MEASURING STRUCTURE IN GRAPH BENCHMARK DATASETS USING GRAPH INVARIANTS

Anonymous authors
Paper under double-blind review

ABSTRACT

Progress in graph learning is hindered by benchmark practices that conflate the contributions of node features and graph structure, making it hard to tell whether a model actually learns from the graph, or whether it even needs to. We propose measuring this using *graph invariants*, i.e., permutation-invariant, task-agnostic structural descriptors. Our analysis on 26 datasets substantiates three tacit assumptions in graph learning, namely that (i) a curated subset of invariants is more expressive than standard GNNs, (ii) benchmark datasets exhibit structural heterogeneity even when originating from the same domain, and (iii) simple models are often competitive with, and sometimes exceed, approaches based on transformers or message passing. We thus posit that graph invariants should become a standard tool for measuring graph learning task complexity and the relevance of graph structure.

1 INTRODUCTION

Graphs offer a parsimonious representation for a wide variety of datasets and tasks (Veličković, 2023), giving rise to efficient machine learning models that are predominantly based on the message-passing paradigm (Corso et al., 2024). However, whereas a plethora of graph learning models exists, the *datasets* used to compare and evaluate them have received far less attention, leading to stagnating or even decreasing performance (Bechler-Speicher et al., 2025). Recent work thus started pointing out issues concerning datasets and evaluation practices (Ballester et al., 2025; Bechler-Speicher et al., 2025; Coupette et al., 2025; Palowitch et al., 2022; Tönshoff et al., 2024), including (i) small dataset sizes, which often prevent making statistically significant statements concerning predictive performance, (ii) a lack of structural diversity within datasets, thus preventing model generalization, and (iii) a disproportionate focus on metrics like *expressivity* that are not a priori aligned *with* or even required *for* a specific task.

In the absence of an agreed-upon methodology for assessing these issues, our work emphasizes assessing the graph structure and its impact on tasks. Since common graph learning models tend to intertwine graph structure with node and edge features, we adopt a principled approach that uses *graph invariants* to create structural fingerprints of a graph. Within the field of *complex network analysis* (Newman, 2003), specific types of invariants have already been successfully used. This includes, for instance, motif counts for structural characterizations (Yin et al., 2026) or topological indices for quantifying network complexity (Dehmer and Emmert-Streib, 2015; Dehmer et al., 2017). In graph learning, by contrast, invariants are less prevalent and their use is typically restricted to serving as an additional inductive bias of a model, tailored to improve predictive performance; examples include *homomorphism counts* (Bao et al., 2025; Jin et al., 2024), *clique counts* (Bouritsas et al., 2023) and *Laplace spectra* (Alain et al., 2026). Our work substantially expands on this perspective by using a comprehensive and diverse set of graph invariants that can serve as computable proxies for graph structure. This enables us to study to what extent structure is relevant for solving graph-level tasks in current graph benchmark datasets. Next to raising questions about dataset quality, we contribute a novel experimental evaluation methodology based on *graph invariant baselines*. We show that such baselines are often competitive, providing a simple diagnostic tool to discover model–task misalignment.

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

2 USING GRAPH INVARIANTS IN GRAPH LEARNING

Invariants. Although local aggregation is central to architectures such as graph convolutional networks (Kipf and Welling, 2016) and graph attention networks (Veličković et al., 2018), incorporating global structural information remains challenging. One natural way to encode such information is through graph invariants, i.e., interpretable, permutation-invariant quantities. Formally, let $\mathcal{G}$ denote the space of finite undirected graphs without self-loops. A function $I\colon \mathcal{G} \to \mathbb{R}$ is a graph invariant if, for any two isomorphic graphs $G, H \in \mathcal{G}$, we have $I(G) = I(H)$. For example, the *Wiener index* (Wiener, 1947) measures the spread of a connected graph $G = (V, E)$ by summing shortest-path distances over all pairs of vertices. Path graphs have a large Wiener index, whereas complete graphs have a small Wiener index. The Wiener index has therefore been widely used as a simple descriptor of molecular structure and biological activity (Rouvray, 2002). Numerous other such graph invariants (see Section A.1 for an overview of the ones used in this paper) provide a natural lens through which to study graph expressivity. While a *single* invariant is usually insufficient to distinguish all non-isomorphic graphs, families of invariants from different domains can capture increasingly rich structural information. Thus, graph invariants offer a concrete way to encode global graph structure rather than only local aggregation patterns. Similarly, local topological profiles have previously served as structural descriptors (Adamczyk and Czech, 2023).

Using invariants with machine learning models. To use graph invariants as structural fingerprints, we require an approach that lets us measure their expressive and predictive power on a multitude of datasets in a reproducible fashion while remaining resource-efficient. At the same time, a suitable model should perform at comparable levels *across* datasets to make for a suitable baseline. An obstruction to all three, i.e., reproducibility, comparability, and ease of application, is that model performance usually depends on the choice of hyperparameters that themselves depend on the specific task. Hence, even if finding a suitable set of hyperparameters is possible, it may take an infeasible amount of resources. To reduce the number of manual choices as much as possible while remaining flexible with respect to the choice of dataset, we thus propose using XGBoost (Chen and Guestrin, 2016) as a base model, providing feature vectors built from invariants as inputs. Compared to GNNs, XGBoost is computationally more efficient, exposes fewer hyperparameters, and is therefore considerably easier to tune. Specifically, XGBoost permits fully-automated hyperparameter tuning via Optuna (Akiba et al., 2019), resulting in high flexibility and reproducibility; in fact, we can easily apply the same tuning procedure to new datasets, obtaining an overall *robust* and flexible baseline model. By contrast, the higher complexity of GNNs results in a substantially slower tuning procedure; for our experiments in Section 3.3, we observe that GNNs finish at most a single tuning iteration in the time it takes to finish tuning XGBoost.

Selecting invariants. Another obstruction to a reproducible invariant baseline model is the selection of invariants for a specific task. We ideally would want a *small*, maximally *independent* set of invariants, each of which is *computationally efficient*. At the same time, we want invariants to constitute a comprehensive structural summary of a graph. We first prioritize comprehensiveness and therefore include even strongly-correlated invariants (cf. Figure S.1) in our experiments, since we observe that this does not markedly reduce predictive performance in *any* of the tasks we investigate. This approach, while computationally inefficient, constitutes a practical way of gauging upper bounds for predictive performance while enabling us to avoid manually curating invariants per task/dataset (which would contradict the idea of a *general* baseline model based on graph invariants). Practically, we observe that invariant calculation (which has to be performed only *once* per dataset) is feasible for typical graph-level tasks (Table S.6) and can in fact be trivially parallelized. However, to accommodate datasets in which the size of individual graphs makes the calculation of certain invariants infeasible, we also define smaller subsets of invariants and empirically show that they provide similar insights (cf. Section B).

3 EXPERIMENTS: HOW INVARIANTS SUPPORT GRAPH LEARNING TASKS

In the following, we assess the utility of invariants in three different scenarios. The commonality of all subsequent experiments is that, whenever possible, we precompute invariants for *all* graphs of a dataset. Subsequently, we use them as static features for expressivity analysis as well as augmentations of graph-level representations for graph-level prediction tasks.

Table 1: Excerpt from Wang and Zhang (2024) with six additional rows showing the performance of invariant-based approaches for the full invariant set $\mathtt{I}$, the expressive subset $\mathtt{S}$, and their efficient-regime counterparts $\mathtt{I}_{\text{eff}}$ and $\mathtt{S}_{\text{eff}}$ (cf. Table S.7). "(raw)" denotes differentiated pairs based on differences with an error tolerance of 10^{-6} between invariant vectors. "(MLP)" denotes an MLP trained on invariant vectors, following Wang and Zhang (2024). For this expressivity experiment, we observe *equal performance* between both types of approaches.

		Basic Graphs (60)		Regular Graphs (140)		Extension Graphs (100)		CFI Graphs (100)		Total (400)	
Type	Model	Number	Accuracy	Number	Accuracy	Number	Accuracy	Number	Accuracy	Number	Accuracy
Non-GNNs	3-WL	60	100%	50	35.7%	100	100%	60	60.0%	270	67.5%
	SPD-WL	16	26.7%	14	11.7%	41	41%	12	12%	83	20.8%
	N_2	60	100%	138	98.6%	100	100%	0	0%	298	74.5%
Subgraph GNNs	NGNN	59	98.3%	48	34.3%	59	59%	0	0%	166	41.5%
	DSS-GNN	58	96.7%	48	34.3%	100	100%	15	15%	221	55.2%
	SUN	60	100%	50	35.7%	100	100%	13	13%	223	55.8%
	GNN-AK	60	100%	50	35.7%	97	97%	15	15%	222	55.5%
	I^2-GNN	60	100%	100	71.4%	100	100%	21	21%	281	70.2%
k-WL GNNs	PPGN	60	100%	50	35.7%	100	100%	23	23%	233	58.2%
Substructure GNNs	GSN	60	100%	99	70.7%	95	95%	0	0%	254	63.5%
Random GNNs	DropGNN	52	86.7%	41	29.3%	82	82%	2	2%	177	44.2%
	OSAN	56	93.3%	8	5.7%	79	79%	5	5%	148	37%
Transformer GNNs	Graphormer	16	26.7%	12	8.6%	41	41%	10	10%	79	19.8%
Ours	$\mathtt{I}$ (raw)	60	100%	120	85.7%	100	100%	12	12%	292	73%
	$\mathtt{I}$ (MLP)	60	100%	120	85.7%	100	100%	12	12%	292	73%
	$\mathtt{S}$ (raw)	60	100%	120	85.7%	100	100%	12	12%	292	73%
	$\mathtt{I}_{\text{eff}}$ (raw)	60	100%	119	85%	100	100%	12	12%	291	72.8%
	$\mathtt{S}_{\text{eff}}$ (raw)	60	100%	119	85%	100	100%	12	12%	291	72.8%

Our main questions are:

Q1: How *expressive* are invariants?
Q2: To what extent do invariants *characterize* existing datasets?
Q3: Do invariants *capture* the relevance of graph structure?

3.1 Invariants are expressive

Certain types of invariants (e.g., *local homomorphism counts*, *Laplacian eigenvalues*, *molecular indices*) have already been shown to capture relevant structural information for graph learning tasks (Alain et al., 2024a; Dwivedi and Bresson, 2020; Jin et al., 2024; Moriwaki et al., 2018). We thus hypothesize that the combination of multiple invariants should be capable of capturing structural information across many classes of graphs. To quantify this, we investigate the ability of invariants to differentiate between non-isomorphic pairs of graphs.

Our experiment uses the BREC dataset (Wang and Zhang, 2024), which consists of 400 pairs of graphs that are not distinguishable by 1-WL, the Weisfeiler–Lehman test for graph isomorphism (Morris et al., 2023). This dataset is commonly used to assess the realized expressivity of graph learning models. To determine whether an encoding method can distinguish between two graphs, the *encodings* of both graphs are compared to *encodings* of their isomorphic relabellings, with the requirement that differences between the pair have to be consistently large, whereas differences between their relabelled versions have to be consistently small (Wang and Zhang, 2024). Since graph invariants are *invariant* under any graph isomorphism, this criterion is substantially simplified and only requires us to check whether two invariants have the *same* value for a pair of non-isomorphic graphs.

Table 1 compares the differentiated pairs of graphs using the full set of invariants, denoted by $\mathtt{I}$, versus the methods listed in Wang and Zhang (2024). In this comparison, invariants are the *second* most expressive method after N_2, a 2-hop neighbourhood counting method (Papp and Wattenhofer, 2022), outperforming *all* GNN-based methods. We note that by *combining* invariants with existing methods, additional expressivity gains are possible. For instance, local homomorphism counts can be combined with PPGN (Maron et al., 2019) to differentiate up to 76.25% of all pairs (Jin et al., 2024).[1]

[1]Care must be taken to perform the necessary ablations, though, since our analysis shows that homomorphism counts on their own are not sufficient to distinguish between large numbers of graphs on the BREC dataset.



Figure 1: Heatmap that shows BREC graph pairs on the x-axis and invariants on the y-axis. The colour of a cell indicates the difference of an invariant value when computed for both graphs. Brighter cells indicate larger differences. The first four rows form an expressive subset.

A closer analysis of our full results indicates that expressivity does not require *all* invariants. Through greedy selection, i.e., by choosing invariants that differentiate the largest number of pairs on BREC, we observe that a subset of four invariants is sufficient to capture the *same* expressivity as all combined invariants. Figure 1 shows this *expressive subset*, denoted by $\mathtt{S}$, with the shading of each cell indicating the relative difference when evaluating the invariants. The expressive subset comprises (i) analytic torsion, (ii) mean commute time, (iii) magnitude, and (iv) the graph radius. Looking at the mean pairwise distance correlation between invariants over datasets in Figure S.1, we see that this subset is *not* strongly correlated. We furthermore observe that the differentiated pairs of graphs substantially differ between invariants; the invariants are thus *complementary*. Notably, selecting invariants based on *computational efficiency* yields a similar result. That is, removing computationally complex invariants (cf. Table S.6 for compute times), which yields the *efficient* set $\mathtt{I}_{\text{eff}}$ and its expressive subset $\mathtt{S}_{\text{eff}}$, has negligible impact on expressivity (more precisely, the difference is only a *single* pair of graphs). Hence, even for datasets with large graphs, which pose computational bottlenecks for the calculation of some invariants, we are able to choose subsets of invariants that are not redundant, maintain expressivity, and are efficient to compute. More information about the datasets and the selected subsets of invariants is available in Section B and Table S.7, respectively.

In a nutshell

Invariants are expressive on their own. A subset of four invariants is sufficient to reach high expressivity on the BREC dataset. This result also holds when restricting the selection to invariants that remain computationally efficient for large graphs.

3.2 Invariants highlight dataset heterogeneity

Given their effectiveness as "graph fingerprints," we next quantify to what extent invariants serve to *structurally* characterize graph datasets *as a whole*. This analysis is complementary to prior work (Palowitch et al., 2022), which shows that existing benchmark repositories such as the Open Graph Benchmark (Hu et al., 2020) only cover a small fraction of the space of all possible graphs (measured based on the clustering coefficient and the Gini coefficient of the degree distribution). By contrast, our subsequent analysis aims to predict the dataset a graph belongs to based on its structural information only. To this end, we treat dataset characterization as a "meta-classification" task: given a collection of datasets, we enumerate them and assign each graph the index of its source dataset as a label. We then sample graphs from each dataset and split them into training and test data. As input features, we use the *concatenated invariants* of the graph, with the prediction target being dataset membership (see Table S.10 for XGBoost hyperparameters). Test accuracy then indicates to what extent the datasets are structurally separable.

Pairwise confounder-free meta-classification. An obstruction to meaningful measurements of dataset heterogeneity is that simple graph statistics such as the number of nodes already differentiate a large fraction of graphs due to differences in average graph size. To measure dataset heterogeneity beyond these simple statistics, we remove these confounders for the following analysis. To do so, we sample graphs in groups (i.e., one per dataset), that all agree with respect to the following confounders: the number of nodes, the number of edges, and the second degree moment. Hence, for any pair of datasets, this implies that we only sample pairs where these confounders coincide. As a result, the three confounders, as well as all invariants derived from them (such as the first degree moment or the graph density) are *uninformative* for classification. Due to the similar graph sizes in MoleculeNet (Wu et al., 2017) datasets, we are able to sample graphs for an array of pairwise

Table 2: Pairwise meta-classification under matched sampling on MoleculeNet datasets. We repeat sampling and classification with the full set of invariants for 5 seeds. Cells show mean test accuracy and standard deviation. All datasets contribute graphs with matched confounder invariants, so confounders carry no signal. Pairs with fewer than 400 matched graphs per dataset are excluded and marked with "—".

	BACE	BBBP	HIV	Lipo	PCBA	QM9
BACE	—	0.856 ± 0.028	0.885 ± 0.008	0.884 ± 0.024	0.887 ± 0.016	—
BBBP	0.856 ± 0.028	—	0.670 ± 0.014	0.660 ± 0.029	0.669 ± 0.023	—
HIV	0.885 ± 0.008	0.670 ± 0.014	—	0.687 ± 0.024	0.579 ± 0.018	0.718 ± 0.019
Lipo	0.884 ± 0.024	0.660 ± 0.029	0.687 ± 0.024	—	0.665 ± 0.018	—
PCBA	0.887 ± 0.016	0.669 ± 0.023	0.579 ± 0.018	0.665 ± 0.018	—	0.738 ± 0.015
QM9	—	—	0.718 ± 0.019	—	0.738 ± 0.015	—

meta-classification experiments *without* confounders. Whenever possible, we sample up to 1,000 samples per dataset (matched graphs per pair in Table S.12), splitting all sampled graphs into a train/test set with an $80/20$ split. As Table 2 shows, even graphs within the *same* domain are structurally separable via graph invariants.

Meta-classification across domains. Moving to classification *across* domains, we are unable to match all datasets based on confounders, so we sample 1,000 graphs per dataset and otherwise follow the same evaluation procedure as above. As Table S.13 shows, invariants obtain an overall accuracy of $\approx 61\%$ (efficient set $\texttt{I}_{\text{eff}}$) for classifying 26 datasets from different domains (social networks, small molecules, circuits, ...), whereas confounders achieve an accuracy of $\approx 56\%$. Further restricting the classification to one dataset per category (cf. Section D.2), we achieve an accuracy of $\approx 99\%$. The corresponding confusion matrix (Figure S.4) shows that datasets *within* the same domain tend to get confused with one another more than datasets from different domains. Together with the results of the pairwise classification experiment, these results indicate a pronounced dataset shift, which remains prevalent in graphs from the same domain. Since this is known to pose an obstacle for generalization behaviour of graph models (Gui et al., 2022), our analysis serves as a partial explanation as to why we still only observe limited generalization *across* datasets and tasks in graph learning.

In a nutshell

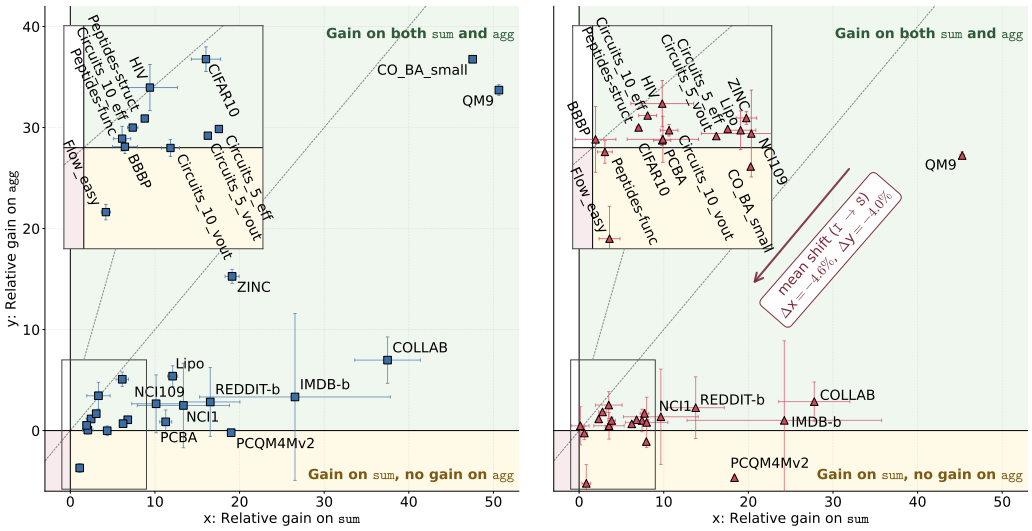
Invariants can characterize existing graph datasets as a whole, highlighting a lack of homogeneous benchmark datasets that is conjectured to negatively impact graph learning research if not addressed (Bechler-Speicher et al., 2025).

3.3 Invariants capture structural relevance

Finally, we carry out an extensive set of graph-level benchmarking operations, analysing the performance of invariants, features, or a mix of both. All figures and tables in this section are based on Table S.14, which shows the results for eight input configurations on 26 datasets. The aim of these experiments is to disentangle the task-specific relevance of (i) features, (ii) message-passing, (iii) expressivity, and (iv) global structure that is *not* captured by expressivity alone. To this end, we evaluate four input configurations, chosen so that combinations of them reveal the contributing factors. Each configuration produces a single feature vector per graph, which is then supplied to XGBoost. For every dataset, we compute the set of tractable invariants specified in Section B. The four configurations are:

1. `sum`: sum-based feature representation
2. `agg`: aggregation-based representation
3. `I`: all invariants
4. `S`: a minimal subset of expressive invariants (cf. Section 3.1)

For datasets of the efficient regime (Section B), `I` and `S` refer to $\texttt{I}_{\text{eff}}$ and $\texttt{S}_{\text{eff}}$, respectively. For the feature-based configurations `sum` and `agg`, we define an initial feature matrix that depends on whether a graph has edge features. If no such features exist, we set $X_{\text{init}} = X$, where X denotes the



(a) Appending `I` (b) Appending `S`

Figure 2: Relative gains from appending invariants. The x-axis shows relative gains over a `sum`-based classifier. The y-axis shows the relative gains over an `agg` based classifier. Zero on a single axis shows the performance of `sum` or `agg`, respectively.

node feature matrix of the graph. For graphs with edge features, we concatenate the sum of features of adjacent edges to each node, that is we define $X_{\text{init}} = X \parallel BE$, where B is the vertex–edge incidence matrix, E the edge feature matrix, and $\parallel$ denotes concatenation. This does *not* increase the expressivity of message passing: for graphs with unit features, where expressivity is conventionally analysed, this operation only appends the node degree. Finally, we calculate $\texttt{sum} \coloneqq \sum_i (X_{\text{init}})_i$ as the *sum* of the feature matrix and $\texttt{agg} \coloneqq \|_i \sum_j (A^i X)_j$, i.e., the iterative *concatenation* of neighbourhood features. This calculation mimics message-passing as found in GIN, for instance, albeit without any nonlinearities or trainable aspects. For the two invariant-based configurations, we concatenate invariants into a feature vector and make predictions based on combinations of these configurations to isolate the role of each factor (i.e., features, message-passing, expressivity, or global structure).

Experimental setup. We keep hyperparameters and their ranges the same for all experiments (see Section C) and perform 150 tuning iterations over the training data with early stopping, followed by a final fit with the *best* set of hyperparameters. The final fit of the model and its subsequent evaluation is repeated for 5 seeds, over which we report mean and standard deviations. We take care to work with official splits and according to the standard evaluation procedure whenever available (see Table S.5 for details). For datasets with an official split, hyperparameters are tuned based on a 10-fold cross-validation over combined train and validation data where feasible and based on validation score alone for datasets with more than 100,000 samples. For datasets with multiple targets, we tune one classifier per target. For cross-validation, hyperparameters are tuned for each outer fold separately, with an inner 10-fold cross-validation over the remaining folds. We then fit and evaluate the best model for each outer fold and report the mean over the results. This follows the suggested cross-validation protocol outlined by Errica et al. (2020) and avoids overoptimistic reporting (O'Bray et al., 2021). We find that when trying to replicate the same approach with a conventional GNN like GIN (Xu et al., 2019), the model finishes *at most* a single tuning iteration in the same time it takes to finish the tuning process for XGBoost. This highlights the computational simplicity of our proposed graph invariant baseline predictor and we conclude that more complex models require larger parameter budgets, with their runtimes making a similar strategy infeasible. We reiterate that our goal is *not* to find the best set of hyperparameters to beat the state of the art, but to provide a reliable, reproducible, and simple performance baseline that can be readily applied to new datasets. In fact, we believe that more complex models will benefit from such comparisons and should, in theory, be able to outperform it.

Insight: The importance of expressivity is questionable. Figure 2 shows the relative performance gains obtained from adding invariants to `sum` and `agg`, respectively. Whenever we observe a performance gain based on combining invariants and aggregated features, we see that the relative gain from adding invariants to summed features is larger. This indicates that our message-passing scheme already captures a substantial amount of the relevant structure—however, being upper-bounded by 1-WL, this structure is not highly expressive. When we reduce the number of invariants to the expressive subset of invariants `S` or to the efficient subset $\texttt{S}_{\text{eff}}$, both relative performance gains *diminish* while expressivity stays the same. In fact, there are even cases where the inverse is true. Predictive performance of the 1-WL bounded `agg` does not increase to the same degree by concatenating invariants on all datasets. Examples are the BBBP and PCQM4Mv2 datasets (cf. Table S.14), where `agg` improves performance over `sum`, but concatenating invariants yields almost no further improvements. This demonstrates that expressivity *on its own* is not a good indicator of predictive performance: Assuming that settings like BREC are valid for measuring expressivity, a subset of invariants with the same expressivity as the full set of invariants nevertheless exhibits reduced predictive performance.

Insight: Invariants improve predictive performance over features alone. We also observe that in many cases, adding invariants to summed or aggregated features improves performance; cf. Figure 3, in which invariants exhibit *consistent* and *substantial* relative performance gains over baselines (both with the full set of invariants and with the subset). These gains are particularly pronounced for the QM9 dataset, in which a combination of invariants and features exhibits competitive performance (cf. Table S.15; we will subsequently discuss this in more detail). Notably, invariants on their own may perform poorly (cf. ZINC in Table S.14) and still capture complementary information to aggregated features. Nevertheless, this analysis suggests that a combination of features and invariants results in a useful baseline model, which, as we demonstrated before, can be efficiently trained. To further contextualize such relative performance gains, we next compare our graph invariant baseline model to results in literature.

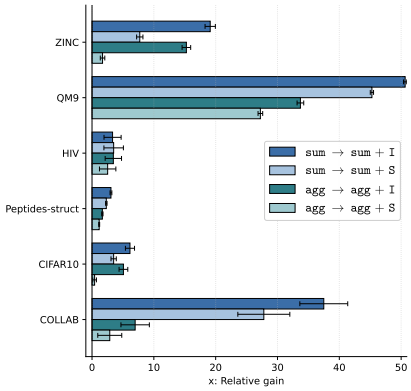


Figure 3: Datasets where we achieve consistent gains by appending invariants to the `sum` or `agg` representation.

Insight: Tree ensembles are sufficient to solve some tasks. We next compare our invariants baseline against results from the literature. Such results usually emerge from a careful manual hyperparameter search, which tends to be more optimistic than what one might expect from a real application. We therefore believe that our task-agnostic tuning procedure should not grant us any unfair advantages. As Figure 4 shows, on many datasets, invariant-based models perform on a par (or better) with more powerful graph learning methods, including transformer models and message-passing architectures. When performing the same comparison for the *minimal expressive subset*, with `agg` + `S` in Figure S.6, we observe slightly reduced predictive performance, which nevertheless remains indistinguishable from that of more complex methods. To substantiate this observation, we perform a *multiple comparisons with the best* analysis (Horrace and Schmidt, 2000; Hsu, 1984, MCB). This analysis indicates that `agg` + `I` is not statistically significantly worse than the best method on the TU, MoleculeNet, and LRGB datasets, respectively (see Section D.3 for details). Notice that for COLLAB, IMDB-b and REDDIT-b, we use the efficient invariant regime, but nevertheless obtain remarkably competitive predictive performance in comparison to more complex methods. Most surprisingly, on novel datasets for solving max-flow problems (Stoll et al., 2026), our graph invariant baseline method even outperforms *all* reported models by a substantial margin. A bootstrap confidence interval analysis further corroborates that these results are *stable* (see Tables S.16 and S.17). We do *not* interpret this as a sign to go back to feature engineering, but rather as a "sanity check" that provides information about the *difficulty* of a task. As a practical consideration, this nevertheless raises questions about the suitability of datasets, highly-parametrized models, and may ultimately indicate the need for additional hyperparameter tuning. We therefore argue that standard baseline comparisons should be extended to include graph invariants in combination with features.

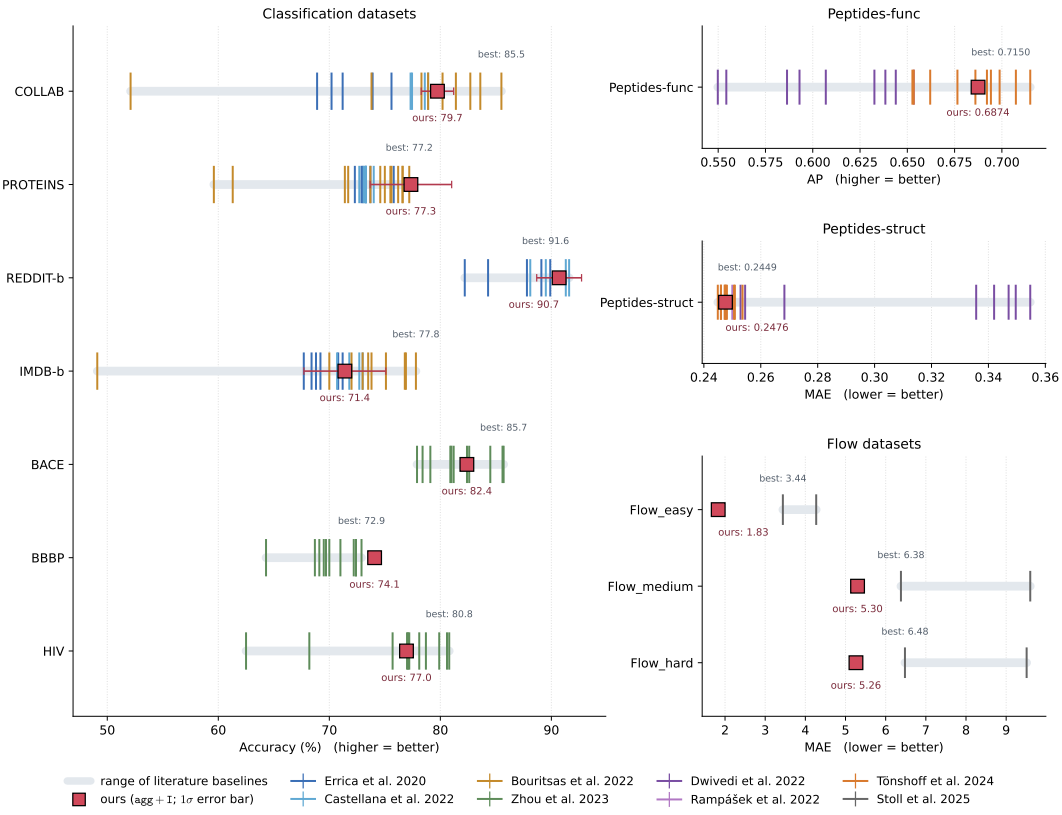


Figure 4: Datasets where `agg+I` performs similarly or better than established methods. Tick colours correspond to source papers of reported performance values. Numeric values are in Table S.18 and Table S.19 for classification datasets, in Table S.20 for Peptides and in Table S.21 for flow datasets.

Recommendation: Invariants should become a standard baseline. Given their empirical performance and simplicity, we believe that invariants constitute a natural minimum baseline, especially for datasets that do not provide features. Furthermore, understanding the relevance of each input configuration helps to find appropriate structural biases. For example, on BACE (Wu et al., 2017), a more complex model should be eschewed in favour of a smaller one since summed features alone lead to the same performance as aggregated features combined with invariants. In other cases, the choice of baseline heavily influences the *perceived* performance of a method. For instance, homomorphism counts (Jin et al., 2024) were compared to GNN-FiLM (Brockschmidt, 2020), with homomorphism counts clearly outperforming GNN-FiLM. However, if we also consider EquiformerV2 (Liao et al., 2024) as a comparison partner, performance differs roughly by a factor of 100. The performance of `agg+I`, by contrast, is situated in-between (i.e., *better* than homomorphism count but worse than EquiformerV2) for most targets on QM9 (cf. Table S.15).

Notice that invariants do not always lead to high performance, as can be seen in Table S.14 for CIFAR10 or "Electronic circuits," for instance. This is unsurprising for CIFAR10 since images are represented as pixel grid graphs. With all graphs then sharing the same underlying topology, purely structural invariants cannot distinguish them.

Nevertheless, we believe that a complex model should be *at least as good* as XGBoost with aggregated features, and ideally even better when adding invariants. This would indicate that such a model is able to extract structure and features in a meaningful manner. We thus posit that simple invariant-based models should become a standard baseline to assess (i) the relevance of structure in graph datasets (by providing a simple structural baseline), as well as (ii) the *ability* of a model to make use of such structure (by comparing model performance to invariant performance); as such, they may serve to address recent issues raised concerning graph machine learning benchmarks (Bechler-Speicher et al., 2025; Coupette et al., 2025).

In a nutshell

Invariants form a minimal yet competitive baseline, expose expressivity as measured on BREC as a poor predictor of task performance, and indicate when graph structure could matter for a task. They should thus be evaluated prior to employing more complex models in order to gauge the suitability of tasks and models early on.

4 Discussion

Graph invariants are simple, permutation-invariant, task-agnostic structural descriptors and serve as a versatile diagnostic lens for disentangling structure from features in graph benchmarks and as a surprisingly competitive baseline for graph learning. Returning to the three questions that frame this work, we found that (i) invariants are more expressive than standard GNNs in the BREC sense, (ii) they characterize structural heterogeneity *within* and *across* benchmark datasets, and, somewhat alarmingly, (iii) simple invariant-based models are statistically on a par with transformer and message-passing baselines on the TU, MoleculeNet, and LRGB benchmarks, and outperform them on the novel Flow datasets.

Limitations. Our analysis is restricted to graph-level tasks, and not all informative invariants are tractable on dense or large graphs. Yet, also with a computationally efficient set of invariants (cf. Section B), invariants retain their diagnostic and predictive merits. Due to the nature of the datasets and the models, gains of models that combine features and invariants can also not directly be attributed to leveraging more structural information; it can only provide a hint as to whether certain aspects of the structure, captured by invariants, are conducive for a task. Hence, the invariant sets we selected for this work should not be taken to be *universally optimal*; our selection was necessarily driven by ease of implementation and we believe that finding "the" comprehensive set of invariants requires a different setup and may ultimately strongly depend on the task.

Implications for graph benchmarks. Two takeaways emerge from our experiments. First, the emphasis on *expressivity* as a primary design objective deserves reconsideration: Invariant subsets with identical expressivity can yield different performance, and a 1-WL-bounded aggregation scheme already captures most of the relevant structure on many datasets. Conversely, this means that a non-expressive model is not automatically a useless one. Second, the structural shifts we observe across domains, and even *within* domains, indicate that current benchmarks do not support learning the structural diversity necessary for foundation models, offering a partial explanation for the lingering difficulty of graph pretraining (Bechler-Speicher et al., 2025; Liu et al., 2025). Rather than proposing a return to hand-crafted features, we consider graph invariants to be a simple, useful diagnostic tool for demonstrating task complexity and model suitability. That is, we see the performance of graph invariants to be a *lower bound* that a model should easily surpass before pursuing more ambitious goals like generalization across domains.

Practical recommendations. Following our extensive experiments, we make the following recommendations for practitioners: **(R1)** For any new graph dataset, fit the four configurations `sum`, `agg`, `I` (or `S`), and their combinations with XGBoost. The resulting baseline helps pinpoint to what extent features, aggregation, or global structure are required. At the same time, this baseline suggests the structural biases a model should incorporate. **(R2)** When computationally intensive methods fail to substantially outperform invariant-based ones, the deficit likely reflects under-tuning or model–task misalignment (Tönshoff et al., 2024); where they do outperform, the gap quantifies the learned interplay between structure and features beyond static structure. **(R3)** If invariants provide no gain over features on a dataset, structure likely only plays a marginal role. This questions its suitability as a *graph* dataset, regardless of its size.

AI USE STATEMENT

In this work, we used generative AI tools for skimming over experimental results to povide an initial asssesment, followed by a more detailed manual assessment. We have *not* used generative AI tools for developing models or conceptual frameworks, proposing or refining hypotheses, designing or providing feedback on research methodology/experiments, implementing methods, assisting with translation, cleaning/reformatting datasets or supporting qualitative/thematic data analysis. The rest of the required disclosure tasks are not applicable to this work.

Additionally, we used generative AI tools for drafting scientific figures, brainstorming, drafting code for already designed experiments, discovering research, searching for information, getting feedback on readability/consistency. We have not used generative AI for the other research subtasks recommended for disclosure. We have reviewed all AI-assisted work. LLM-generated code was verified line by line. Generated code as well as figures were extensively manually adjusted to fit our requirements. We take full responsibility for the final content of this work, including text, claims or artifacts produced with the aid of generative AI.

REPRODUCIBILITY STATEMENT

We have taken the following steps to ensure reproducibility. All datasets are publicly available, Table S.5 lists their sources and the split schemes we use. All graph invariants are formally defined in Section A.1, and the invariant sets used in each regime are specified in Section B. The hardware used for the XGBoost experiments is specified in Section C. The complete hyperparameter configurations and search spaces for all XGBoost and GIN models are listed in Section C. Procedures are described with each experiment: Section 3.1 follows Wang and Zhang (2024). Sampling, matching and the task setup for meta-classification is detailed in Section 3.2. Input configurations and evaluation protocols for XGBoost are detailed in Section 3.3. All results are reported with mean and standard deviation over 5 random seeds, or over 10 folds for datasets evaluated with nested cross validation. We additionally provide bootstrap confidence intervals (Tables S.16 and S.17) and a statistical comparison with published results in Section D.3. Literature values in Tables S.18 to S.21 are taken from the papers cited in the same table. The source code for computing all invariants and reproducing the main results is provided as an anonymous repository under https://anonymous.4open.science/r/measuring-structure-using-invariants.

REFERENCES

J. Adamczyk and W. Czech. Strengthening structural baselines for graph classification using local topological profile. In J. Mikyška, C. de Mulatier, M. Paszynski, V. V. Krzhizhanovskaya, J. J. Dongarra, and P. M. Sloot, editors, *Computational Science – ICCS*, pages 597–611, Cham, Switzerland, 2023. Springer.

T. Akiba, S. Sano, T. Yanase, T. Ohta, and M. Koyama. Optuna: A next-generation hyperparameter optimization framework. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 2623–2631, New York, NY, USA, 2019. Association for Computing Machinery. doi: 10.1145/3292500.3330701.

M. Alain, S. Takao, X. Dong, B. Rieck, and E. Noutahi. Graph classification gaussian processes via hodgelet spectral features. In *Advances in Neural Information Processing Systems (NeurIPS) Workshop on Bayesian Decision-Making and Uncertainty*, 2024a. URL https://arxiv.org/abs/2410.10546.

M. Alain, S. Takao, B. Paige, and M. P. Deisenroth. Gaussian processes on cellular complexes. In R. Salakhutdinov, Z. Kolter, K. Heller, A. Weller, N. Oliver, J. Scarlett, and F. Berkenkamp, editors, *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pages 879–905. PMLR, 2024b.

M. Alain, S. Takao, X. Dong, B. Rieck, and E. Noutahi. Graph and simplicial complex prediction gaussian process via hodgelet representations. In *Advances in Neural Information Processing Systems*, 2026. URL https://arxiv.org/abs/2505.10877.

R. Ballester, E. Röell, D. B. Schmid, M. Alain, S. Escalera, C. Casacuberta, and B. Rieck. MANTRA: The Manifold Triangulations Assemblage. In *International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=X6y5CC44HM.

L. Bao, E. Jin, M. M. Bronstein, I. I. Ceylan, and M. Lanzinger. Homomorphism counts as structural encodings for graph learning. In *International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=qFw2RFJS5g.

M. Bechler-Speicher, B. Finkelshtein, F. Frasca, L. Müller, J. Tönshoff, A. Siraudin, V. Zaverkin, M. M. Bronstein, M. Niepert, B. Perozzi, M. Galkin, and C. Morris. Position: Graph learning will lose relevance due to poor benchmarks. In A. Singh, M. Fazel, D. Hsu, S. Lacoste-Julien, F. Berkenkamp, T. Maharaj, K. Wagstaff, and J. Zhu, editors, *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, pages 81067–81089. PMLR, 2025.

G. Bouritsas, F. Frasca, S. Zafeiriou, and M. M. Bronstein. Improving graph neural network expressivity via subgraph isomorphism counting. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 45(1):657–668, 2023. doi: 10.1109/TPAMI.2022.3154319.

M. Brockschmidt. GNN-FiLM: Graph neural networks with feature-wise linear modulation. In H. Daumé III and A. Singh, editors, *Proceedings of the 37th International Conference on Machine Learning*, volume 119 of *Proceedings of Machine Learning Research*, pages 1144–1152. PMLR, 13–18 Jul 2020.

D. Castellana, F. Errica, D. Bacciu, and A. Micheli. The infinite contextual graph Markov model. In K. Chaudhuri, S. Jegelka, L. Song, C. Szepesvari, G. Niu, and S. Sabato, editors, *Proceedings of the 39th International Conference on Machine Learning*, volume 162 of *Proceedings of Machine Learning Research*, pages 2721–2737. PMLR, 17–23 Jul 2022.

T. Chen and C. Guestrin. XGBoost: A scalable tree boosting system. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 785–794. Association for Computing Machinery, 2016. doi: 10.1145/2939672.2939785.

G. Corso, H. Stark, S. Jegelka, T. Jaakkola, and R. Barzilay. Graph neural networks. *Nature Reviews Methods Primers*, 4(1):17, 2024. doi: 10.1038/s43586-024-00294-7.

C. Coupette, J. Wayland, E. Simons, and B. Rieck. No metric to rule them all: Toward principled evaluations of graph-learning datasets. In A. Singh, M. Fazel, D. Hsu, S. Lacoste-Julien, F. Berkenkamp, T. Maharaj, K. Wagstaff, and J. Zhu, editors, *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, pages 11405–11434. PMLR, 2025.

M. Dehmer and F. Emmert-Streib, editors. *Quantitative Graph Theory: Mathematical Foundations and Applications*. Chapman and Hall/CRC, 2015. doi: 10.1201/b17645.

M. Dehmer, F. Emmert-Streib, and Y. Shi. Quantitative graph theory: A new branch of graph theory and network science. *Information Sciences*, 418-419:575–580, 2017. doi: 10.1016/j.ins.2017.08.009.

J. Demšar. Statistical comparisons of classifiers over multiple data sets. *Journal of Machine Learning Research*, 7(1):1–30, 2006.

Y. Du, W. M. Czarnecki, S. M. Jayakumar, M. Farajtabar, R. Pascanu, and B. Lakshminarayanan. Adapting auxiliary losses using gradient similarity. *arXiv:1812.0224*, 2020.

J. Duda. Simple inexpensive vertex and edge invariants distinguishing dataset strongly regular graphs, 2024.

V. P. Dwivedi and X. Bresson. A generalization of transformer networks to graphs. *arXiv:2012.09699*, 2020.

V. P. Dwivedi, L. Rampášek, M. Galkin, A. Parviz, G. Wolf, A. T. Luu, and D. Beaini. Long range graph benchmark. In *Thirty-sixth Conference on Neural Information Processing Systems Datasets and Benchmarks Track*, 2022.

V. P. Dwivedi, C. K. Joshi, A. T. Luu, T. Laurent, Y. Bengio, and X. Bresson. Benchmarking graph neural networks. *Journal of Machine Learning Research*, 24(43):1–48, 2023.

F. Errica, M. Podda, D. Bacciu, and A. Micheli. A fair comparison of graph neural networks for graph classification. In *International Conference on Learning Representations*, 2020. URL https://openreview.net/forum?id=HygDF6NFPB.

M. Fey and J. E. Lenssen. Fast graph representation learning with Pytorch Geometric. *arXiv:1903.02428*, 2019.

F. Frasca, F. Jogl, M. Eliasof, M. Ostrovsky, C.-B. Schönlieb, T. Gärtner, and H. Maron. Towards foundation models on graphs: An analysis on cross-dataset transfer of pretrained GNNs. In *NeurIPS 2024 Workshop on Symmetry and Geometry in Neural Representations*, 2025.

S. Gui, X. Li, L. Wang, and S. Ji. Good: A graph out-of-distribution benchmark. In S. Koyejo, S. Mohamed, A. Agarwal, D. Belgrave, K. Cho, and A. Oh, editors, *Advances in Neural Information Processing Systems*, volume 35, pages 2059–2073. Curran Associates, Inc., 2022. doi: 10.52202/068431-0150.

W. C. Horrace and P. Schmidt. Multiple comparisons with the best, with economic applications. *Journal of Applied Econometrics*, 15(1):1–26, 2000.

J. C. Hsu. Constrained simultaneous confidence intervals for multiple comparisons with the best. *The Annals of Statistics*, 12(3):1136–1144, 1984.

W. Hu, M. Fey, M. Zitnik, Y. Dong, H. Ren, B. Liu, M. Catasta, and J. Leskovec. Open graph benchmark: Datasets for machine learning on graphs. In H. Larochelle, M. Ranzato, R. Hadsell, M. Balcan, and H. Lin, editors, *Advances in Neural Information Processing Systems*, volume 33, pages 22118–22133. Curran Associates, Inc., 2020.

W. Hu*, B. Liu*, J. Gomes, M. Zitnik, P. Liang, V. Pande, and J. Leskovec. Strategies for pre-training graph neural networks. In *International Conference on Learning Representations*, 2020. URL https://openreview.net/forum?id=HJlWWJSFDH.

E. Jin, M. M. Bronstein, I. I. Ceylan, and M. Lanzinger. Homomorphism counts for graph neural networks: All about that basis. In R. Salakhutdinov, Z. Kolter, K. Heller, A. Weller, N. Oliver, J. Scarlett, and F. Berkenkamp, editors, *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pages 22075–22098. PMLR, 2024.

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

O. Knill. Analytic torsion for graphs. *arXiv:2201.9412*, 2022.

T. Leinster. The magnitude of a graph. *Mathematical Proceedings of the Cambridge Philosophical Society*, 166(2):247–264, 2017. doi: 10.1017/s0305004117000810.

Y.-L. Liao, B. M. Wood, A. Das, and T. Smidt. EquiformerV2: Improved equivariant transformer for scaling to higher-degree representations. In *International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=mCOBKZmrzD.

H. Liu, J. Feng, L. Kong, N. Liang, D. Tao, Y. Chen, and M. Zhang. One for all: Towards training one graph model for all classification tasks. In *International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=4IT2pgc9v6.

J. Liu, C. Yang, Z. Lu, J. Chen, Y. Li, M. Zhang, T. Bai, Y. Fang, L. Sun, P. S. Yu, and C. Shi. Graph foundation models: Concepts, opportunities and challenges. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 47(6):5023–5044, 2025. doi: 10.1109/TPAMI.2025.3548729.

H. Maron, H. Ben-Hamu, H. Serviansky, and Y. Lipman. Provably powerful graph networks. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc., 2019.

H. Moriwaki, Y. Tian, N. Kawashita, and T. Takagi. Mordred: A molecular descriptor calculator. *Journal of Cheminformatics*, 10, 02 2018. doi: 10.1186/s13321-018-0258-y.

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.

C. Morris, Y. Lipman, H. Maron, B. Rieck, N. M. Kriege, M. Grohe, M. Fey, and K. Borgwardt. Weisfeiler and Leman go machine learning: The story so far. *Journal of Machine Learning Research*, 24(333):1–59, 2023.

M. E. J. Newman. The structure and function of complex networks. *SIAM Review*, 45(2):167–256, 2003. doi: 10.1137/S003614450342480.

L. O'Bray, B. Rieck, and K. Borgwardt. Filtration curves for graph representation. In *Proceedings of the 27th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 1267–1275, New York, NY, USA, 2021. Association for Computing Machinery. doi: 10.1145/3447548.3467442.

J. Palowitch, A. Tsitsulin, B. Mayer, and B. Perozzi. GraphWorld: Fake graphs bring real insights for GNNs. In *Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pages 3691–3701. Association for Computing Machinery, 2022. doi: 10.1145/3534678.3539203.

P. A. Papp and R. Wattenhofer. A theoretical comparison of graph neural network extensions. In K. Chaudhuri, S. Jegelka, L. Song, C. Szepesvari, G. Niu, and S. Sabato, editors, *Proceedings of the 39th International Conference on Machine Learning*, volume 162 of *Proceedings of Machine Learning Research*, pages 17323–17345. PMLR, 2022.

J. Qiu, Q. Chen, Y. Dong, J. Zhang, H. Yang, M. Ding, K. Wang, and J. Tang. GCC: Graph contrastive coding for graph neural network pre-training. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, page 1150–1160. Association for Computing Machinery, 2020. doi: 10.1145/3394486.3403168.

C. Ramos-Carreño and J. L. Torrecilla. dcor: Distance correlation and energy statistics in Python. *SoftwareX*, 22, 2023. doi: 10.1016/j.softx.2023.101326.

L. Rampášek, M. Galkin, V. P. Dwivedi, A. T. Luu, G. Wolf, and D. Beaini. Recipe for a general, powerful, scalable graph transformer. In S. Koyejo, S. Mohamed, A. Agarwal, D. Belgrave, K. Cho, and A. Oh, editors, *Advances in Neural Information Processing Systems*, volume 35, pages 14501–14515. Curran Associates, Inc., 2022.

D. H. Rouvray. he rich legacy of half a century of the Wiener index. In *Topology in Chemistry*, pages 16–37. Woodhead Publishing, 2002.

T. Stoll, C. Qian, B. Finkelshtein, A. Parviz, D. Weber, F. Frasca, H. Shavit, A. Siraudin, A. Mielke, M. Anastacio, E. Müller, M. Bechler-Speicher, M. Bronstein, M. Galkin, H. Hoos, M. Niepert, B. Perozzi, J. Tönshoff, and C. Morris. Graphbench: Next-generation graph learning benchmarking. *arXiv:2512.04475*, 2026.

G. J. Székely, M. L. Rizzo, and N. K. Bakirov. Measuring and testing dependence by correlation of distances. *The Annals of Statistics*, 35(6):2769–2794, 2007. doi: 10.1214/009053607000000505.

J. Tönshoff, M. Ritzert, E. Rosenbluth, and M. Grohe. Where did the gap go? reassessing the long-range graph benchmark. *Transactions on Machine Learning Research*, 2024. URL https://openreview.net/forum?id=Nm0WX86sKv.

P. Veličković, G. Cucurull, A. Casanova, A. Romero, P. Liò, and Y. Bengio. Graph attention networks. In *International Conference on Learning Representations*, 2018. URL https://openreview.net/forum?id=rJXMpikCZ.

P. Veličković. Everything is connected: Graph neural networks. *Current Opinion in Structural Biology*, 79:102538, 2023. doi: 10.1016/j.sbi.2023.102538.

Y. Wang and M. Zhang. An empirical study of realized GNN expressiveness. In R. Salakhutdinov, Z. Kolter, K. Heller, A. Weller, N. Oliver, J. Scarlett, and F. Berkenkamp, editors, *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pages 52134–52155. PMLR, 2024.

H. Wiener. Structural determination of paraffin boiling points. *Journal of the American Chemical Society*, 69(1):17–20, 1947.

Z. Wu, B. Ramsundar, E. N. Feinberg, J. Gomes, C. Geniesse, A. S. Pappu, K. Leswing, and V. Pande. MoleculeNet: a benchmark for molecular machine learning. *Chemical Science*, 9(2):513–530, 2017. doi: 10.1039/c7sc02664a.

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.

H. Yin, K. Wang, W. Zhang, Y. He, Y. Zhang, and X. Lin. Motif counting in complex networks: A comprehensive survey. *ACM Computing Surveys*, 2026. doi: 10.1145/3844607.

T. Yu, S. Kumar, A. Gupta, S. Levine, K. Hausman, and C. Finn. Gradient surgery for multi-task learning. In *Proceedings of the 34th International Conference on Neural Information Processing Systems*, Red Hook, NY, USA, 2020. Curran Associates Inc.

G. Zhou, Z. Gao, Q. Ding, H. Zheng, H. Xu, Z. Wei, L. Zhang, and G. Ke. Uni-Mol: A universal 3d molecular representation learning framework. In *International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=6K2RM6wVqKu.

Appendix (Supplementary Materials)

A Invariants and methods **15**

A.1 Invariants 15

A.2 Methods 19

B Dataset regimes and invariant sets **20**

C Hyperparameters **24**

D Additional results **26**

D.1 Expressivity 26

D.2 Meta-classification 28

D.3 Benchmarks 30

E Invariants reveal obstructions to multi-task learning **37**

A Invariants and methods

This section provides technical background for our analyses: the graph invariants used throughout the paper, and an introduction to GNNs.

A.1 Invariants

Here, we present the graph invariants used in the experiments, grouped into basic graph-theoretic invariants (Table S.1), entropy-based invariants (Table S.2), geometric and topological invariants (Table S.3), and graph indices (Table S.4). Before we start, we need to clarify the notation as well as recall and present some concepts.

We consider finite, undirected graphs without self-loops, denoted by $G \coloneqq (V, E)$, where $V \coloneqq \{1, \dots, n_V\}$ is the set of vertices and $E \subseteq \binom{V}{2}$ is the set of edges, with $n_E \coloneqq |E|$. We denote the space of such graphs by $\mathcal{G}$. For a vertex $i \in V$, its neighbourhood is $N(i) \coloneqq \{j \in V \mid \{i, j\} \in E\}$. Vertices and edges may carry features, denoted respectively by $\boldsymbol{v}_i \in \mathbb{R}^{d_V}$ and $\boldsymbol{e}_{ij} \in \mathbb{R}^{d_E}$.

The connectivity of G is encoded by its adjacency matrix $\boldsymbol{A} \in \{0, 1\}^{n_V \times n_V}$, defined by

$$(\boldsymbol{A})_{ij} \coloneqq \begin{cases} 1, & \text{if } j \in N(i), \\ 0, & \text{otherwise.} \end{cases}$$

The degree of a vertex i is $\deg(i) \coloneqq \sum_{j=1}^{n_V} (\boldsymbol{A})_{ij}$, and the shortest-path distance between two vertices $i, j \in V$ is denoted by $d(i, j)$. Equivalently, for $i \neq j$,

$$d(i, j) = \min\{k \geq 1 \mid (\boldsymbol{A}^k)_{ij} > 0\},$$

with the convention that $d(i, j) = \infty$ if i and j lie in different connected components, and $d(i, i) = 0$.

We denote by $\operatorname{spec}(\boldsymbol{M})$ the spectrum of a square matrix $\boldsymbol{M}$, that is, the set of its eigenvalues. The degree matrix $\boldsymbol{D}$ is defined by $(\boldsymbol{D})_{ii} \coloneqq \deg(i)$, and the symmetrically normalized graph Laplacian is

$$\boldsymbol{L}_{\text{sym}} \coloneqq \boldsymbol{I} - \boldsymbol{D}^{-1/2} \boldsymbol{A} \boldsymbol{D}^{-1/2}.$$

If isolated vertices are present, we use the standard convention $(\boldsymbol{D}^{-1/2})_{ii} = 0$ whenever $\deg(i) = 0$.

Table S.1: **Basic graph-theoretic invariants**. The number of unique triangles, i.e. three vertices forming a triangle, is denoted by n_{triangle}, while the number of three vertices connected by at least two edges is n_{triplet}. A graph is a *tree* if it is connected with no *cycles*. A graph $H = (V', E')$ is said to be a *subgraph* of G, denoted by $H \subseteq G$, if $V' \subseteq V$, $E' \subseteq E$, and any edge in E' connects two vertices in V'

Invariant	**Definition**
Number of vertices	n_V
Number of edges	n_E
Circuit rank	$\text{rank}(G)$
Diameter	$\max_{i,j \in V} d(i,j)$
Radius	$\min_{i \in V} \max_{j \in V} d(i,j)$
Transitivity	$3\frac{n_{\text{triangle}}}{n_{\text{triplet}}}$
Density	$\frac{2n_E}{n_V(n_V-1)}$
Eigenvalues of the normalized graph laplacian	$\text{spec}(\boldsymbol{L}_{\text{sym}})$
Natural logarithm of the number of spanning trees	$\ln \lvert\{H \subseteq G \mid H \text{ is a tree and } V' = V\}\rvert$
Geometric over arithmetic average of degree	$\frac{\left(\prod_{i \in V} \deg(i)\right)^{1/n_V}}{\frac{1}{n_V}\sum_{i \in V} \deg(i)}$

Table S.2: **Entropy-based invariants**. The function $\text{order} : \{i, j\} \mapsto (\min(i,j), \max(i,j))$ converts an edge into an ordered pair. The function list takes an edge set, applies order to every edge and then sorts them by lexicographical order to produce a sequence. The function bytes converts a sequence of edges into a byte sequence. The function compress reduces the byte sequence into a smaller or equal byte sequence. The function length returns the number of bytes in a given byte sequence.

Invariant	**Definition**
Degree entropy	$-\sum_{k=1}^{n_V-1} \frac{\lvert\{i \in V \mid \deg(i)=k\}\rvert}{n_V} \log\left(\frac{\lvert\{i \in V \mid \deg(i)=k\}\rvert}{n_V}\right)$
von Neumann entropy	$-\sum_{i=2}^{n_V} \frac{\lambda_i}{n_V} \log\left(\frac{\lambda_i}{n_V}\right), \quad \lambda_i \in \text{spec}(\boldsymbol{L}_{\text{sym}})$
Kolmogorov complexity	$\text{length}(\text{compress}(\text{bytes}(E)))$

The number of connected components of G is denoted by n_{connect}. The circuit rank, also called the cyclomatic number, is

$$\text{rank}(G) \coloneqq n_E - n_V + n_{\text{connect}}.$$

Now, we present below the graph invariants from four families: basic graph-theoretic invariants, entropy-based invariants, geometric and topological invariants, and graph indices.

Table S.3: **Geometric and topological invariants**. The square matrix $\boldsymbol{Z}_G(q) \in \mathbb{R}^{n_V \times n_V}$ is defined such that $(\boldsymbol{Z}_G(q))_{ij} \coloneqq q^{d(i,j)}$, $q \in \mathbb{R}$. For homomorphism counts, $\hom(F, G)$ denotes the set of all homomorphisms from a graph F to G, and $\Omega^{\mathrm{con}}_{\leq 5}$ denotes the family of all connected graphs with up to five vertices. For every $p > 0$, we have the incidence matrix $\boldsymbol{B}_p$ (see (Alain et al., 2024b) for more detail). For $p = 0$, $\boldsymbol{B}_0 = \boldsymbol{0}$. The p-Hodge Laplacian $\boldsymbol{L}_p$ is defined by $\boldsymbol{L}_p \coloneqq \boldsymbol{B}_p^\top \boldsymbol{B}_p + \boldsymbol{B}_{p+1}\boldsymbol{B}_{p+1}^\top$. We denote Wasserstein distance of order 1 by W_1 and μ_i is the probability measure of a lazy random walk associated with the vertex i. Both Forman-Ricci and Ollivier-Ricci curvature form distributions over the edges. We use either the empiric mean, variance, skewness or kurtosis of these distributions as an invariant. The commute time is defined for every pair $(i, j) \in V \times V$, with $\boldsymbol{L}^\dagger$ the Moore–Penrose pseudoinverse of the unnormalized Laplacian $\boldsymbol{L} \coloneqq \boldsymbol{D} - \boldsymbol{A}$ and $\mathrm{vol}(V) = 2n_E$; we use the empirical mean and maximum over $V \times V$ as invariants. For the neighbourhood power trace, $\boldsymbol{A}|_{N(i)}$ denotes the principal submatrix of $\boldsymbol{A}$ indexed by the neighbourhood $N(i)$ of vertex i, following the original definition of Duda (2024). As a variant, we also consider the closed neighbourhood $N[i] \coloneqq N(i) \cup \{i\}$ in place of $N(i)$.

Invariant	**Definition**
Magnitude, Leinster (2017)	$\sum_{i \in V} \sum_{j \in V} (\boldsymbol{Z}(q)^{-1})_{ij}$
Analytic torsion, Knill (2022)	$\prod_{p=0}^{P} \det(\mathbf{L}_p)^{p(-1)^{p+1}}$
Homomorphism counts, Jin et al. (2024)	$\lvert\hom(F, G)\rvert, \quad F \in \Omega^{\mathrm{con}}_{\leq 5}$
Forman–Ricci curvature	$4 - (\deg(i) + \deg(j))$
Ollivier–Ricci curvature	$1 - \frac{W_1(\mu_i, \mu_j)}{d(i,j)}$
Commute time	$\mathrm{vol}(V)\big((\boldsymbol{L}^\dagger)_{ii} + (\boldsymbol{L}^\dagger)_{jj} - 2(\boldsymbol{L}^\dagger)_{ij}\big)$
Neighbourhood power trace, Duda (2024)	$\sum_{i \in V} \mathrm{tr}\big((\boldsymbol{A}|_{N(i)})^p\big), \quad p \in \{4, 8\}$

Table S.4: **Graph indices**. These graph invariants are commonly used in molecular applications. For an edge $\{i, j\} \in E$, the number of vertices that are closer to i than to j is $n_i(\{i, j\}) \coloneqq |\{k \in V \mid d(k, i) < d(k, j)\}|$.

Invariant	**Definition**
Wiener index	$\frac{1}{2} \sum_{i \in V} \sum_{j \in V} d(i, j)$
Randić index	$\sum_{\{i,j\} \in E} \frac{1}{\sqrt{\deg(i) \deg(j)}}$
General Randić index	$\sum_{\{i,j\} \in E} \big(\deg(i) \deg(j)\big)^c, \quad c \in \mathbb{R}$
Atom bond connectivity	$\sum_{\{i,j\} \in E} \sqrt{\frac{\deg(i)+\deg(j)-2}{\deg(i) \deg(j)}}$
Geometric Arithmetic index	$\sum_{\{i,j\} \in E} 2 \frac{\sqrt{\deg(i) \deg(j)}}{\deg(i)+\deg(j)}$
Hyper-Wiener index	$\frac{1}{2} \sum_{i \in V} \sum_{j \in V} \big(d(i, j) + d(i, j)^2\big)$
Estrada index	$\sum_{\lambda_i \in \mathrm{spec}(\boldsymbol{A})} \exp(\lambda_i)$
First Zagreb index	$\sum_{i \in V} \deg(i)^2$
Second Zagreb index	$\sum_{\{i,j\} \in E} \deg(i) \deg(j)$
Schultz index	$\frac{1}{2} \sum_{i \in V} \sum_{j \in V} d(i, j) \big(\deg(i) + \deg(j)\big)$
Gutman index	$\frac{1}{2} \sum_{i \in V} \sum_{j \in V} d(i, j) \deg(i) \deg(j)$
Szeged index	$\sum_{\{i,j\} \in E} n_i(\{i, j\}) n_j(\{i, j\})$
Forgotten index	$\sum_{i \in V} \deg(i)^3$
Balaban index	$\frac{m}{\mathrm{rank}(G)+1} \sum_{\{i,j\} \in E} \big(\sum_{k \in V} d(i, k) \sum_{l \in V} d(j, l)\big)^{-1/2}$

A.2 METHODS

Graph neural networks. In this subsection, we briefly present graph neural networks. Graphs contain structural information that is often essential for machine learning tasks, and the central question is how to incorporate this structure into a learning system.

Modern graph neural networks (GNNs) commonly rely on the message-passing (MP) paradigm, which consists of two steps: *aggregation* and *update*. For simplicity, we discuss only vertex features; edge features can be handled similarly. The algorithm proceeds layer by layer. At layer $k = 0$, each vertex i is initialized with its feature vector,

$$\boldsymbol{h}_i^{(0)} \coloneqq \boldsymbol{v}_i.$$

For $k \geq 0$, the hidden representations of neighbouring vertices are aggregated into a message

$$\boldsymbol{m}_i^{(k+1)} \coloneqq \mathrm{aggregate}\left(\left[\boldsymbol{h}_j^{(k)} \mid j \in \mathcal{N}(i)\right]\right),$$

where $\mathrm{aggregate}$ is permutation-invariant and $[\cdot]$ denotes a multiset. The vertex representation is then updated by

$$\boldsymbol{h}_i^{(k+1)} \coloneqq \mathrm{update}\left(\boldsymbol{h}_i^{(k)}, \boldsymbol{m}_i^{(k+1)}\right).$$

These operations are applied in parallel to all vertices. Typically, aggregation uses the adjacency matrix $\boldsymbol{A}$ to collect information from neighbouring vertices, while the update step is implemented by learnable transformations and nonlinear activation functions.

In short, message passing iteratively incorporates local graph structure into vertex representations. However, MP-based GNNs are at most as expressive as the 1-*Weisfeiler–Lehman* test for graph isomorphism (Xu et al., 2019). Consequently, globally different graphs may be indistinguishable from the local perspective of such models, motivating the use of global inductive biases.

B DATASET REGIMES AND INVARIANT SETS

This appendix shows the datasets used in our experiments as well as the invariant regimes and sets based on computational constraints.

Table S.5: Statistics of investigated Graph Datasets. An asterisk * denotes splits were provided from an official source, otherwise splitting was done locally. Statistics were all computed locally. For datasets without node features, we introduced constant unit node features. For PCQM4Mv2 we are only using training and validation split data, since test data is not publicly available. Category denotes the application domain, or the task family for the synthetic benchmarks of Stoll et al. (2026).

Category	Name	Node Feat.	Edge Feat.	Targ.	Split Scheme	Task Type	Metric	# Graphs	Avg. $\lvert V\rvert$	Avg. $\lvert E\rvert$	Avg. Deg.
Small molecules	BACE Fey and Lenssen (2019); Wu et al. (2017)	Yes	Yes	1	Scaffold	Binary Class.	ROC-AUC	1,513	34.1	36.9	2.16
Small molecules	BBBP Fey and Lenssen (2019); Wu et al. (2017)	Yes	Yes	1	Scaffold	Binary Class.	ROC-AUC	2,039	24.1	26.0	2.16
Small molecules	HIV Hu et al. (2020)	Yes	Yes	1	Scaffold*	Binary Class.	ROC-AUC	41,127	25.5	27.5	2.15
Small molecules	ZINC (12k) Dwivedi et al. (2023); Fey and Lenssen (2019)	Yes	Yes	1	Random*	Regression	MAE	12,000	23.2	24.9	2.15
Small molecules	Lipo Fey and Lenssen (2019); Wu et al. (2017)	Yes	Yes	1	Scaffold	Regression	MSE	4,200	27.0	24.5	2.18
Small molecules	PCQM4Mv2 Fey and Lenssen (2019); Hu et al. (2020)	Yes	Yes	1	PubChem ID*	Regression	MAE	~3.6 million	14.1	14.5	2.06
Small molecules	PCBA Hu et al. (2020)	Yes	Yes	128	Scaffold*	Multi-label	AP	437,929	26.0	28.1	2.16
Small molecules	QM9 Fey and Lenssen (2019); Wu et al. (2017)	Yes	Yes	12	Random	Regression	MAE	130,831	18.0	18.7	2.07
Small molecules	NCI1 Fey and Lenssen (2019); Morris et al. (2020)	Yes	No	1	10-fold CV	Binary Class.	Accuracy	4,110	29.9	32.3	2.16
Small molecules	NCI109 Fey and Lenssen (2019); Morris et al. (2020)	Yes	No	1	10-fold CV	Binary Class.	Accuracy	4,127	29.7	32.1	2.17
Peptides	Peptides-func Dwivedi et al. (2022); Fey and Lenssen (2019)	Yes	Yes	10	Stratified*	Multi-label	AP	15,535	150.9	153.7	2.04
Peptides	Peptides-struct Dwivedi et al. (2022); Fey and Lenssen (2019)	Yes	Yes	11	Stratified*	Regression	MAE	15,535	150.9	153.7	2.04
Bioinformatics	PROTEINS Fey and Lenssen (2019); Morris et al. (2020)	Yes	No	1	10-fold CV	Binary Class.	Accuracy	1,113	39.1	72.8	3.73
Comp. vision	CIFAR10 Dwivedi et al. (2023); Fey and Lenssen (2019)	Yes	Yes	10	Stratified*	Multi-class	Accuracy	60,000	117.6	470.5	8.0
Social networks	COLLAB Fey and Lenssen (2019); Morris et al. (2020)	No	No	3	10-fold CV	Multi-class	Accuracy	5,000	74.5	2457.2	65.97
Social networks	IMDB-b Fey and Lenssen (2019); Morris et al. (2020)	No	No	1	10-fold CV	Binary Class.	Accuracy	1,000	19.8	96.5	9.76
Social networks	REDDIT-b Fey and Lenssen (2019); Morris et al. (2020)	No	No	1	10-fold CV	Binary Class.	Accuracy	2,000	429.6	497.8	2.32
El. circuits	Circuits_5_eff Stoll et al. (2026)	Yes	No	1	Random*	Regression	RSE	334,410	10.1	10.0	1.98
El. circuits	Circuits_5_vout Stoll et al. (2026)	Yes	No	1	Random*	Regression	RSE	334,419	10.1	10.0	1.98
El. circuits	Circuits_7_eff Stoll et al. (2026)	Yes	No	1	Random*	Regression	RSE	1,295	12.1	14.0	2.32
El. circuits	Circuits_10_eff Stoll et al. (2026)	Yes	No	1	Random*	Regression	RSE	4,630	16.0	20.0	2.51
El. circuits	Circuits_10_vout Stoll et al. (2026)	Yes	No	1	Random*	Regression	RSE	4,630	16.0	40.0	2.51
Comb. optimization	CO_BA_small Stoll et al. (2026)	No	No	1	Not found*	Binary Class.	MAE	50,000	249.9	495.8	3.97
Alg. reasoning	Flow_easy Stoll et al. (2026)	Yes	Yes	1	Not found*	Regression	MAE	1,010,000	16.5	40.2	4.88
Alg. reasoning	Flow_medium Stoll et al. (2026)	Yes	Yes	1	Not found*	Regression	MAE	1,010,000	16.5	24.2	2.93
Alg. reasoning	Flow_hard Stoll et al. (2026)	Yes	Yes	1	Not found*	Regression	MAE	1,010,000	16.5	24.2	2.93

Computation time. Table S.6 reports the mean wall-clock time per graph of each invariant on four datasets, ranging from small molecules to large pixel-grid graphs. These measurements are the basis for the regimes defined below.

Invariant	*Time per graph (ms)*				
	PCQM4Mv2	**BACE**	**Peptides-struct**	**CIFAR10**	**COLLAB**
Analytic torsion	0.40	0.84	4.4	129	—
Atom bond connectivity	0.15	0.37	1.6	7.5	404
Balaban index	0.17	0.40	7.1	9.8	14
Circuit rank	0.045	0.099	0.43	0.63	1.8
Commute time	0.35	0.59	4.2	4.4	12
Degree entropy	0.033	0.062	0.23	0.34	1.4
Density	0.022	0.049	0.21	0.31	1.4
Diameter	0.31	1.6	40	33	75
Eigenvalues of the normalized graph Laplacian	0.43	0.60	2.4	2.5	11
Estrada index	0.12	0.28	2.7	2.5	—
Forgotten index	0.032	0.071	0.31	0.40	1.5
Forman–Ricci curvature	0.13	0.32	1.3	12	1270
General Randić index	0.098	0.24	1.1	4.3	207
Geometric Arithmetic index	0.13	0.32	1.4	7.3	407
Geometric over arithmetic average of degree	0.20	0.28	0.64	0.89	3.2
Gutman index	0.14	0.32	6.9	9.4	9.5
Homomorphism counts ($\Omega^{\mathrm{con}}_{\leq 5}$)	76	82	83	687	—
Hyper-Wiener index	0.30	1.7	42	35	78
Magnitude	0.14	0.31	7.0	9.5	8.2
Neighbourhood power trace	0.84	3.1	52	55	1492
Number of edges	0.022	0.049	0.21	0.32	1.4
Natural logarithm of the number of spanning trees	0.35	0.48	1.3	2.1	9.0
Number of vertices	0.0019	0.0025	0.0055	0.0077	0.019
Ollivier–Ricci curvature	1.3	3.1	18	51	29604
Radius	0.30	1.6	40	33	76
Randić index	0.10	0.25	1.1	4.3	210
Schultz index	0.18	0.58	14	13	13
Szeged index	0.41	2.3	56	72	518
Transitivity	0.097	0.22	0.94	4.1	209
Wiener index	0.11	0.26	6.6	9.1	7.9
Zagreb indices	0.11	0.26	1.2	4.4	209
von Neumann entropy	0.37	0.49	2.1	2.0	8.3
Total	84	103	400	1206	34863

Table S.6: **Computation time of the invariants.** Mean wall-clock time per graph (ms) on a single CPU core. Means are taken over $5{,}000$ randomly sampled graphs for PCQM4Mv2, Peptides-struct and CIFAR10, and over all graphs for BACE and COLLAB. Entries marked "—" were not computable on this dataset.

Dataset regimes and invariant sets. We distinguish two regimes of datasets, summarized in Table S.7. In the *full regime*, comprising all categories of Table S.5 except Social networks and Comp. vision, we compute the complete set $\mathtt{I}$ of invariants introduced in Section A.1. The *efficient regime* comprises the remaining datasets COLLAB, IMDB-b, REDDIT-b, and CIFAR10. COLLAB, CIFAR10, and IMDB-b have the highest average node degree in our collection, which makes the most expensive invariants intractable; for REDDIT-b, we suspect that a small number of unusually complex graphs prevent a successful computation. For these datasets, we use the pruned set $\mathtt{I}_{\mathrm{eff}}$, which omits the prohibitively expensive or numerically unstable analytic torsion, homomorphism counts, and Estrada index. In both regimes, we additionally consider an *expressive* subset ($\mathtt{S}$ or $\mathtt{S}_{\mathrm{eff}}$), i.e., the smallest subset of invariants that retains full expressivity on BREC (Section 3.1). For all sets, the magnitude is evaluated at $q = e^{-0.42}$, selected by maximizing expressivity on BREC over a grid of candidate values.

Table S.7: **Invariant sets.** The full set `I` contains all invariants introduced in Section A.1, with its efficient-regime counterpart $\texttt{I}_{\text{eff}}$. The expressive subsets `S` and $\texttt{S}_{\text{eff}}$ contain a small number of invariants identified as highly expressive on BREC (Section 3.1). Datasets are given by their category in Table S.5; each category comprises all of its datasets.

Regime	Datasets	Set	Invariants
Full	Small molecules, Peptides, Bioinformatics, El. circuits, Comb. optimization, Alg. reasoning	`I`	All invariants from the invariants section
		`S`	Analytic torsion, mean commute time, magnitude ($q = e^{-0.42}$), graph radius
Efficient	Social networks, Comp. vision	$\texttt{I}_{\text{eff}}$	All invariants from the invariants section, except: analytic torsion, homomorphism counts, and the Estrada index
		$\texttt{S}_{\text{eff}}$	Algebraic connectivity (second-smallest Laplacian eigenvalue), mean Ollivier–Ricci curvature, magnitude ($q = e^{-0.42}$), closed-neighbourhood power trace with $p = 8$, graph radius

Redundancy between invariants. Figure S.1 shows the pairwise distance correlation between invariants, computed over the datasets of the full regime. Distance correlation measures the dependence between two random vectors and vanishes if and only if they are independent (Székely et al., 2007). It is thus preferable to the Pearson correlation in our setting, since invariants may depend on each other in a non-linear manner, which a purely linear measure would not detect. We compute it per dataset with the `dcor` package (Ramos-Carreño and Torrecilla, 2023) and report the mean and the minimum across datasets. While some invariants are strongly correlated, those of the expressive subset `S` are not, indicating that they capture complementary structural information.

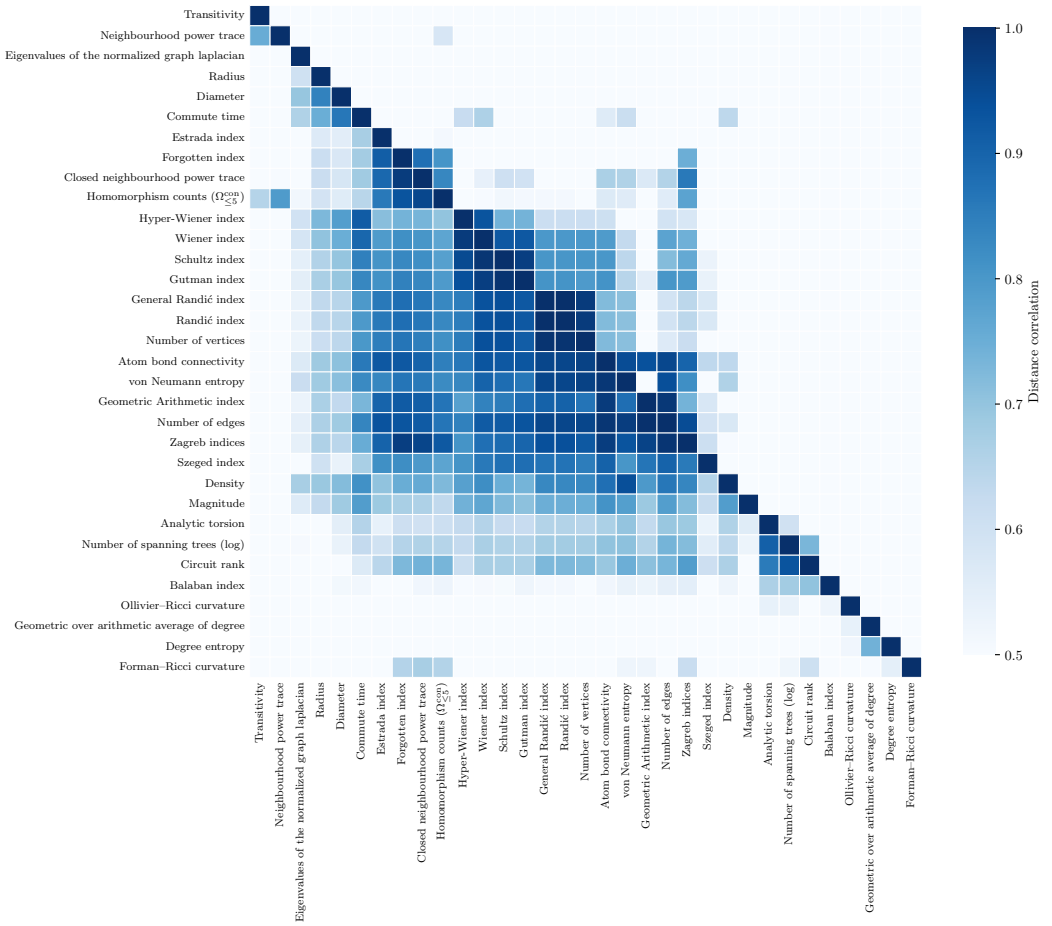


Figure S.1: Distance correlation over the full regime. The lower triangle shows the mean, while the upper triangle shows the minimum of the distance correlation over the full regime.

C HYPERPARAMETERS

This appendix details the hyperparameters used for the XGBoost classifiers in our heterogeneity experiments (Section 3.2) and downstream experiments (Section 3.3), and for the shared GIN graph-encoder used in the multi-task experiments in Section E. In terms of hardware, our training used 6 NVIDIA RTX 3080 and 4 NVIDIA L40s GPUs.

XGBoost. We use XGBoost Chen and Guestrin (2016) with histogram-based tree construction. For the downstream experiments (Section 3.3), we train on GPU; the base configuration is given in Table S.8, and the hyperparameter search space explored by the Optuna Akiba et al. (2019) Bayesian sampler is listed in Table S.9. The number of boosting rounds is not tuned directly; instead, training stops early once the validation metric has not improved for 8 rounds, and evaluation uses the best number of rounds. We run 150 tuning iterations.

Table S.8: **Architectural XGBoost parameters** used in the downstream experiments.

Hyperparameter	Value
`device`	`cuda`
`tree_method`	`hist`
`grow_policy`	`lossguide`
Early-stopping patience	8

Table S.9: **XGBoost hyperparameter search space** explored by the Optuna sampler.

Hyperparameter	Range	Sampling
`learning_rate`	$[5\times10^{-3},\ 1.0]$	log-uniform
`max_leaves`	$[5,\ 1000]$	log-uniform (integer)
`min_child_weight`	$[10^{-3},\ 200]$	log-uniform
`subsample`	$[0.1,\ 1.0]$	uniform
`colsample_bytree`	$[0.1,\ 1.0]$	uniform
`reg_lambda`	$[10^{-5},\ 100]$	log-uniform
`reg_alpha`	$[10^{-5},\ 100]$	log-uniform

XGBoost for meta-classification. For the meta-classification experiments (Section 3.2), we do not tune hyperparameters but use the fixed configuration in Table S.10 for all runs, training with a multi-class cross-entropy objective for 300 boosting rounds.

Table S.10: **XGBoost hyperparameters for meta-classification**, kept fixed across all meta-classification experiments.

Hyperparameter	Value
`tree_method`	`hist`
`grow_policy`	`depthwise`
`learning_rate`	0.1
`max_depth`	10
`min_child_weight`	0.1
`subsample`	0.8
`colsample_bytree`	0.8
`reg_lambda`	1.0
`num_boost_round`	300

GIN graph-encoder. For the multi-task learning experiments, we use a shared GIN (Xu et al., 2019) graph-encoder. Its hyperparameters are held fixed across all dataset pairs, except for the output layer size, which depends on the task head. We optimize with AdamW, using PyTorch defaults except for the learning rate and weight decay, which are set as in Table S.11. Training is stopped early once the validation loss on both datasets has not improved for 100 epochs.

Table S.11: **GIN graph-encoder hyperparameters** used in all multi-task experiments. The output dimension of the task head is the only hyperparameter that varies across experiments and is determined by the target dimension of the corresponding dataset.

Hyperparameter	Value
Number of layers	4
Hidden dimension	128
Normalization type	Batch
Dropout	0.2
Pooling	sum
Task-head hidden layers	$[256,\ 128,\ 128]$
Optimizer	AdamW (PyTorch defaults)
Learning rate	10^{-3}
Weight decay	10^{-4}
Batch size	256
Early-stopping patience	100

D ADDITIONAL RESULTS

In this appendix, we collect extended results, complementing each of the three experiments in the main text.

D.1 EXPRESSIVITY

Per-invariant expressivity heatmaps for the full and efficient regimes (Figures S.2 and S.3), complementing Section 3.1.

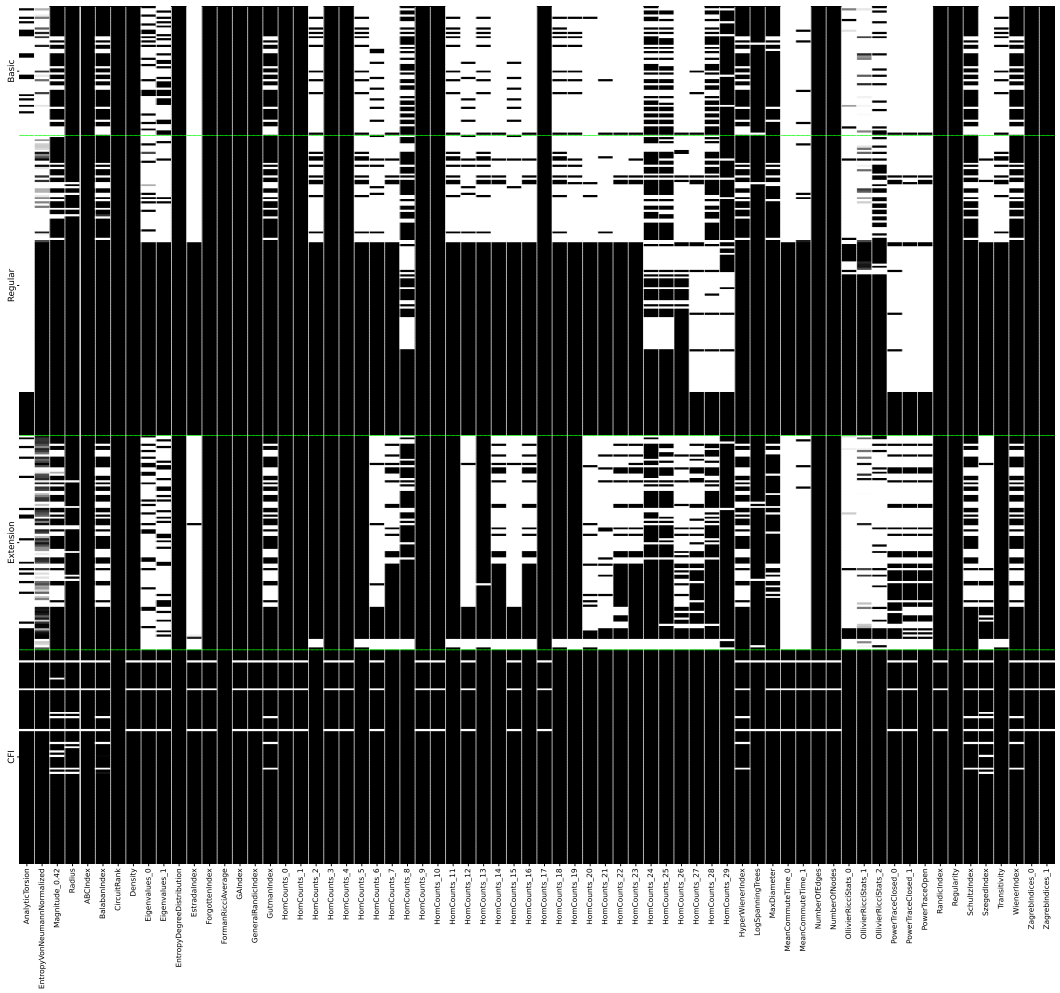


Figure S.2: Heatmap that shows BREC graph pairs on the x-axis and invariants on the y-axis. The colour of a cell indicates the difference of an invariant value when computed for both graphs. Brighter cells indicate larger differences. The first four rows form an expressive subset.

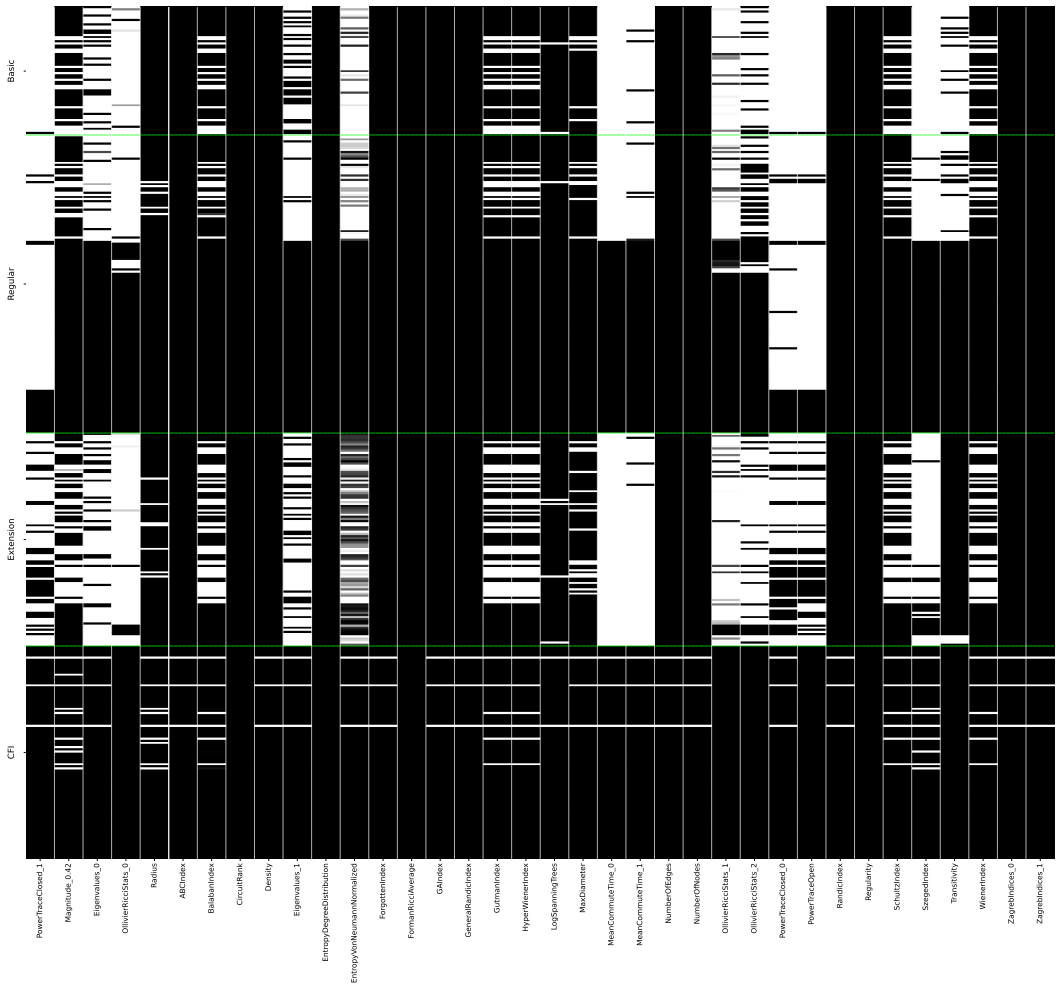


Figure S.3: Heatmap that shows BREC graph pairs on the x-axis and invariants on the y-axis. The colour of a cell indicates the difference of an invariant value when computed for both graphs. Brighter cells indicate larger differences. The first four rows form an expressive subset.

D.2 META-CLASSIFICATION

Meta-classification accuracies and confusion matrices, complementing Section 3.2.

Matched pairs. For the pairwise confounder-free meta-classification on MoleculeNet, Table S.12 lists the number of graphs that can be matched exactly on the confounders, i.e., the number of nodes, the number of edges, and the second degree moment. Pairs with fewer than 400 matched graphs (BACE–QM9, BBBP–QM9, and Lipo–QM9) are omitted from Table 2.

Table S.12: **Exactly matched graphs per dataset pair.** Graphs available per dataset in a cohort matched on the confounders (number of nodes, number of edges, and second degree moment); the diagonal gives the dataset size.

	BACE	BBBP	HIV	Lipo	PCBA	QM9
BACE	*1,513*	400	1,317	800	1,427	5
BBBP	400	*2,039*	1,910	1,284	1,997	209
HIV	1,317	1,910	*41,127*	4,111	37,375	2,406
Lipo	800	1,284	4,111	*4,200*	4,188	71
PCBA	1,427	1,997	37,375	4,188	*437,929*	4,018
QM9	5	209	2,406	71	4,018	*130,831*

Diverse subset. Besides classifying all datasets, we restrict meta-classification to a *diverse subset* comprising one dataset per category of Table S.5: BACE, Peptides-func, PROTEINS, CIFAR10, COLLAB, Circuits_5_eff, CO_BA_small, and Flow_easy. This selection preserves the structural diversity of our collection in a more compact form; the corresponding confusion matrix is shown in Figure S.5.

Table S.13: **Meta-classification results.** Test accuracy, mean $\pm$ standard deviation. Cells show meta-classification results for combinations of datasets and invariant sets with a random sampling of 1,000 graphs per dataset over 5 seeds. *Diverse* uses one dataset per category (eight datasets).

Datasets	$\mathtt{I}_{\text{eff}}$	$\mathtt{S}_{\text{eff}}$	Confounders
All datasets	$0.612_{\pm 0.006}$	$0.585_{\pm 0.005}$	$0.559_{\pm 0.004}$
Diverse	$0.996_{\pm 0.002}$	$0.993_{\pm 0.002}$	$0.989_{\pm 0.003}$

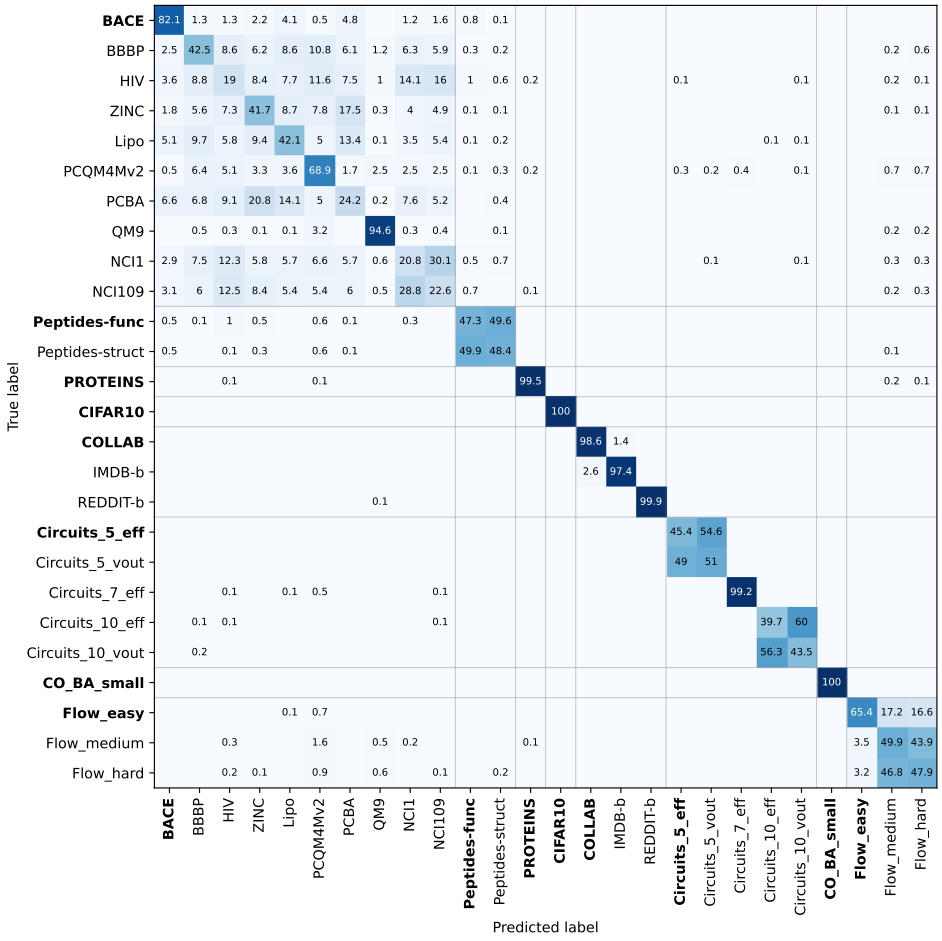


Figure S.4: **Confusion matrix, all datasets, $\mathtt{I}_{\text{eff}}$.** The datasets are shown in Table S.5 and the invariant set $\mathtt{I}_{\text{eff}}$ in Section B. The cells show the percentage of each dataset's test graphs assigned to each predicted dataset, aggregated over 5 seeds; empty cells indicate that no graph was assigned. Thin lines separate the dataset categories, and datasets of the diverse subset (Figure S.5) are shown in bold.

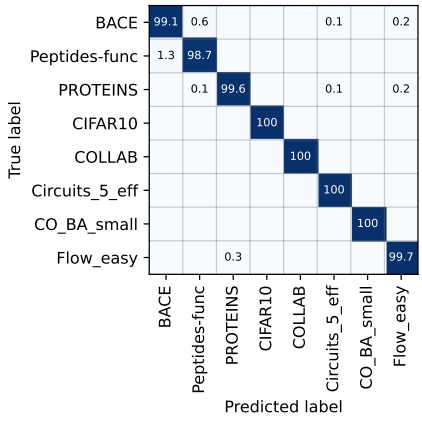


Figure S.5: **Confusion matrix, diverse subset, $\mathtt{I}_{\text{eff}}$.** The diverse subset is shown in Section D.2 and the invariant set $\mathtt{I}_{\text{eff}}$ in Section B. The cells show the percentage of each dataset's test graphs assigned to each predicted dataset, aggregated over 5 seeds; empty cells indicate that no graph was assigned.

D.3 BENCHMARKS

Full benchmark tables and domain comparisons referenced in Section 3.3. Results reported in the literature for the TU, MoleculeNet, LRGD, and Flow datasets are listed in Tables S.18 to S.21.

Table S.14: XGBoost trained across 26 datasets and eight input configurations. Each cell shows performance after tuning hyperparameters for 150 iterations with a Bayesian optimization sampler. Official splits and evaluation procedures were used whenever available. Test metric (%, mean ± std) reported over 5 seeds or 10 folds for nested cross validation. In each row, the best mean is shown in bold and the second best is underlined, respecting the direction of each metric; tied means share the same marking. For datasets of the efficient regime (COLLAB, CIFAR10, IMDB-b, REDDIT-b), the invariant sets `I` and `S` refer to $\texttt{I}_{\text{eff}}$ and $\texttt{S}_{\text{eff}}$ (Table S.7).

Metric	Dataset	agg + I	agg + S	I	agg	sum	S	sum + I	sum + S
AP	PCBA	**0.1465 ± 0.0016**	0.1459 ± 0.0018	0.0704 ± 0.0003	0.1452 ± 0.0005	0.1102 ± 0.0007	0.0554 ± 0.0009	0.1226 ± 0.0003	0.1140 ± 0.0018
	Peptides-func	**0.6874 ± 0.0022**	0.6822 ± 0.0010	0.5836 ± 0.0014	0.6838 ± 0.0043	0.6637 ± 0.0019	0.5120 ± 0.0025	0.6765 ± 0.0021	0.6678 ± 0.0015
MAE	PCQM4Mv2	0.1520 ± 0.0000	0.1587 ± 0.0002	0.7397 ± 0.0002	**0.1516 ± 0.0002**	0.2591 ± 0.0001	0.7641 ± 0.0003	0.2099 ± 0.0002	0.2116 ± 0.0002
	QM9	**4.0590 ± 0.0283**	4.4554 ± 0.0156	90.8535 ± 0.0757	6.1227 ± 0.0170	9.0812 ± 0.0151	151.1581 ± 0.0548	4.4817 ± 0.0182	4.9699 ± 0.0187
	ZINC	**0.4450 ± 0.0032**	0.5162 ± 0.0013	1.3188 ± 0.0018	0.5251 ± 0.0012	0.6177 ± 0.0021	1.3643 ± 0.0015	0.4996 ± 0.0044	0.5700 ± 0.0021
	CO_BA_small	**1.5531 ± 0.0012**	2.4824 ± 0.0012	1.5542 ± 0.0007	2.4559 ± 0.0019	2.9617 ± 0.0009	2.7281 ± 0.0010	1.5535 ± 0.0012	2.7265 ± 0.0006
	Flow_easy	1.8328 ± 0.0062	1.8597 ± 0.0307	3.7735 ± 0.0186	**1.7674 ± 0.0039**	3.6538 ± 0.0078	3.7452 ± 0.0050	3.6133 ± 0.0000	3.6228 ± 0.0163
	Flow_hard	5.2603 ± 0.0305	5.3428 ± 0.0000	9.2036 ± 0.1258	**5.2001 ± 0.0287**	8.2483 ± 0.0000	10.3463 ± 0.0495	8.9556 ± 0.1325	9.6316 ± 0.0471
	Flow_medium	5.2954 ± 0.1131	5.3698 ± 0.0417	9.4187 ± 0.3461	**5.1565 ± 0.0532**	8.0768 ± 0.0563	10.7584 ± 0.0374	9.2379 ± 0.2110	9.8586 ± 0.1685
	Peptides-struct	**0.2476 ± 0.0002**	0.2489 ± 0.0001	0.2576 ± 0.0003	0.2518 ± 0.0002	0.2558 ± 0.0003	0.2627 ± 0.0001	0.2479 ± 0.0003	0.2498 ± 0.0002
MSE	Lipo	**0.6929 ± 0.0009**	0.7251 ± 0.0031	1.0200 ± 0.0120	0.7324 ± 0.0070	0.8118 ± 0.0021	1.0206 ± 0.0108	0.7136 ± 0.0041	0.7514 ± 0.0055
RSE	Circuits_10_eff	0.7539 ± 0.0010	**0.7486 ± 0.0003**	0.9474 ± 0.0031	0.7627 ± 0.0000	0.8725 ± 0.0021	0.9505 ± 0.0005	0.8511 ± 0.0004	0.8484 ± 0.0030
	Circuits_10_vout	0.8456 ± 0.0030	**0.8371 ± 0.0011**	0.9667 ± 0.0003	0.8455 ± 0.0024	0.9872 ± 0.0041	0.9738 ± 0.0009	0.9442 ± 0.0011	0.9492 ± 0.0005
	Circuits_5_eff	**0.1965 ± 0.0001**	**0.1965 ± 0.0000**	0.9267 ± 0.0000	0.1986 ± 0.0000	0.7394 ± 0.0001	0.9267 ± 0.0000	0.6892 ± 0.0000	0.6892 ± 0.0001
	Circuits_5_vout	**0.2340 ± 0.0000**	**0.2340 ± 0.0000**	0.9286 ± 0.0000	0.2356 ± 0.0000	0.7495 ± 0.0000	0.9286 ± 0.0000	0.7028 ± 0.0000	0.7029 ± 0.0000
	Circuits_7_eff	0.8811 ± 0.0094	0.8696 ± 0.0054	1.0157 ± 0.0041	**0.8690 ± 0.0023**	0.9428 ± 0.0022	1.0160 ± 0.0086	0.9600 ± 0.0058	0.9538 ± 0.0004
Accuracy	CIFAR10	0.3009 ± 0.0021	0.2873 ± 0.0010	0.1842 ± 0.0018	0.2854 ± 0.0010	0.2845 ± 0.0014	0.1524 ± 0.0026	**0.3022 ± 0.0019**	0.2934 ± 0.0018
	COLLAB	0.7992 ± 0.0122	0.7703 ± 0.0107	0.7977 ± 0.0128	0.7474 ± 0.0113	0.5866 ± 0.0164	0.7493 ± 0.0156	**0.8011 ± 0.0124**	0.7571 ± 0.0172
	IMDB-b	0.7109 ± 0.0349	0.7020 ± 0.0275	0.7030 ± 0.0352	0.6883 ± 0.0366	0.5610 ± 0.0483	0.6961 ± 0.0376	**0.7150 ± 0.0394**	0.7075 ± 0.0406
	NCI1	**0.7996 ± 0.0237**	0.7924 ± 0.0231	0.7530 ± 0.0274	0.7852 ± 0.0221	0.6968 ± 0.0249	0.7085 ± 0.0219	0.7869 ± 0.0268	0.7632 ± 0.0166
	NCI109	**0.7967 ± 0.0147**	0.7857 ± 0.0098	0.7400 ± 0.0146	0.7740 ± 0.0128	0.7074 ± 0.0132	0.6935 ± 0.0207	0.7765 ± 0.0152	0.7647 ± 0.0114
	PROTEINS	0.7728 ± 0.0389	0.7676 ± 0.0363	0.7415 ± 0.0361	0.7665 ± 0.0296	0.7661 ± 0.0412	0.7357 ± 0.0403	0.7714 ± 0.0399	**0.7753 ± 0.0343**
	REDDIT-b	**0.9049 ± 0.0198**	0.9015 ± 0.0162	0.9031 ± 0.0209	0.8812 ± 0.0201	0.7775 ± 0.0190	0.8567 ± 0.0223	**0.9049 ± 0.0174**	0.8843 ± 0.0192
ROC-AUC	BACE	0.8239 ± 0.0050	0.8363 ± 0.0048	**0.8364 ± 0.0047**	0.8278 ± 0.0091	0.8263 ± 0.0084	0.8266 ± 0.0054	0.8264 ± 0.0041	0.8330 ± 0.0090
	BBBP	0.7408 ± 0.0000	**0.7439 ± 0.0129**	0.6562 ± 0.0048	0.7404 ± 0.0027	0.7181 ± 0.0042	0.6770 ± 0.0035	0.7329 ± 0.0000	0.7191 ± 0.0055
	HIV	**0.7695 ± 0.0032**	0.7626 ± 0.0032	0.7147 ± 0.0057	0.7438 ± 0.0087	0.7402 ± 0.0097	0.7366 ± 0.0062	0.7648 ± 0.0000	0.7661 ± 0.0053

Table S.15: XGBoost trained across targets on QM9 and eight input configurations. Each cell shows performance after tuning hyperparameters for 150 iterations with a bayesian optimization sampler. For each target, a separate model was fit. Test mean absolute error (mean ± std) reported over 5 seeds. In each row, the lowest mean is shown in bold and the second lowest is underlined.

Target	agg + I	agg + S	I	agg	sum	S	sum + I	sum + S
μ, D	**0.5846 ± 0.0012**	0.5961 ± 0.0012	0.7940 ± 0.0001	0.6047 ± 0.0009	0.7634 ± 0.0006	0.8919 ± 0.0003	0.6509 ± 0.0002	0.6948 ± 0.0000
α, a_0^3	**0.4803 ± 0.0007**	0.5405 ± 0.0009	1.4409 ± 0.0011	0.6569 ± 0.0021	1.0894 ± 0.0001	2.1792 ± 0.0011	0.5895 ± 0.0010	0.7512 ± 0.0007
ϵ_{HOMO}, eV	**0.1200 ± 0.0001**	0.1319 ± 0.0004	0.2006 ± 0.0001	0.1413 ± 0.0004	0.2186 ± 0.0001	0.2636 ± 0.0001	0.1473 ± 0.0003	0.1808 ± 0.0001
ϵ_{LUMO}, eV	**0.1456 ± 0.0004**	0.1501 ± 0.0002	0.3146 ± 0.0007	0.1586 ± 0.0004	0.2895 ± 0.0001	0.4563 ± 0.0000	0.1895 ± 0.0001	0.2257 ± 0.0004
Δ_ϵ, eV	**0.1798 ± 0.0002**	0.1946 ± 0.0005	0.3146 ± 0.0003	0.2085 ± 0.0006	0.3626 ± 0.0002	0.4806 ± 0.0005	0.2317 ± 0.0004	0.2856 ± 0.0002
$\langle R^2 \rangle, a_0^2$	**35.2234 ± 0.1031**	40.0148 ± 0.0000	66.4269 ± 0.0501	60.6087 ± 0.0480	102.1372 ± 0.0090	78.9205 ± 0.0689	42.8080 ± 0.0218	52.5202 ± 0.0714
ZPVE, eV	**0.0065 ± 0.0000**	0.0071 ± 0.0000	0.0315 ± 0.0000	0.0110 ± 0.0001	0.0225 ± 0.0000	0.0404 ± 0.0001	0.0079 ± 0.0000	0.0100 ± 0.0000
U_0, eV	2.8585 ± 0.0999	2.6446 ± 0.1476	255.2492 ± 0.2968	2.6059 ± 0.0917	**0.7626 ± 0.0000**	431.6066 ± 0.0000	2.1957 ± 0.1375	1.1485 ± 0.0000
U, eV	2.9647 ± 0.0000	2.8989 ± 0.2647	254.1861 ± 0.2659	2.5184 ± 0.0518	**0.7841 ± 0.0315**	432.0574 ± 0.4709	2.2673 ± 0.0000	1.3797 ± 0.1004
H, eV	2.9476 ± 0.1286	2.9254 ± 0.2682	256.4001 ± 0.0000	2.5142 ± 0.1329	**0.8186 ± 0.0935**	432.9514 ± 0.5786	2.2147 ± 0.1574	1.0580 ± 0.1373
G, eV	3.0204 ± 0.2566	3.1680 ± 0.0816	254.3722 ± 0.3803	3.1520 ± 0.0000	**0.9200 ± 0.1166**	433.3830 ± 0.2723	2.2124 ± 0.0750	1.0408 ± 0.0917
$c_v, \frac{cal}{molK}$	**0.1762 ± 0.0004**	0.1930 ± 0.0004	0.5107 ± 0.0007	0.2925 ± 0.0004	0.8059 ± 0.0002	0.6658 ± 0.0007	0.2659 ± 0.0004	0.3438 ± 0.0008

Table S.16: 95% percentile bootstrap confidence intervals (10,000 resamples) for the results in Table S.14. Each cell shows the mean test metric with its confidence interval in brackets, computed over 5 seeds or 10 folds for nested cross validation. Note that cross-validation folds share training data and are therefore not independent. Bootstrap bias is *negligible* in all cells (max $|\text{bias}|/\text{SE} < 0.03$), where SE denotes the bootstrap standard error. Intervals with equal bounds are narrower than the displayed precision.

Metric	Dataset	agg + I	agg + S	I	agg	sum	S	sum + I	sum + S
AP	PCBA	0.147 [0.145, 0.148]	0.146 [0.144, 0.147]	0.070 [0.070, 0.071]	0.145 [0.145, 0.146]	0.110 [0.110, 0.111]	0.055 [0.055, 0.056]	0.123 [0.122, 0.123]	0.114 [0.113, 0.115]
	Peptides-func	0.687 [0.686, 0.689]	0.682 [0.681, 0.683]	0.584 [0.582, 0.585]	0.684 [0.681, 0.687]	0.664 [0.662, 0.665]	0.512 [0.510, 0.514]	0.676 [0.675, 0.678]	0.668 [0.667, 0.669]
MAE	PCQM4Mv2	0.152 [0.152, 0.152]	0.159 [0.159, 0.159]	0.740 [0.740, 0.740]	0.152 [0.151, 0.152]	0.259 [0.259, 0.259]	0.764 [0.764, 0.764]	0.210 [0.210, 0.210]	0.212 [0.211, 0.212]
	QM9	4.059 [4.038, 4.081]	4.455 [4.441, 4.466]	90.853 [90.799, 90.914]	6.123 [6.110, 6.135]	9.081 [9.068, 9.091]	151.158 [151.111, 151.198]	4.482 [4.466, 4.494]	4.970 [4.954, 4.984]
	ZINC	0.445 [0.442, 0.447]	0.516 [0.515, 0.517]	1.319 [1.317, 1.320]	0.525 [0.524, 0.526]	0.618 [0.616, 0.620]	1.364 [1.363, 1.365]	0.500 [0.496, 0.504]	0.570 [0.568, 0.572]
	CO_BA_small	1.553 [1.552, 1.554]	2.482 [2.481, 2.484]	1.554 [1.554, 1.555]	2.456 [2.454, 2.458]	2.962 [2.961, 2.962]	2.728 [2.727, 2.729]	1.553 [1.552, 1.554]	2.726 [2.726, 2.727]
	Flow_easy	1.833 [1.829, 1.839]	1.860 [1.837, 1.887]	3.773 [3.758, 3.789]	1.767 [1.764, 1.770]	3.654 [3.648, 3.660]	3.745 [3.741, 3.749]	3.613 [3.613, 3.613]	3.623 [3.609, 3.635]
	Flow_hard	5.260 [5.237, 5.288]	5.343 [5.343, 5.343]	9.204 [9.107, 9.300]	5.200 [5.179, 5.225]	8.248 [8.248, 8.248]	10.346 [10.304, 10.384]	8.956 [8.843, 9.062]	9.632 [9.593, 9.670]
	Flow_medium	5.295 [5.199, 5.383]	5.370 [5.337, 5.406]	9.419 [9.115, 9.717]	5.157 [5.104, 5.190]	8.077 [8.035, 8.123]	10.758 [10.730, 10.790]	9.238 [9.040, 9.379]	9.859 [9.732, 10.005]
	Peptides-struct	0.248 [0.247, 0.248]	0.249 [0.249, 0.249]	0.258 [0.257, 0.258]	0.252 [0.252, 0.252]	0.256 [0.256, 0.256]	0.263 [0.263, 0.263]	0.248 [0.248, 0.248]	0.250 [0.250, 0.250]
MSE	Lipo	0.693 [0.692, 0.693]	0.725 [0.722, 0.727]	1.020 [1.010, 1.030]	0.732 [0.727, 0.738]	0.812 [0.810, 0.814]	1.021 [1.010, 1.027]	0.714 [0.711, 0.717]	0.751 [0.747, 0.755]
RSE	Circuits_10_eff	0.754 [0.753, 0.755]	0.749 [0.748, 0.749]	0.947 [0.945, 0.950]	0.763 [0.763, 0.763]	0.873 [0.871, 0.874]	0.951 [0.950, 0.951]	0.851 [0.851, 0.851]	0.848 [0.845, 0.850]
	Circuits_10_vout	0.846 [0.843, 0.848]	0.837 [0.836, 0.838]	0.967 [0.966, 0.967]	0.846 [0.843, 0.847]	0.987 [0.984, 0.991]	0.974 [0.973, 0.975]	0.944 [0.943, 0.945]	0.949 [0.949, 0.950]
	Circuits_5_eff	0.196 [0.196, 0.197]	0.196 [0.196, 0.196]	0.927 [0.927, 0.927]	0.199 [0.199, 0.199]	0.739 [0.739, 0.739]	0.927 [0.927, 0.927]	0.689 [0.689, 0.689]	0.689 [0.689, 0.689]
	Circuits_5_vout	0.234 [0.234, 0.234]	0.234 [0.234, 0.234]	0.929 [0.929, 0.929]	0.236 [0.236, 0.236]	0.750 [0.749, 0.750]	0.929 [0.929, 0.929]	0.703 [0.703, 0.703]	0.703 [0.703, 0.703]
	Circuits_7_eff	0.881 [0.873, 0.889]	0.870 [0.865, 0.874]	1.016 [1.012, 1.019]	0.869 [0.867, 0.871]	0.943 [0.941, 0.944]	1.016 [1.009, 1.022]	0.960 [0.956, 0.965]	0.954 [0.953, 0.954]
Accuracy	CIFAR10	0.301 [0.299, 0.302]	0.287 [0.287, 0.288]	0.184 [0.184, 0.185]	0.285 [0.285, 0.286]	0.285 [0.284, 0.285]	0.152 [0.152, 0.153]	0.302 [0.301, 0.304]	0.293 [0.293, 0.294]
	COLLAB	0.799 [0.792, 0.806]	0.770 [0.765, 0.775]	0.798 [0.790, 0.805]	0.747 [0.742, 0.753]	0.587 [0.577, 0.595]	0.749 [0.745, 0.754]	0.801 [0.795, 0.809]	0.757 [0.751, 0.763]
	IMDB-b	0.711 [0.690, 0.730]	0.702 [0.689, 0.714]	0.703 [0.682, 0.721]	0.688 [0.667, 0.707]	0.561 [0.535, 0.591]	0.696 [0.676, 0.716]	0.715 [0.693, 0.738]	0.707 [0.687, 0.730]
	NCI1	0.800 [0.786, 0.814]	0.792 [0.779, 0.806]	0.753 [0.736, 0.768]	0.785 [0.773, 0.798]	0.697 [0.682, 0.712]	0.709 [0.696, 0.722]	0.787 [0.772, 0.803]	0.763 [0.754, 0.774]
	NCI109	0.797 [0.788, 0.805]	0.786 [0.781, 0.791]	0.740 [0.732, 0.749]	0.774 [0.767, 0.780]	0.707 [0.700, 0.715]	0.693 [0.681, 0.705]	0.776 [0.768, 0.785]	0.765 [0.759, 0.771]
	PROTEINS	0.773 [0.750, 0.796]	0.768 [0.748, 0.789]	0.741 [0.720, 0.763]	0.766 [0.750, 0.783]	0.766 [0.744, 0.790]	0.736 [0.713, 0.760]	0.771 [0.748, 0.795]	0.775 [0.756, 0.795]
	REDDIT-b	0.905 [0.894, 0.917]	0.902 [0.893, 0.911]	0.903 [0.891, 0.915]	0.881 [0.869, 0.892]	0.778 [0.766, 0.789]	0.857 [0.846, 0.868]	0.905 [0.895, 0.915]	0.884 [0.872, 0.895]
ROC-AUC	BACE	0.824 [0.820, 0.828]	0.836 [0.833, 0.840]	0.836 [0.833, 0.840]	0.828 [0.821, 0.836]	0.826 [0.820, 0.833]	0.827 [0.822, 0.831]	0.826 [0.823, 0.830]	0.833 [0.826, 0.841]
	BBBP	0.741 [0.741, 0.741]	0.744 [0.733, 0.755]	0.656 [0.653, 0.660]	0.740 [0.738, 0.743]	0.718 [0.715, 0.722]	0.677 [0.674, 0.680]	0.733 [0.733, 0.733]	0.719 [0.715, 0.724]
	HIV	0.769 [0.767, 0.772]	0.763 [0.760, 0.765]	0.715 [0.710, 0.719]	0.744 [0.736, 0.752]	0.740 [0.731, 0.748]	0.737 [0.731, 0.742]	0.765 [0.765, 0.765]	0.766 [0.761, 0.770]

Table S.17: 95% percentile bootstrap confidence intervals (10,000 resamples) for the results in Table S.15. Each cell shows the test mean absolute error with its confidence interval in brackets, computed over 5 seeds. Bootstrap bias is negligible in all cells (max $|\text{bias}|/\text{SE} < 0.03$), where SE denotes the bootstrap standard error. Intervals with equal bounds are narrower than the displayed precision.

Target	agg + I	agg + S	I	agg	sum	S	sum + I	sum + S
μ, D	0.585 [0.584, 0.586]	0.596 [0.595, 0.597]	0.794 [0.794, 0.794]	0.605 [0.604, 0.606]	0.763 [0.763, 0.764]	0.892 [0.892, 0.892]	0.651 [0.651, 0.651]	0.695 [0.695, 0.695]
α, a_0^3	0.480 [0.480, 0.481]	0.540 [0.540, 0.541]	1.441 [1.440, 1.442]	0.657 [0.655, 0.659]	1.089 [1.089, 1.090]	2.179 [2.178, 2.180]	0.590 [0.589, 0.590]	0.751 [0.751, 0.752]
ϵ_{HOMO}, eV	0.120 [0.120, 0.120]	0.132 [0.132, 0.132]	0.201 [0.201, 0.201]	0.141 [0.141, 0.142]	0.219 [0.219, 0.219]	0.264 [0.264, 0.264]	0.147 [0.147, 0.147]	0.181 [0.181, 0.181]
ϵ_{LUMO}, eV	0.146 [0.145, 0.146]	0.150 [0.150, 0.150]	0.315 [0.314, 0.315]	0.159 [0.158, 0.159]	0.289 [0.289, 0.290]	0.456 [0.456, 0.456]	0.190 [0.189, 0.190]	0.226 [0.225, 0.226]
Δ_ϵ, eV	0.180 [0.180, 0.180]	0.195 [0.194, 0.195]	0.315 [0.314, 0.315]	0.208 [0.208, 0.209]	0.363 [0.362, 0.363]	0.481 [0.480, 0.481]	0.232 [0.231, 0.232]	0.286 [0.285, 0.286]
$\langle R^2 \rangle, a_0^2$	35.223 [35.127, 35.302]	40.015 [40.015, 40.015]	66.427 [66.381, 66.462]	60.609 [60.564, 60.645]	102.137 [102.129, 102.144]	78.920 [78.857, 78.964]	42.808 [42.791, 42.828]	52.520 [52.459, 52.578]
ZPVE, eV	0.006 [0.006, 0.006]	0.007 [0.007, 0.007]	0.031 [0.031, 0.032]	0.011 [0.011, 0.011]	0.022 [0.022, 0.022]	0.040 [0.040, 0.040]	0.008 [0.008, 0.008]	0.010 [0.010, 0.010]
U_0, eV	2.858 [2.775, 2.937]	2.645 [2.514, 2.752]	255.249 [255.004, 255.483]	2.606 [2.527, 2.674]	0.763 [0.763, 0.763]	431.607 [431.607, 431.607]	2.196 [2.081, 2.310]	1.149 [1.149, 1.149]
U, eV	2.965 [2.965, 2.965]	2.899 [2.679, 3.119]	254.186 [253.993, 254.427]	2.518 [2.470, 2.555]	0.784 [0.756, 0.809]	432.057 [431.711, 432.472]	2.267 [2.267, 2.267]	1.380 [1.285, 1.442]
H, eV	2.948 [2.839, 3.046]	2.925 [2.681, 3.133]	256.400 [256.400, 256.400]	2.514 [2.400, 2.619]	0.819 [0.742, 0.896]	432.951 [432.440, 433.400]	2.215 [2.066, 2.320]	1.058 [0.949, 1.167]
G, eV	3.020 [2.838, 3.259]	3.168 [3.101, 3.235]	254.372 [254.074, 254.714]	3.152 [3.152, 3.152]	0.920 [0.830, 1.028]	433.383 [433.134, 433.605]	2.212 [2.145, 2.274]	1.041 [0.975, 1.126]
c_v, $\frac{cal}{molK}$	0.176 [0.176, 0.177]	0.193 [0.193, 0.193]	0.511 [0.510, 0.511]	0.292 [0.292, 0.293]	0.806 [0.806, 0.806]	0.666 [0.665, 0.666]	0.266 [0.266, 0.266]	0.344 [0.343, 0.344]

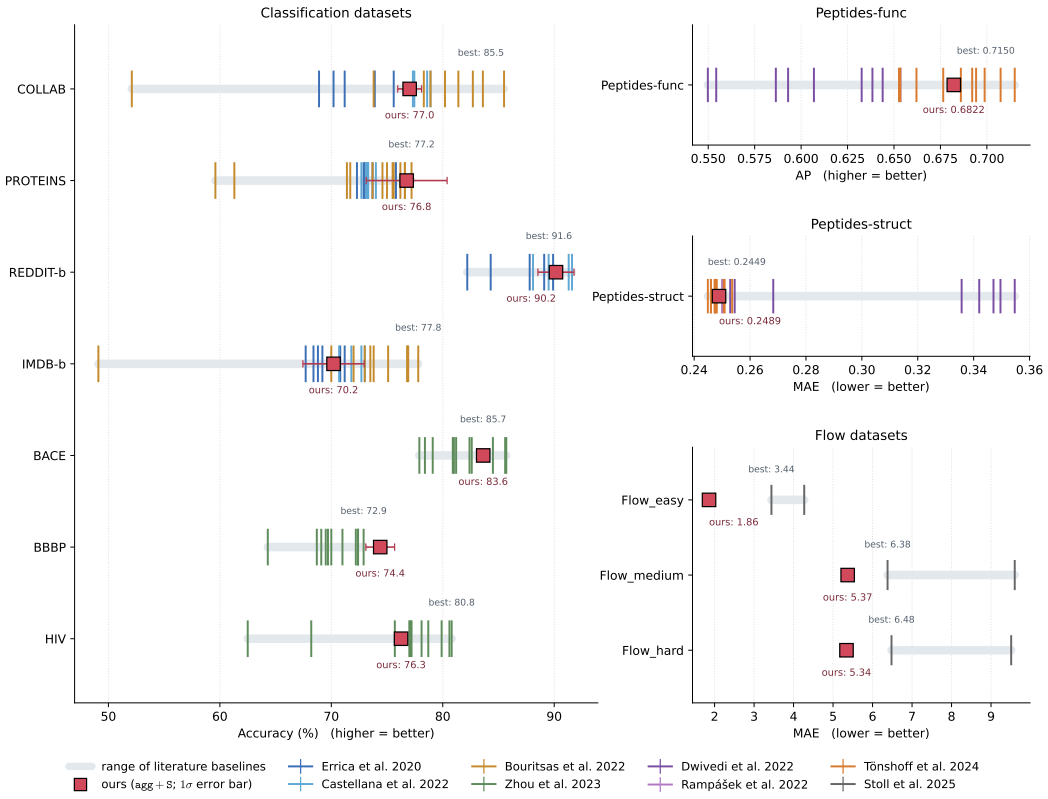


Figure S.6: Same comparison as in Figure 4, but for `agg + S`, i.e., using the expressive subset of invariants. Tick colours correspond to source papers of reported performance values. Numeric values are in Table S.18 and Table S.19 for classification datasets, in Table S.20 for Peptides and in Table S.21 for flow datasets.

Table S.18: **TU graph classification.** Test accuracy (%, mean ± std). Numbers are taken from the cited papers. We include every model that reports a value on at least one of the four datasets. "OOR" marks Out-Of-Resource entries reported in the source. Bold entries mark the single best reported value in a column.

	Accuracy (%) ↑			
Model	**COLLAB**	**PROTEINS**	**REDDIT-b**	**IMDB-b**
Errica et al. (2020)				
Baseline (MLP)	70.2 ± 1.5	75.8 ± 3.7	82.2 ± 3.0	70.8 ± 5.0
DGCNN	71.2 ± 1.9	72.9 ± 3.5	87.8 ± 2.5	69.2 ± 3.0
DiffPool	68.9 ± 2.0	73.7 ± 3.5	89.1 ± 1.6	68.4 ± 3.3
ECC	*OOR*	72.3 ± 3.4	*OOR*	67.7 ± 2.8
GIN	75.6 ± 2.3	73.3 ± 4.0	89.9 ± 1.9	71.2 ± 3.9
GraphSAGE	73.9 ± 1.7	73.0 ± 4.5	84.3 ± 1.9	68.8 ± 4.5
Castellana et al. (2022)				
CGMM	77.32 ± 2.2	74.0 ± 3.9	88.1 ± 1.9	72.7 ± 3.6
E-CGMM	77.45 ± 2.3	73.3 ± 4.1	89.5 ± 1.3	70.7 ± 3.8
ICGMM$_{\alpha\gamma}$	—	72.7 ± 3.4	—	—
ICGMM$^{f}_{\alpha\gamma}$	78.6 ± 2.8	73.3 ± 2.9	91.3 ± 1.8	73.0 ± 4.3
ICGMM	—	73.1 ± 3.9	—	—
ICGMM^{f}	78.9 ± 1.7	73.2 ± 3.9	**91.6 ± 2.1**	71.8 ± 4.4
Bouritsas et al. (2023)				
RWK	—	59.6 ± 0.1	—	—
GK (k=3)	—	71.4 ± 0.31	—	—
PK	—	73.7 ± 0.7	—	—
WL kernel	78.9 ± 1.9	75.0 ± 3.1	—	73.8 ± 3.9
GNTK	83.6 ± 1.0	75.6 ± 4.2	—	76.9 ± 3.6
DCNN	52.1 ± 0.7	61.3 ± 1.6	—	49.1 ± 1.4
DGCNN	73.8 ± 0.5	75.5 ± 0.9	—	70.0 ± 0.9
IGN	78.3 ± 2.5	76.6 ± 5.5	—	72.0 ± 5.5
GIN	80.2 ± 1.9	76.2 ± 2.8	—	75.1 ± 5.1
PPGN	81.4 ± 1.4	77.2 ± 4.7	—	73.0 ± 5.8
Natural GN	—	71.7 ± 1.04	—	73.5 ± 2.01
GSN-e	**85.5 ± 1.2**	76.6 ± 5.0	—	**77.8 ± 3.3**
GSN-v	82.7 ± 1.5	74.59 ± 5.0	—	76.8 ± 2.0
`agg + I` (ours)	79.74 ± 1.46	**77.35 ± 3.67**	90.7 ± 2.02	71.4 ± 3.69

Table S.19: **MoleculeNet.** Test area under receiver operating characteristic curve (%, mean ± std evaluated on a scaffold-split). Numbers are taken from the cited paper. We include every model that reports a value on at least one of the three datasets. Bold entries mark the single best reported value in a column.

	AUROC (%) ↑ *(scaffold split)*		
Model	**BACE**	**BBBP**	**HIV**
Zhou et al. (2023)			
D-MPNN	80.9 ± 0.6	71.0 ± 0.3	77.1 ± 0.5
AttentiveFP	78.4 ± 0.0	64.3 ± 1.8	75.7 ± 1.4
N-Gram$_{\mathrm{RF}}$	77.9 ± 1.5	69.7 ± 0.6	77.2 ± 0.1
N-Gram$_{\mathrm{XGB}}$	79.1 ± 1.3	69.1 ± 0.8	78.7 ± 0.4
PretrainGNN	84.5 ± 0.7	68.7 ± 1.3	79.9 ± 0.7
GROVER$_{\mathrm{base}}$	82.6 ± 0.7	70.0 ± 0.1	62.5 ± 0.9
GROVER$_{\mathrm{large}}$	81.0 ± 1.4	69.5 ± 0.1	68.2 ± 1.1
GraphMVP	81.2 ± 0.9	72.4 ± 1.6	77.0 ± 1.2
MolCLR	82.4 ± 0.9	72.2 ± 2.1	78.1 ± 0.5
GEM	85.6 ± 1.1	72.4 ± 0.4	80.6 ± 0.9
Uni-Mol	**85.7 ± 0.2**	72.9 ± 0.6	**80.8 ± 0.3**
`agg + I` (ours)	82.39 ± 0.53	**74.08 ± 0.0**	76.95 ± 0.34

Table S.20: **LRGB peptides.** Test average precision (multi-label classification, mean ± std) and test mean absolute error (multi-task regression, mean ± std). Numbers are taken from the cited papers. We include every model that reports a value on at least one of the three datasets. Bold entries mark the single best reported value in a column.

Model	**Peptides-func** *AP* ↑	**Peptides-struct** *MAE* ↓
Dwivedi et al. (2022)		
GCN	0.5930 ± 0.0023	0.3496 ± 0.0013
GCNII	0.5543 ± 0.0078	0.3471 ± 0.0010
GINE	0.5498 ± 0.0079	0.3547 ± 0.0045
GatedGCN	0.5864 ± 0.0077	0.3420 ± 0.0013
GatedGCN+RWSE	0.6069 ± 0.0035	0.3357 ± 0.0006
Transformer+LapPE	0.6326 ± 0.0126	0.2529 ± 0.0016
SAN+LapPE	0.6384 ± 0.0121	0.2683 ± 0.0043
SAN+RWSE	0.6439 ± 0.0075	0.2545 ± 0.0012
Rampášek et al. (2022)		
GPS	0.6535 ± 0.0041	0.2500 ± 0.0005
Tönshoff et al. (2024)		
GCN (retuned)	0.6860 ± 0.0050	0.2460 ± 0.0007
GINE (retuned)	0.6621 ± 0.0067	0.2473 ± 0.0017
GatedGCN (retuned)	0.6765 ± 0.0047	0.2477 ± 0.0009
GPS (retuned)	0.6534 ± 0.0091	0.2509 ± 0.0014
CRaWl	0.7074 ± 0.0032	0.2506 ± 0.0022
DRew	**0.7150 ± 0.0044**	0.2536 ± 0.0015
Exphormer	0.6527 ± 0.0043	0.2481 ± 0.0007
GRIT	0.6988 ± 0.0082	0.2460 ± 0.0012
Graph ViT	0.6942 ± 0.0075	**0.2449 ± 0.0016**
G-MLPMixer	0.6921 ± 0.0054	0.2475 ± 0.0015
`agg + I` (ours)	0.6874 ± 0.0022	0.2476 ± 0.0002

Table S.21: **Flow datasets.** Test mean absolute error (mean or mean ± std). Numbers are taken from the cited paper. We include every model that reports a value on at least one of the three datasets. Bold entries mark the single best reported value in a column.

	MAE ↓		
Model	**Flow_easy**	**Flow_medium**	**Flow_hard**
Stoll et al. (2026)			
GIN	3.44	9.60	9.51
Graph transformer	4.27	6.38	6.48
`agg + I` (ours)	**1.83 ± 0.01**	**5.30 ± 0.12**	**5.26 ± 0.03**

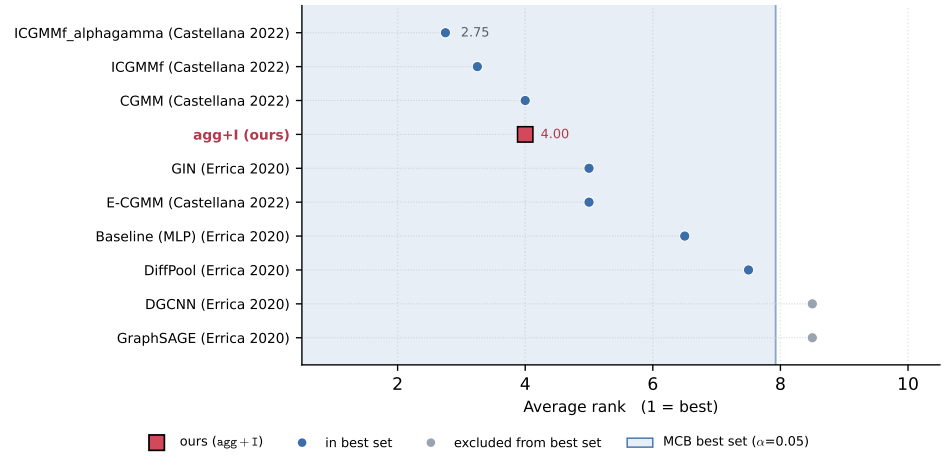


Figure S.7: Multiple comparisons with the best (MCB) for the TU datasets of Table S.18. Points show the average rank of each method over the $n = 4$ datasets for the $k = 10$ methods that report all four datasets. The shaded band is the MCB best set at $\alpha = 0.05$ (critical distance 5.17): methods inside it (blue) cannot be ruled out as the best, methods outside it (grey) are significantly worse than the best. Our `agg + I` is shown in red.

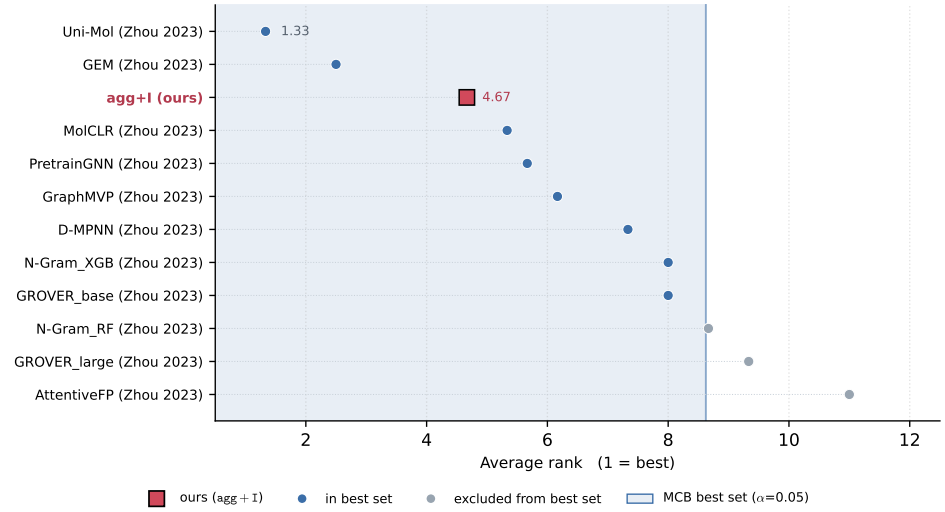


Figure S.8: Multiple comparisons with the best (MCB) for the MoleculeNet datasets of Table S.19. Points show the average rank of each method over the $n = 3$ datasets for all $k = 12$ methods. The shaded band is the MCB best set at $\alpha = 0.05$ (critical distance 7.29): methods inside it (blue) cannot be ruled out as the best, methods outside it (grey) are significantly worse than the best. Our `agg + I` is shown in red.

Multiple comparisons with the best. We compare methods within each benchmark table using Hsu's multiple comparisons with the best (Horrace and Schmidt, 2000; Hsu, 1984, MCB) on average ranks (Demšar, 2006); cf. Figures S.7 to S.9. On every dataset, methods are ranked by their reported mean, and methods with missing entries are excluded from that table's analysis. A method belongs to the *best set* if its average rank is within the critical distance c from the best average rank. Following Hsu (1984) and Demšar (2006), the critical distance is $c = d_\alpha \sqrt{k(k+1)/(6n)}$, where k is the number of methods, n the number of datasets, and d_α the one-sided Dunnett constant for $k - 1$ comparisons at $\alpha = 0.05$. With only $n = 2$–4 datasets per table, c is large and MCB has limited power: Membership in the best set means that a method is not significantly worse than the best.

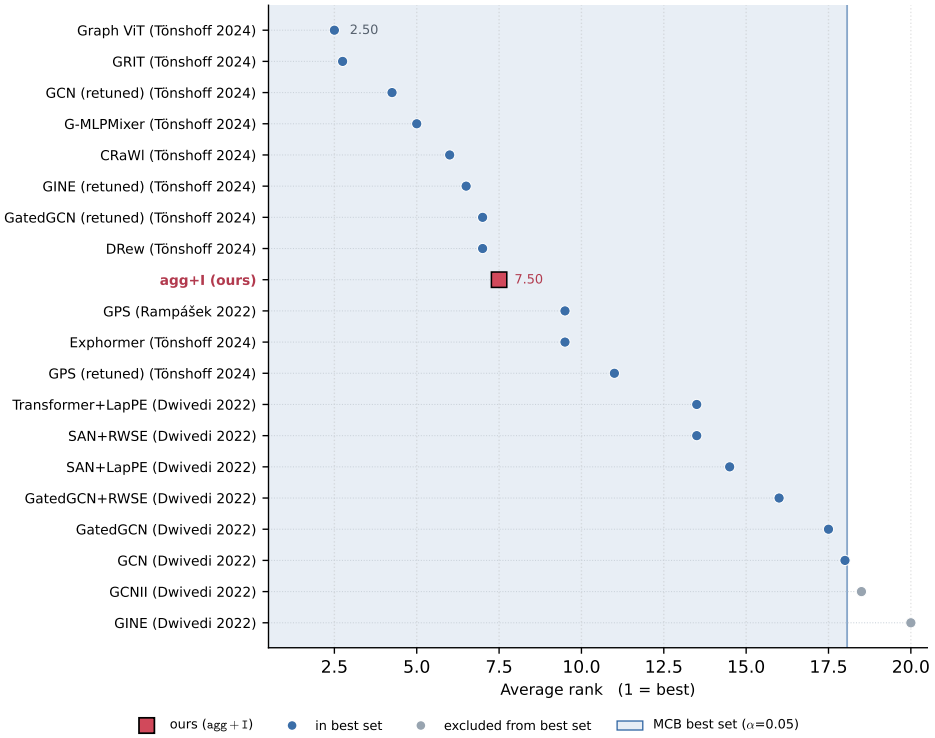


Figure S.9: Multiple comparisons with the best (MCB) for the LRGB datasets of Table S.20. Points show the average rank of each method over the $n = 2$ datasets for all $k = 20$ methods. The shaded band is the MCB best set at $\alpha = 0.05$ (critical distance 15.56): methods inside it (blue) cannot be ruled out as the best, methods outside it (grey) are significantly worse than the best. Our $\texttt{agg} + \texttt{I}$ is shown in red. With only two datasets, this comparison is descriptive only.

E INVARIANTS REVEAL OBSTRUCTIONS TO MULTI-TASK LEARNING

We investigate whether the structural differences between datasets observed in Section 3.2 can *impact* multi-task learning performance. Unlike most multi-task and pretraining approaches on graphs (Du et al., 2020; Frasca et al., 2025; Hu* et al., 2020; Liu et al., 2024; Qiu et al., 2020; Yu et al., 2020), we avoid a separate pretraining objective and train directly on two downstream tasks simultaneously, so that structural differences directly act on the prediction tasks. To our knowledge, neither multi-task training across graph domains nor gradient alignment during multi-task training on graphs has been studied before. We address this gap by assessing the correlation between structural differences and gradient alignment, which is known to be an important indicator of the learning process (Du et al., 2020; Yu et al., 2020), as well as structural differences and the performance outcomes. We measure structural difference as the accuracy of pairwise meta-classification: an accuracy near 50% indicates structurally similar datasets, a high accuracy structurally dissimilar ones.

Model and setup. We use a shared Graph Isomorphism Network (Xu et al., 2019, GIN) graph-encoder, a message-passing network (Corso et al., 2024), followed by sum pooling and one task head per dataset (hyperparameters in Table S.11). We train on all 66 pairs of 12 datasets (BACE, BBBP, HIV, Lipo, ZINC, QM9, NCI1, NCI109, PROTEINS, Peptides-func, Peptides-struct, and CO_BA_small), which span structurally similar and dissimilar pairs, i.e., both frequently confused and easily distinguished datasets in the meta-classification confusion matrix (Fig. S.4). In each step, one batch per dataset is passed through the shared encoder, and gradients are computed separately for both batches to measure their alignment in the encoder. All models are trained from scratch for each pair, repeated for 5 random seeds. To isolate structure and allow training across domains, node features are replaced by constant unit features. We train from scratch for each pair and each of 5 seeds; an epoch iterates over the larger dataset, while cycling through the smaller one, and training stops once the validation loss on both datasets has plateaued. We evaluate on each test set at the epoch with minimal validation loss.

Results. Higher meta-classification accuracy coincides with worse multi-task test performance on 10 of 12 datasets, with no clear correlation for BBBP and PROTEINS (Figure S.10), and with lower gradient alignment on all 12 datasets (Figure S.11): the more structurally distinguishable two datasets are, the worse they tend to train together. Although we do not observe strictly negative alignment Du et al. (2020); Hu* et al. (2020), we hypothesize that the reduced alignment impedes multi-task performance. Invariants can thus serve as a diagnostic for obstructions to joint training or pretraining. As a limitation, each pair introduces both a second graph distribution and a second task, and we do not disentangle the two effects.

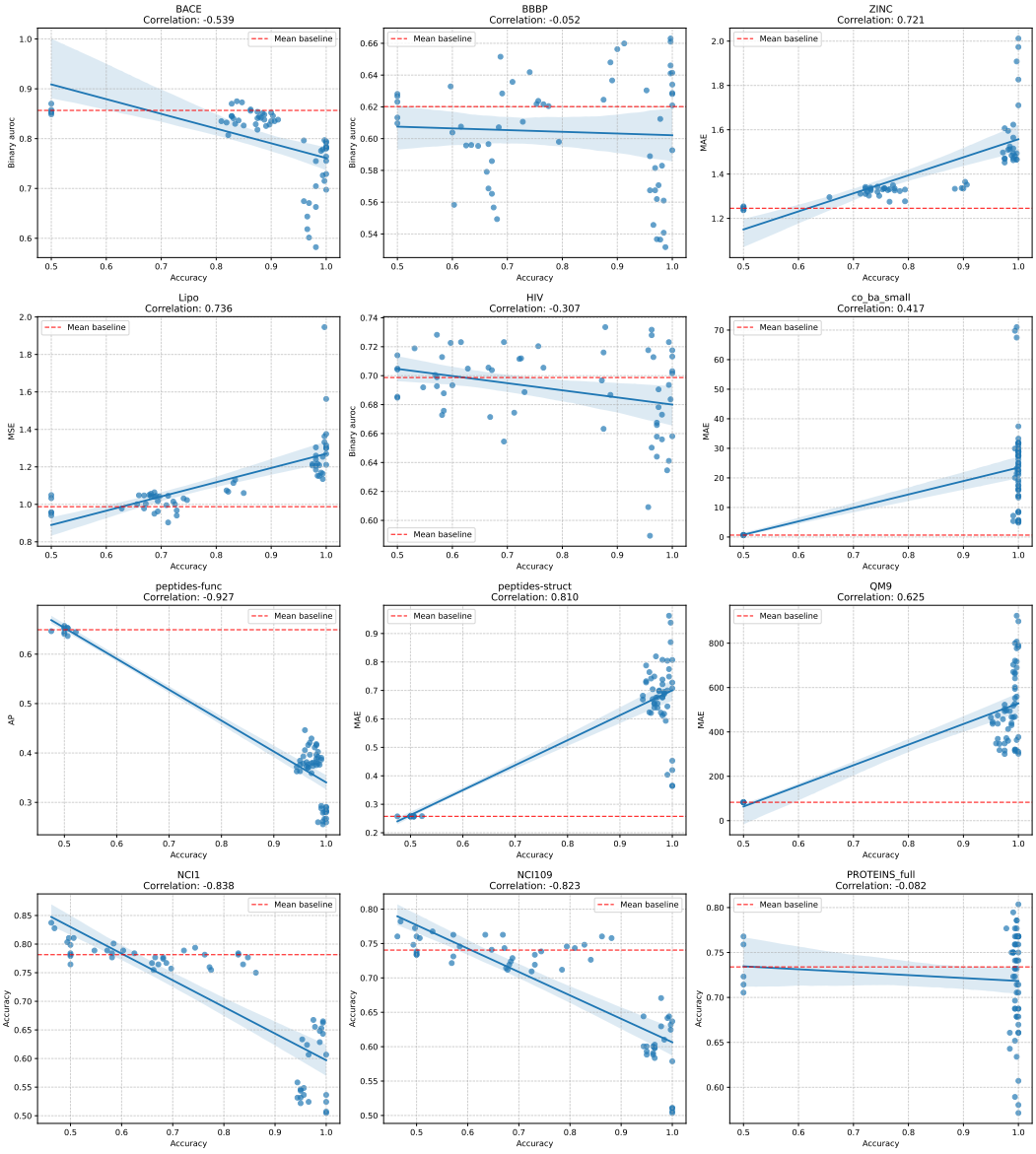


Figure S.10: **Multi-task test performance, all datasets.** Correlation between meta-accuracy and test performance after multi-task learning. Higher accuracy is correlated with lower performance after multi-task learning on all datasets except BBBP and PROTEINS, which show no clear correlation (notice that for some metrics, lower values are preferable). We repeat multi-task training and classification for 5 seeds for each pair of datasets, showing one seed as one dot.

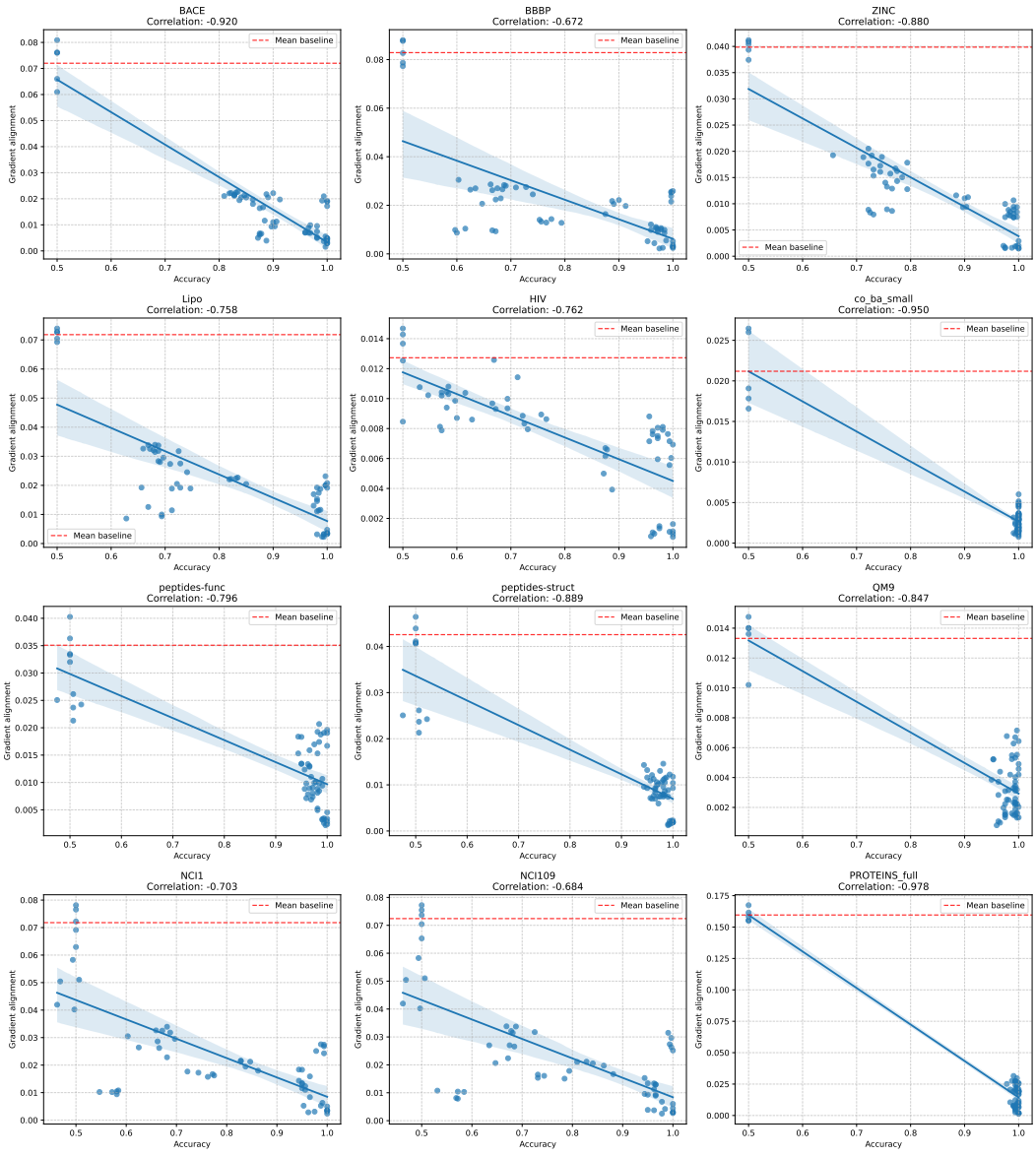


Figure S.11: **Multi-task gradient alignment, all datasets.** Correlation between meta-accuracy and gradient alignment during multi-task learning. Higher accuracy is consistently correlated with lower gradient alignment. We repeat multi-task training and classification for 5 seeds for each pair of datasets, showing one seed as one dot.