (SGC), Multi-level Graph Coarsening (MLGC), Spectral Clustering (SC), K-Means (KMEANS).

ing (SGC) and Multi-level Graph Coarsening (MLGC) introduced by Jin et al. [32], and two baselines, referred to as SC and KMEANS, based on standard spectral clustering [28] and K-Means clustering [40]. For the baselines we apply the clustering algorithm, merge the nodes belonging to the same cluster into a single super-node, and create an edge between two super-nodes if there is at least one edge in the original graph going from a node in one cluster to the other. For all coarsening methods we use the same coarsening ratios as above (i.e., $C = \{0.8, 0.9\}$), and we apply the same training procedure and regularization weight λ, keeping all hyperparameters equal across methods. We notice that our regularization strategy is quite robust towards the choice of the coarsening method, but, on average, specialised methods like SGC and MLGC tend to provide the best results.

6 Related Work

Graph neural networks and size-generalization. Since the first instances of graph neural networks have appeared [60, 57], the field has exploded in recent years with a plethora of different architectures (e.g., [36, 65, 15, 62]), applications (e.g., [69, 67, 37, 23, 8]), and studies on their expressivity and properties (e.g., [65, 44, 47, 9]). For a complete review of the field, we refer to the excellent surveys by Chami et al. [12] and by Wu et al. [64]. Many works have noticed poor size-generalization in GNNs [33, 25, 6, 68]. For instance, Joshi et al. [33] consider the task of learning solutions for the traveling salesman problem, and show that GNNs struggle at generalizing to graphs larger than those appearing in the training set. Similarly, Gasteiger et al. [25] observe that GNNs applied to the molecular domain can be very sensible to several shifts in the data, including shifts in the size of the graphs. Some works have reported good size-generalization by creating ad-hoc models that leverage the properties of the domain [2, 3, 56, 61, 22]. For example, Tsubaki and Mizoguchi [61] impose constraints dictated by the physical properties of their data, and report good size-generalization in predicting the energy of a molecule. While these works show that placing the right inductive biases can help the size-generalization capabilities of GNNs, the current literature is lacking a general method that can be applied on generic GNNs.

Bevilacqua et al. [6] tackle the issue of poor size-generalization by assuming a causal model describing the generative process for the graphs in the dataset, and designing a specific model which can be invariant to the size of the graphs obtained through this causal model. While this strategy shows great size-generalization capabilities on synthetic graphs generated according to their causal model, its benefits decrease when applied to real-world graphs where there are no guarantees that the causal model is correct. This method requires an ad-hoc model and additional computations at both train and test time, while our method can be applied on *any* GNN and does not require pre-computations at test time. Yehudai et al. [68] provide a theoretical and empirical analysis showing that, even for simple tasks, it is not trivial to obtain a GNN with good size-generalization properties. Yehudai et al. [68] consider a different scenario from ours, as, together with the labelled training graphs, they assume access to graphs from the test distribution, and treat the task as a domain-adaptation scenario.

Graph coarsening. While there is no consensus on what is the best metric to consider for coarsening a graph, many methods and metrics have been proposed in recent years [45, 43, 18, 7, 32]. There have also been some attempts at using GNNs for the task of graph coarsening [10, 46]. Orthogonally, graph coarsening has been used to reduce the computational resources needed for training GNNs on large graphs [71, 31]. For a detailed presentation of graph coarsening we refer to Chen et al. [13].

Invariant risk minimization, domain adaptation, and regularization with discrepancy measures. Our method is conceptually related to Invariant Risk Minimization (IRM) [1], which aims at learning representations that are invariant across training environments (where different training environments are intended as sets of data points collected under different conditions). IRM has been shown to have several shortcomings, specially when the data comes from a single environment (e.g. when there is access only to small graphs) [54, 6]. Our method does not require access to different training environments, which may not be easy to obtain in practical scenarios, and our results show that, as also observed in [6, 17], IRM seems ineffective for improving size-generalization in GNNs.

Similarly to IRM, the field of domain adaptation [4] subsumes access to (unlabelled) external data, in addition to labelled training data, and aims at transferring knowledge between domains. Domain adaptation has been used in a large variety of fields, and many surveys are available [16, 63, 27]. Ding et al. [17] analyze how existing domain adaptation algorithms perform on graph data, and observe that these tend to not be effective, indicating that new methods specific for graph data are needed.

Discrepancy measures that have been used for regularizing deep learning models are the Maximum mean discrepancy (MMD) [19, 41, 66, 42] and the central moment discrepancy (CMD) [52, 70, 72], which has been shown to be empirically more effective. We further mention the work by Zhu et al. [72] which, similarly to ours, uses CMD to obtain a GNN that is robust to biases in the sampling process that is used to select the nodes for the training set in node classification scenarios. Finally, we mention the work by Joshi et al. [34] which uses CMD to study the representations of GNNs.

7 Conclusions

GNNs are heavily used models for the task of graph classification. In many domains it is typical for graphs to vary in size, and while GNNs are designed to be able to process graphs of any size, empirical results across literature have highlighted that GNNs struggle at generalizing to sizes unseen during training. In this work we introduce a regularization strategy that can be applied on any GNN, and that improves size-generalization performance from smaller to larger graphs by up to 30%.

Acknowledgments and Disclosure of Funding

The authors thank Chaitanya Joshi and Iulia Duta for the great feedback provided on earlier versions of the paper.

This work is supported, in part, by the Italian Ministry of Education, University and Research (MIUR), under PRIN Project n. 20174LF3T8 "AHeAD" and the initiative "Departments of Excellence" (Law 232/2016), and by University of Padova under project "SID 2020: RATED-X".

References

[1] M. Arjovsky, L. Bottou, I. Gulrajani, and D. Lopez-Paz. Invariant risk minimization. *arXiv preprint arXiv:1907.02893*, 2019.

[2] P. Battaglia, R. Pascanu, M. Lai, D. Jimenez Rezende, et al. Interaction networks for learning about objects, relations and physics. *Advances in neural information processing systems*, 29, 2016.

[3] I. Bello, H. Pham, Q. V. Le, M. Norouzi, and S. Bengio. Neural combinatorial optimization with reinforcement learning. *arXiv preprint arXiv:1611.09940*, 2016.

[4] S. Ben-David, J. Blitzer, K. Crammer, A. Kulesza, F. Pereira, and J. W. Vaughan. A theory of learning from different domains. *Machine learning*, 79(1):151–175, 2010.

[5] Y. Bengio, A. Lodi, and A. Prouvost. Machine learning for combinatorial optimization: A methodological tour d'horizon. *European Journal of Operational Research*, 290(2):405–421, apr 2021. doi: 10.1016/j.ejor.2020.07.063. URL `https://doi.org/10.1016%2Fj.ejor.2020.07.063`.

[6] B. Bevilacqua, Y. Zhou, and B. Ribeiro. Size-invariant graph representations for graph classification extrapolations. In M. Meila and T. Zhang, editors, *Proceedings of the 38th International Conference on Machine Learning, ICML 2021, 18-24 July 2021, Virtual Event*, volume 139 of *Proceedings of Machine Learning Research*, pages 837–851. PMLR, 2021. URL `http://proceedings.mlr.press/v139/bevilacqua21a.html`.

[7] G. Bravo-Hermsdorff and L. M. Gunderson. *A Unifying Framework for Spectrum-Preserving Graph Sparsification and Coarsening*. Curran Associates Inc., Red Hook, NY, USA, 2019.

[8] D. Buffelli and F. Vandin. Graph representation learning for multi-task settings: a meta-learning approach. In *2022 International Joint Conference on Neural Networks (IJCNN)*, pages 1–8, 2022. doi: 10.1109/IJCNN55064.2022.9892010.

[9] D. Buffelli and F. Vandin. The impact of global structural information in graph neural networks applications. *Data*, 7(1), 2022. ISSN 2306-5729. doi: 10.3390/data7010010. URL `https://www.mdpi.com/2306-5729/7/1/10`.

[10] C. Cai, D. Wang, and Y. Wang. Graph coarsening with neural networks. In *International Conference on Learning Representations*, 2021. URL `https://openreview.net/forum?id=uxpzitPEooJ`.

[11] Q. Cappart, D. Chételat, E. B. Khalil, A. Lodi, C. Morris, and P. Veličković. Combinatorial optimization and reasoning with graph neural networks. In *Proceedings of the Thirtieth International Joint Conference on Artificial Intelligence*. International Joint Conferences on Artificial Intelligence Organization, aug 2021. doi: 10.24963/ijcai.2021/595. URL `https://doi.org/10.24963%2Fijcai.2021%2F595`.

[12] I. Chami, S. Abu-El-Haija, B. Perozzi, C. Ré, and K. Murphy. Machine learning on graphs: A model and comprehensive taxonomy. *arXiv preprint arXiv:2005.03675*, 2020.

[13] J. Chen, Y. Saad, and Z. Zhang. Graph coarsening: from scientific computing to machine learning. *SeMA Journal*, 79(1):187–223, jan 2022. doi: 10.1007/s40324-021-00282-x. URL `https://doi.org/10.1007%2Fs40324-021-00282-x`.

[14] D. Chicco and G. Jurman. The advantages of the matthews correlation coefficient (MCC) over f1 score and accuracy in binary classification evaluation. *BMC Genomics*, 21(1), jan 2020. doi: 10.1186/s12864-019-6413-7. URL `https://doi.org/10.1186%2Fs12864-019-6413-7`.

[15] G. Corso, L. Cavalleri, D. Beaini, P. Liò, and P. Veličković. Principal neighbourhood aggregation for graph nets. In H. Larochelle, M. Ranzato, R. Hadsell, M. Balcan, and H. Lin, editors, *Advances in Neural Information Processing Systems*, volume 33, pages 13260–13271. Curran Associates, Inc., 2020. URL `https://proceedings.neurips.cc/paper/2020/file/99cad265a1768cc2dd013f0e740300ae-Paper.pdf`.

[16] G. Csurka. *A Comprehensive Survey on Domain Adaptation for Visual Applications*, pages 1–35. Springer International Publishing, Cham, 2017. ISBN 978-3-319-58347-1. doi: 10.1007/978-3-319-58347-1_1. URL `https://doi.org/10.1007/978-3-319-58347-1_1`.

[17] M. Ding, K. Kong, J. Chen, J. Kirchenbauer, M. Goldblum, D. Wipf, F. Huang, and T. Goldstein. A closer look at distribution shifts and out-of-distribution generalization on graphs. In *NeurIPS 2021 Workshop on Distribution Shifts: Connecting Methods and Applications*, 2021. URL `https://openreview.net/forum?id=XvgPGWazqRH`.

[18] D. Durfee, Y. Gao, G. Goranci, and R. Peng. Fully dynamic spectral vertex sparsifiers and applications. In *Proceedings of the 51st Annual ACM SIGACT Symposium on Theory of Computing*, pages 914–925, 2019.

[19] G. K. Dziugaite, D. M. Roy, and Z. Ghahramani. Training generative neural networks via maximum mean discrepancy optimization. In *Proceedings of the Thirty-First Conference on Uncertainty in Artificial Intelligence*, pages 258–267, 2015.

[20] W. Falcon et al. Pytorch lightning. *GitHub. Note: https://github.com/PyTorchLightning/pytorch-lightning*, 3, 2019.

[21] M. Fey and J. E. Lenssen. Fast graph representation learning with pytorch geometric, 2019. URL https://arxiv.org/abs/1903.02428.

[22] Z.-H. Fu, K.-B. Qiu, and H. Zha. Generalize a small pre-trained model to arbitrarily large tsp instances. *Proceedings of the AAAI Conference on Artificial Intelligence*, 35(8):7474–7482, May 2021. URL https://ojs.aaai.org/index.php/AAAI/article/view/16916.

[23] V. Fung, J. Zhang, E. Juarez, and B. G. Sumpter. Benchmarking graph neural networks for materials chemistry. *npj Computational Materials*, 7(1):1–8, 2021.

[24] M. Gasse, D. Chételat, N. Ferroni, L. Charlin, and A. Lodi. *Exact Combinatorial Optimization with Graph Convolutional Neural Networks*. Curran Associates Inc., Red Hook, NY, USA, 2019.

[25] J. Gasteiger, M. Shuaibi, A. Sriram, S. Günnemann, Z. Ulissi, C. L. Zitnick, and A. Das. How do graph networks generalize to large and diverse molecular systems?, 2022. URL https://arxiv.org/abs/2204.02782.

[26] J. Gilmer, S. S. Schoenholz, P. F. Riley, O. Vinyals, and G. E. Dahl. Neural message passing for quantum chemistry. In *Proceedings of the 34th International Conference on Machine Learning - Volume 70*, ICML'17, pages 1263–1272. JMLR.org, 2017.

[27] H. Guan and M. Liu. Domain adaptation for medical image analysis: a survey. *IEEE Transactions on Biomedical Engineering*, 2021.

[28] C. Guo, S. Zheng, Y. Xie, and W. Hao. A survey on spectral clustering. In *World Automation Congress 2012*, pages 53–56, 2012.

[29] W. L. Hamilton. Graph representation learning. *Synthesis Lectures on Artificial Intelligence and Machine Learning*, 14(3):1–159, 2020.

[30] C. R. Harris, K. J. Millman, S. J. van der Walt, R. Gommers, P. Virtanen, D. Cournapeau, E. Wieser, J. Taylor, S. Berg, N. J. Smith, R. Kern, M. Picus, S. Hoyer, M. H. van Kerkwijk, M. Brett, A. Haldane, J. F. del Río, M. Wiebe, P. Peterson, P. Gérard-Marchant, K. Sheppard, T. Reddy, W. Weckesser, H. Abbasi, C. Gohlke, and T. E. Oliphant. Array programming with NumPy. *Nature*, 585(7825):357–362, Sept. 2020. doi: 10.1038/s41586-020-2649-2. URL https://doi.org/10.1038/s41586-020-2649-2.

[31] W. Jin, L. Zhao, S. Zhang, Y. Liu, J. Tang, and N. Shah. Graph condensation for graph neural networks. In *International Conference on Learning Representations*, 2022. URL https://openreview.net/forum?id=WLEx3Jo4QaB.

[32] Y. Jin, A. Loukas, and J. JaJa. Graph coarsening with preserved spectral properties. In S. Chiappa and R. Calandra, editors, *Proceedings of the Twenty Third International Conference on Artificial Intelligence and Statistics*, volume 108 of *Proceedings of Machine Learning Research*, pages 4452–4462. PMLR, 26–28 Aug 2020. URL https://proceedings.mlr.press/v108/jin20a.html.

[33] C. K. Joshi, Q. Cappart, L.-M. Rousseau, and T. Laurent. Learning TSP Requires Rethinking Generalization. In L. D. Michel, editor, *27th International Conference on Principles and Practice of Constraint Programming (CP 2021)*, volume 210 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 33:1–33:21, Dagstuhl, Germany, 2021. Schloss Dagstuhl – Leibniz-Zentrum für Informatik. ISBN 978-3-95977-211-2. doi: 10.4230/LIPIcs.CP.2021.33. URL https://drops.dagstuhl.de/opus/volltexte/2021/15324.

[34] C. K. Joshi, F. Liu, X. Xun, J. Lin, and C.-S. Foo. On representation knowledge distillation for graph neural networks. *arXiv preprint arXiv:2111.04964*, 2021.

[35] I. Jovanović and Z. Stanić. Spectral distances of graphs. *Linear Algebra and its Applications*, 436(5):1425–1435, 2012. ISSN 0024-3795. doi: https://doi.org/10.1016/j.laa.2011.08.019. URL https://www.sciencedirect.com/science/article/pii/S0024379511006021.

[36] T. N. Kipf and M. Welling. Semi-supervised classification with graph convolutional networks. In *International Conference on Learning Representations (ICLR)*, 2017.

[37] J. Klicpera, F. Becker, and S. Günnemann. Gemnet: Universal directional graph neural networks for molecules. In *Advances in Neural Information Processing Systems*, 2021.

[38] S. Kornblith, M. Norouzi, H. Lee, and G. Hinton. Similarity of neural network representations revisited. In *International Conference on Machine Learning*, pages 3519–3529. PMLR, 2019.

[39] R. Liaw, E. Liang, R. Nishihara, P. Moritz, J. E. Gonzalez, and I. Stoica. Tune: A research platform for distributed model selection and training. *arXiv preprint arXiv:1807.05118*, 2018.

[40] S. Lloyd. Least squares quantization in pcm. *IEEE Transactions on Information Theory*, 28(2): 129–137, 1982. doi: 10.1109/TIT.1982.1056489.

[41] M. Long, Y. Cao, J. Wang, and M. Jordan. Learning transferable features with deep adaptation networks. In *International conference on machine learning*, pages 97–105. PMLR, 2015.

[42] M. Long, H. Zhu, J. Wang, and M. I. Jordan. Deep transfer learning with joint adaptation networks. In *Proceedings of the 34th International Conference on Machine Learning - Volume 70*, ICML'17, pages 2208–2217. JMLR.org, 2017.

[43] A. Loukas. Graph reduction with spectral and cut guarantees. *Journal of Machine Learning Research*, 20(116):1–42, 2019. URL `http://jmlr.org/papers/v20/18-680.html`.

[44] A. Loukas. What graph neural networks cannot learn: depth vs width. In *International Conference on Learning Representations*, 2019.

[45] A. Loukas and P. Vandergheynst. Spectrally approximating large graphs with smaller graphs. In J. Dy and A. Krause, editors, *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pages 3237–3246. PMLR, 10–15 Jul 2018. URL `https://proceedings.mlr.press/v80/loukas18a.html`.

[46] T. Ma and J. Chen. Unsupervised learning of graph hierarchical abstractions with differentiable coarsening and optimal transport. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, pages 8856–8864, 2021.

[47] C. Morris, M. Ritzert, M. Fey, W. L. Hamilton, J. E. Lenssen, G. Rattan, and M. Grohe. Weisfeiler and leman go neural: Higher-order graph neural networks. In *Proceedings of the AAAI conference on artificial intelligence*, volume 33, pages 4602–4609, 2019.

[48] C. Morris, N. M. Kriege, F. Bause, K. Kersting, P. Mutzel, and M. Neumann. Tudataset: A collection of benchmark datasets for learning with graphs. In *ICML 2020 Workshop on Graph Representation Learning and Beyond (GRL+ 2020)*, 2020. URL `www.graphlearning.io`.

[49] R. Murphy, B. Srinivasan, V. Rao, and B. Ribeiro. Relational pooling for graph representations. In K. Chaudhuri and R. Salakhutdinov, editors, *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pages 4663–4673. PMLR, 09–15 Jun 2019. URL `https://proceedings.mlr.press/v97/murphy19a.html`.

[50] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, A. Desmaison, A. Kopf, E. Yang, Z. DeVito, M. Raison, A. Tejani, S. Chilamkurthy, B. Steiner, L. Fang, J. Bai, and S. Chintala. Pytorch: An imperative style, high-performance deep learning library. In H. Wallach, H. Larochelle, A. Beygelzimer, F. Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems 32*, pages 8024–8035. Curran Associates, Inc., 2019. URL `http://papers.neurips.cc/paper/9015-pytorch-an-imperative-style-high-performance-deep-learning-library.pdf`.

[51] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

[52] M. Peng, Q. Zhang, Y.-g. Jiang, and X.-J. Huang. Cross-domain sentiment classification with target domain specific information. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2505–2513, 2018.

[53] M. Prates, P. H. C. Avelar, H. Lemos, L. C. Lamb, and M. Y. Vardi. Learning to solve NP-complete problems: A graph neural network for decision TSP. *Proceedings of the AAAI Conference on Artificial Intelligence*, 33:4731–4738, jul 2019. doi: 10.1609/aaai.v33i01.33014731. URL `https://doi.org/10.1609%2Faaai.v33i01.33014731`.

[54] E. Rosenfeld, P. Ravikumar, and A. Risteski. The risks of invariant risk minimization. In *International Conference on Learning Representations*, volume 9, 2021.

[55] B. Rozemberczki, O. Kiss, and R. Sarkar. Karate Club: An API Oriented Open-source Python Framework for Unsupervised Learning on Graphs. In *Proceedings of the 29th ACM International Conference on Information and Knowledge Management (CIKM '20)*, page 3125–3132. ACM, 2020.

[56] A. Sanchez-Gonzalez, N. Heess, J. T. Springenberg, J. Merel, M. Riedmiller, R. Hadsell, and P. Battaglia. Graph networks as learnable physics engines for inference and control. In *International Conference on Machine Learning*, pages 4470–4479. PMLR, 2018.

[57] F. Scarselli, M. Gori, A. C. Tsoi, M. Hagenbuchner, and G. Monfardini. The graph neural network model. *IEEE Transactions on Neural Networks*, 20(1):61–80, 2009. doi: 10.1109/TNN.2008.2005605.

[58] N. Shervashidze, S. Vishwanathan, T. Petri, K. Mehlhorn, and K. Borgwardt. Efficient graphlet kernels for large graph comparison. In D. van Dyk and M. Welling, editors, *Proceedings of the Twelth International Conference on Artificial Intelligence and Statistics*, volume 5 of *Proceedings of Machine Learning Research*, pages 488–495, Hilton Clearwater Beach Resort, Clearwater Beach, Florida USA, 16–18 Apr 2009. PMLR. URL `https://proceedings.mlr.press/v5/shervashidze09a.html`.

[59] N. Shervashidze, P. Schweitzer, E. J. van Leeuwen, K. Mehlhorn, and K. M. Borgwardt. Weisfeiler-lehman graph kernels. *Journal of Machine Learning Research*, 12(77):2539–2561, 2011. URL `http://jmlr.org/papers/v12/shervashidze11a.html`.

[60] A. Sperduti and A. Starita. Supervised neural networks for the classification of structures. *IEEE Transactions on Neural Networks*, 8(3):714–735, 1997. doi: 10.1109/72.572108.

[61] M. Tsubaki and T. Mizoguchi. On the equivalence of molecular graph convolution and molecular wave function with poor basis set. *Advances in Neural Information Processing Systems*, 33: 1982–1993, 2020.

[62] P. Veličković, G. Cucurull, A. Casanova, A. Romero, P. Liò, and Y. Bengio. Graph attention networks. In *International Conference on Learning Representations*, 2018.

[63] G. Wilson and D. J. Cook. A survey of unsupervised deep domain adaptation. *ACM Transactions on Intelligent Systems and Technology*, 11(5):1–46, oct 2020. doi: 10.1145/3400066. URL `https://doi.org/10.1145%2F3400066`.

[64] Z. Wu, S. Pan, F. Chen, G. Long, C. Zhang, and P. S. Yu. A comprehensive survey on graph neural networks. *IEEE Transactions on Neural Networks and Learning Systems*, 32(1):4–24, 2021. doi: 10.1109/TNNLS.2020.2978386.

[65] K. Xu, W. Hu, J. Leskovec, and S. Jegelka. How powerful are graph neural networks? In *International Conference on Learning Representations*, 2019. URL `https://openreview.net/forum?id=ryGs6iA5Km`.

[66] H. Yan, Y. Ding, P. Li, Q. Wang, Y. Xu, and W. Zuo. Mind the class weight bias: Weighted maximum mean discrepancy for unsupervised domain adaptation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2272–2281, 2017.

[67] M. Yasunaga, H. Ren, A. Bosselut, P. Liang, and J. Leskovec. Qa-gnn: Reasoning with language models and knowledge graphs for question answering. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 535–546, 2021.

[68] G. Yehudai, E. Fetaya, E. Meirom, G. Chechik, and H. Maron. From local structures to size generalization in graph neural networks. In M. Meila and T. Zhang, editors, *Proceedings of the 38th International Conference on Machine Learning*, volume 139 of *Proceedings of Machine Learning Research*, pages 11975–11986. PMLR, 18–24 Jul 2021. URL `https://proceedings.mlr.press/v139/yehudai21a.html`.

[69] R. Ying, R. He, K. Chen, P. Eksombatchai, W. L. Hamilton, and J. Leskovec. Graph convolutional neural networks for web-scale recommender systems. In *Proceedings of the 24th ACM SIGKDD international conference on knowledge discovery & data mining*, pages 974–983, 2018.

[70] W. Zellinger, B. A. Moser, T. Grubinger, E. Lughofer, T. Natschläger, and S. Saminger-Platz. Robust unsupervised domain adaptation for neural networks via moment alignment. *Information Sciences*, 483:174–191, 2019. ISSN 0020-0255. doi: https://doi.org/10.1016/j.ins.2019.01.025. URL `https://www.sciencedirect.com/science/article/pii/S0020025519300301`.

[71] C. X. T. L. Zengfeng Huang, Shengzhong Zhang and M. Zhou. Scaling up graph neural networks via graph coarsening. In *In Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD '21)*, 2021.

[72] Q. Zhu, N. Ponomareva, J. Han, and B. Perozzi. Shift-robust gnns: Overcoming the limitations of localized graph training data. In M. Ranzato, A. Beygelzimer, Y. Dauphin, P. Liang, and J. W. Vaughan, editors, *Advances in Neural Information Processing Systems*, volume 34, pages 27965–27977. Curran Associates, Inc., 2021. URL `https://proceedings.neurips.cc/paper/2021/file/eb55e369affa90f77dd7dc9e2cd33b16-Paper.pdf`.

Checklist

1. For all authors...
   (a) Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope? [Yes]
   (b) Did you describe the limitations of your work? [Yes] Section 3.1
   (c) Did you discuss any potential negative societal impacts of your work? [No]
   (d) Have you read the ethics review guidelines and ensured that your paper conforms to them? [Yes]
2. If you are including theoretical results...
   (a) Did you state the full set of assumptions of all theoretical results? [N/A]
   (b) Did you include complete proofs of all theoretical results? [N/A]
3. If you ran experiments...
   (a) Did you include the code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL)? [Yes] In Supplementary material
   (b) Did you specify all the training details (e.g., data splits, hyperparameters, how they were chosen)? [Yes] Section 5 and Appendix
   (c) Did you report error bars (e.g., with respect to the random seed after running experiments multiple times)? [Yes] All experiments were repeated multiple times and all results in the paper have error bars
   (d) Did you include the total amount of compute and the type of resources used (e.g., type of GPUs, internal cluster, or cloud provider)? [Yes] In Appendix
4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets...
   (a) If your work uses existing assets, did you cite the creators? [Yes] Used existing datasets

(b) Did you mention the license of the assets? [Yes] In Supplementary Material
(c) Did you include any new assets either in the supplemental material or as a URL? [Yes] We provide code for reproducing results
(d) Did you discuss whether and how consent was obtained from people whose data you're using/curating? [N/A]
(e) Did you discuss whether the data you are using/curating contains personally identifiable information or offensive content? [N/A]

5. If you used crowdsourcing or conducted research with human subjects...
   (a) Did you include the full text of instructions given to participants and screenshots, if applicable? [N/A]
   (b) Did you describe any potential participant risks, with links to Institutional Review Board (IRB) approvals, if applicable? [N/A]
   (c) Did you include the estimated hourly wage paid to participants and the total amount spent on participant compensation? [N/A]

Appendix

This Appendix contains the following additional material. In Section A we report more statistics regarding the considered datasets. In Section B we report the results for the CKA analysis on the other datasets not shown in the main paper for space limitations. In Section C we report more details on the hyperparameters of the models used for our experiments. In Section D we report the results for the ablation study on the other models not shown in the main paper for space limitations. In Section E we show the overhead (in terms of training time) incurred by our method. In Section F we report information on the computing resources used to perform our experimental results, links to access the datasets, and the licence of the publicly available libraries used in our code.

A Dataset Information

Table 6: Dataset statistics, this table is taken from Yehudai et al. [68], Bevilacqua et al. [6].

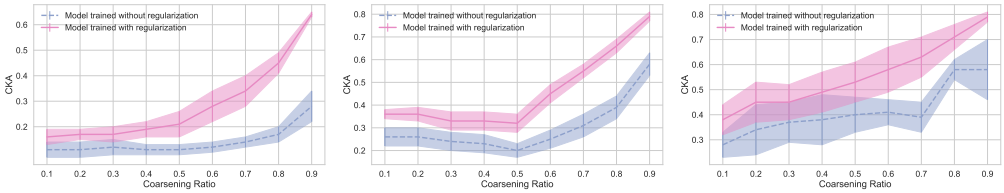
	NCI1			NCI109		
	ALL	SMALLEST 50%	LARGEST 10%	ALL	SMALLEST 50%	LARGEST 10%
CLASS A	49.95%	62.30%	19.17%	49.62%	62.04%	21.37%
CLASS B	50.04%	37.69%	80.82%	50.37%	37.95%	78.62%
NUM OF GRAPHS	4110	2157	412	4127	2079	421
AVG GRAPH SIZE	29	20	61	29	20	61

	PROTEINS			DD		
	ALL	SMALLEST 50%	LARGEST 10%	ALL	SMALLEST 50%	LARGEST 10%
CLASS A	59.56%	41.97%	90.17%	58.65%	35.47%	79.66%
CLASS B	40.43%	58.02%	9.82%	41.34%	64.52%	20.33%
NUM OF GRAPHS	1113	567	112	1178	592	118
AVG GRAPH SIZE	39	15	138	284	144	746

In Table 6 (taken from [68, 6]) we report the information about the graphs in the considered datasets.

B Embedding Analysis Results for Other Datasets

In Figure 4 we show the continuation of Figure 2 (b) from the main paper, including plots on all datasets. In more detail, we report plots for the average CKA [38] values between the node embeddings for the original graphs and the node embeddings for the coarsened versions of the graphs, both generated by the same model, for different ratios. The plot shows that a model trained with our regularization is learning to be more robust to size shifts, as its representation for coarsened versions of the graphs are much more aligned (higher CKA values) to the representations for the original graphs compared to a model trained without regularization. This is even more apparent for low coarsening ratios (i.e., when the shift is stronger), indicating that our regularization is in fact making the model more robust to size shifts. The reason why the CKA values for a model trained with and without regularization tend to become similar for small ratios (which is instead not apparent on the DD dataset shown in Figure 2 (b) in the main paper) is that the graphs in NCI1, NCI109,



(a) NCI1 (b) NCI109 (c) PROTEINS

Figure 4: Average CKA values between the node embeddings generated by the same model for the original graphs in (a) NCI1, (b) NCI109, (c) PROTEINS, and their coarsened versions. A model trained with our regularization produces representations of the coarsened graphs that are more aligned to the representations produced for the original graphs (i.e. it learns to be robust to size shifts).

and PROTEINS are much smaller than the graphs in DD (see Section A), and hence graphs tend to become completely uninformative with ratios lower than 0.5.

C Hyperparameters and Evaluation Procedure

Hyperparameters. As we follow exactly the same procedures as previous work [68, 6], we refer the reader to those papers for an in depth discussion, and we report below the description of the standard GNN models used in our experiments (which follows from the above mentioned papers).

The GCN[36], GIN [65], and PNA [15] models used in our experiments have 3 graph convolution layers and a final multi-layer perceptron with a softmax activation function to obtain the predictions. Batch norm is used in between graph convolution layers, and ReLU is used as activation function. The networks are trained with a dropout of 0.3, and were tuned using the validation set. In particular the batch size was chosen between 64 and 128, the learning rate between $0.01, 0.005$, and 0.001, and the network width between 32 and 64. All models are trained with early stopping (taking the wights related to the epoch with the highest MCC on the validation set), and the different classes were weighted in the supervised loss function according to the frequency of a class in the training data (to deal with the imbalance in the data, as explained in the main paper).

When applying our method in Section 3 and in Table 2 and Table 3 in the main paper, we use $\lambda = 0.1$ and $C = (0.8, 0.9)$ for all models and datasets. These parameters were taken from a tuning procedure on the validation set described below.

Tuning and Evaluation Procedure. To identify the values of λ and C to use for our method, we tried different values ($\lambda = \{1.0, 0.1, 0.01, 0.001\}$ and $C = \{(0.5), (0.8), (0.9), (0.5, 0.8), (0.5, 0.9), (0.8, 0.9)\}$) on the validation set using a GIN model on the PROTEINS dataset. We found that $\lambda = 0.1$ and $C = (0.8, 0.9)$ performed best on the validation set. As the results on the validation sets were good (i.e., they were leading to better performance with respect to a model trained without regularization), we adopted the same values of λ and C for all datasets and models to show that our method can work without extensive (and expensive) hyperparameter tuning. It is possible that dataset-specific and model-specific tuning can lead to higher results.

We first obtained the results for Table 1, Figure 2, Table 2, and Table 3, in the main paper. Then, only *afterwards*, we performed the ablation study. The ablation is used to understand the impact of the components of our method only after having evaluated it, as is the standard procedure for ablation studies.

D Full Ablation Study Results

We report the results of the ablation study also for the GIN and GCN models.

Changing Size of Coarsened Graphs. We report in Table 7 and Table 8 the performance of a GCN [36] and GIN [65] model when trained with our regularization strategy with different coarsening

Table 7: Table shows mean (standard deviation) Matthews correlation coefficient (MCC) for a GCN model trained with our proposed regularization with the Spectral Graph Coarsening (SGC) strategy applied with different coarsening ratios.

Datasets Ratio	**NCI1**	**NCI109**	**PROTEINS**	**DD**
0.1	0.09 ± 0.12	0.23 ± 0.07	0.05 ± 0.14	0.15 ± 0.09
0.2	0.17 ± 0.11	0.23 ± 0.07	0.05 ± 0.16	0.20 ± 0.08
0.3	0.18 ± 0.10	0.23 ± 0.07	0.08 ± 0.18	0.23 ± 0.07
0.4	0.21 ± 0.07	0.24 ± 0.06	0.09 ± 0.17	0.24 ± 0.09
0.5	0.23 ± 0.07	0.22 ± 0.05	0.21 ± 0.14	0.27 ± 0.09
0.6	0.24 ± 0.05	0.21 ± 0.06	0.23 ± 0.17	0.25 ± 0.08
0.7	0.26 ± 0.06	0.21 ± 0.05	0.25 ± 0.16	0.24 ± 0.08
0.8	0.27 ± 0.06	0.20 ± 0.04	0.29 ± 0.15	0.27 ± 0.08
0.9	0.25 ± 0.07	0.19 ± 0.05	0.22 ± 0.14	0.25 ± 0.09
0.1-0.9	0.15 ± 0.11	0.21 ± 0.06	0.09 ± 0.15	0.21 ± 0.09
0.5-0.9	0.22 ± 0.05	0.22 ± 0.06	0.28 ± 0.12	0.25 ± 0.08
0.8-0.9	0.25 ± 0.06	0.19 ± 0.06	0.29 ± 0.13	0.26 ± 0.07
0.3-0.7	0.21 ± 0.07	0.22 ± 0.05	0.10 ± 0.14	0.23 ± 0.07
ALL	0.17 ± 0.08	0.23 ± 0.09	0.12 ± 0.16	0.21 ± 0.09

Table 8: Table shows mean (standard deviation) Matthews correlation coefficient (MCC) for a GIN model trained with our proposed regularization with the Spectral Graph Coarsening (SGC) strategy applied with different coarsening ratios.

Datasets Ratio	**NCI1**	**NCI109**	**PROTEINS**	**DD**
0.1	0.07 ± 0.11	0.04 ± 0.11	0.26 ± 0.12	0.22 ± 0.10
0.2	0.07 ± 0.11	0.07 ± 0.11	0.30 ± 0.10	0.24 ± 0.12
0.3	0.07 ± 0.11	0.10 ± 0.09	0.29 ± 0.15	0.25 ± 0.09
0.4	0.02 ± 0.13	0.09 ± 0.09	0.31 ± 0.15	0.24 ± 0.09
0.5	0.07 ± 0.11	0.10 ± 0.11	0.32 ± 0.12	0.25 ± 0.09
0.6	0.02 ± 0.11	0.08 ± 0.07	0.34 ± 0.12	0.26 ± 0.10
0.7	0.01 ± 0.09	0.09 ± 0.06	0.31 ± 0.14	0.25 ± 0.11
0.8	0.15 ± 0.10	0.15 ± 0.07	0.36 ± 0.11	0.27 ± 0.08
0.9	0.22 ± 0.08	0.19 ± 0.05	0.34 ± 0.10	0.25 ± 0.10
0.1-0.9	0.09 ± 0.10	0.05 ± 0.09	0.32 ± 0.15	0.22 ± 0.10
0.5-0.9	0.03 ± 0.10	0.09 ± 0.07	0.32 ± 0.13	0.23 ± 0.09
0.8-0.9	0.23 ± 0.08	0.20 ± 0.05	0.36 ± 0.11	0.25 ± 0.09
0.3-0.7	0.05 ± 0.12	0.08 ± 0.09	0.26 ± 0.15	0.24 ± 0.10
ALL	0.05 ± 0.12	0.11 ± 0.09	0.25 ± 0.16	0.23 ± 0.10

Table 9: Table shows mean (standard deviation) Matthews correlation coefficient (MCC) for a GCN model trained with our proposed regularization and different coarsening strategies: Spectral Clustering (SC), Spectral Graph Coarsening (SGC), Multilevel Graph Coarsening (MLGC), K-Means (KMEANS).

Datasets	**NCI1**	**NCI109**	**PROTEINS**	**DD**
GCN-SGC	0.25 ± 0.06	0.19 ± 0.06	0.29 ± 0.13	0.26 ± 0.07
GCN-MLGC	0.21 ± 0.07	0.21 ± 0.06	0.27 ± 0.14	0.25 ± 0.06
GCN-SC	0.28 ± 0.07	0.18 ± 0.06	0.24 ± 0.16	0.24 ± 0.07
GCN-KMEANS	0.28 ± 0.06	0.18 ± 0.05	0.25 ± 0.15	0.24 ± 0.07

Table 10: Table shows mean (standard deviation) Matthews correlation coefficient (MCC) for a GIN model trained with our proposed regularization and different coarsening strategies: Spectral Clustering (SC), Spectral Graph Coarsening (SGC), Multilevel Graph Coarsening (MLGC), K-Means (KMEANS).

Datasets	**NCI1**	**NCI109**	**PROTEINS**	**DD**
GIN-SGC	0.23 ± 0.08	0.20 ± 0.05	0.36 ± 0.11	0.25 ± 0.09
GIN-MLGC	0.22 ± 0.07	0.19 ± 0.06	0.36 ± 0.10	0.25 ± 0.10
GIN-SC	0.08 ± 0.11	0.07 ± 0.08	0.32 ± 0.12	0.21 ± 0.10
GIN-KMEANS	0.21 ± 0.09	0.20 ± 0.06	0.35 ± 0.08	0.28 ± 0.09

ratios C_r. All other hyperparameters are kept the same as for the main results in our paper. As for the PNA model shown in the main paper, we notice an overall trend in which the performance decreases as the coarsening ratio decreases. This follows intuitively as very low coarsening ratios may lead to uninformative graphs. Furthermore we notice that using all ratios (from 0.1 to 0.9) is usually not effective and setting $C = \{0.8, 0.9\}$ seems a good default option.

Changing Coarsening Method. In Table 9 and Table 10 we report the performance of a GCN [36] and a GIN [65] model trained with our regularization using different coarsening functions. All other parameters are kept the same. As for the PNA shown in the main paper we notice that our method is quite robust to the choice of the coarsening function. However, in most cases, specialised methods like SGC and MLGC provide the best results.

E Training Times

Table 11 shows the time in seconds for training a model with and without our regularization. We notice that our method introduces an average 50% overhead in training time.

F Compute and Licence Information

All our experiments were performed on a machine with a Nvidia 1080Ti GPU and a CPU cluster equipped with 8 CPUs 12-core Intel Xeon Gold 5118@2.30GHz with 1.5Tb of RAM.

The TUDataset [48] we used for our experiments are publicly available online[2]. Our code makes use of the following publicly available libraries: PyTorch [50] (BSD-Style License), PyTorch Geometric

[2] `https://chrsmrrs.github.io/datasets/`

Table 11: Average (over ten runs) time (s) for performing one epoch during training of standard GNN models trained with (✓) and without (✗) our regularization method. Standard deviation is not reported as it is small and similar for all results.

Dataset	**NCI1**		**NCI109**		**PROTEINS**		**DD**	
Reg.	✗	✓	✗	✓	✗	✓	✗	✓
PNA	2.02	3.1	3.21	4.94	2.33	3.56	3.45	4.88
GCN	1.92	2.91	2.14	3.88	2.29	3.41	3.26	4.74
GIN	1.95	3.02	2.15	3.86	2.3	3.42	3.41	4.75

[21] (MIT License), PyTorch Lightning [20] (Apache-2.0 license), scikit-learn [51] (BSD-3 license), numpy [30] (BSD-3 license), ray[tune] [39] (Apache 2.0).