WOULD AN UNTRAINED ENCODER DO AS WELL? THE MISSING BASELINE IN GRAPH SELF-SUPERVISED LEARNING

Anonymous authors
Paper under double-blind review

ABSTRACT

Graph self-supervised learning is judged by freezing a pretrained encoder, fitting a linear probe, and comparing accuracies on small benchmarks, where methods are separated by a point or two. These comparisons almost never ask whether the same encoder, never trained, would do as well. We run this control for graph joint-embedding predictive architectures (JEPAs) on seven benchmarks, with three partitioners, two read-outs and two implementations, including the official Graph-JEPA code, whose published accuracies we reproduce. We find that pretraining adds no detectable accuracy over a randomly initialised copy of the encoder except on the smallest dataset, and that under the published read-out the untrained encoder is significantly better on three datasets. Pooled over benchmarks, every setting excludes a common gain above one and a half points. Pretraining does reshape the representation, without improving what a linear probe reads, and the most robust partition effect, structural-entropy over METIS partitions on NCI1, is nearly as large for the never-trained encoder as for the trained one. The benchmarks set the boundary: under a corrected test they cannot resolve gains below two to six points, several times the margins that rank published methods. An untrained encoder costs one run with zero epochs, and graph self-supervised learning should be measured against it.

1 INTRODUCTION

Self-supervised learning on graphs is judged almost entirely through one protocol: pretrain an encoder without labels, freeze it, fit a linear classifier on its graph embeddings, and report ten-fold cross-validated accuracy on small benchmarks from the TU collection (Morris et al., 2020; Sun et al., 2020; You et al., 2020; Hou et al., 2022; Skenderi et al., 2025). Methods are then ranked by differences of one or two points, on datasets that hold between a few hundred and a few thousand graphs, where such differences are fragile (Shchur et al., 2018; Errica et al., 2020).

A table of this kind does not say how much of the probed accuracy the pretraining produced. The natural control is the same architecture at its random initialisation, read by the same probe. In vision and language this control has a long and sobering record: random convolutional filters (Jarrett et al., 2009; Saxe et al., 2011) and random sentence encoders (Wieting & Kiela, 2019) come close to trained ones under a linear read-out. For graph self-supervised learning on the TU benchmarks the control has been run, with mixed results: untrained GINs match contrastive and non-contrastive encoders on some benchmarks and trail them on others, depending on the probe (Suresh et al., 2021; Trivedi et al., 2022b;a; Khanna et al., 2025), and several of these studies recommend reporting an untrained baseline. It has also been run for bootstrapped and masked objectives at node level (Thakoor et al., 2022; Song et al., 2025), and for molecular JEPA-style encoders pretrained on large external corpora, which do beat it (Kulczykowski & Łabędzki, 2026; Ariguib et al., 2026). The patch-based graph JEPAs evaluated on the TU benchmarks report no untrained arm (Skenderi et al., 2025; Xu et al., 2026).

We ask this question for graph joint-embedding predictive architectures (JEPAs) (Skenderi et al., 2025; Xu et al., 2026) and find that the answer is largely yes. Across seven benchmarks, three partitioners, two read-outs and two implementations, a pretrained encoder beats a randomly initialised

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.

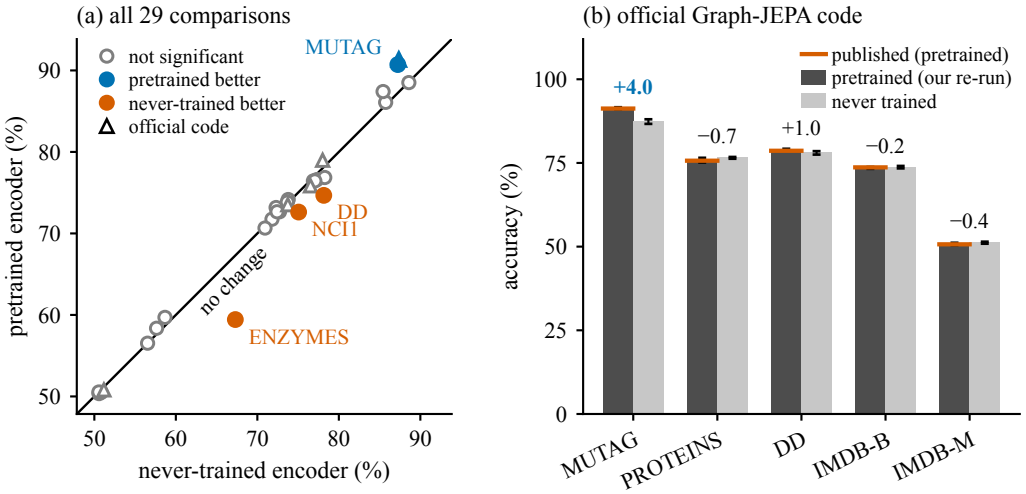

Figure 1: **Pretraining barely moves the probed accuracy.** (a) Accuracy of the pretrained encoder against a never-trained copy of it, for all 29 comparisons (seven datasets, three partitioners, two read-outs, two implementations). Points on the line mean pretraining changed nothing; coloured points differ significantly under the corrected test of Equation 1. The 3 coloured points below the line all come from the backbone's published read-out. (b) The official Graph-JEPA code at its published configurations: published accuracy, our re-run of the pretrained encoder and the never-trained encoder (mean ± SD over five seeds), each paired difference above its pair.

copy of itself significantly in 2 of 29 comparisons, each time on MUTAG, the smallest dataset. Under the backbone's own published read-out the untrained encoder is significantly *better* on 3 of 5 datasets (Figure 1). The official Graph-JEPA code, re-run at its published configurations, reproduces its published accuracies and gives the same verdict.

Pretraining is not inert. It moves the representation further from its initialisation than two independent initialisations are from each other, yet what a linear probe can read from it does not improve. Nor could these benchmarks show it: under a corrected test, five seeds of ten-fold cross-validation cannot resolve differences below 2.0 to 6.4 points on these datasets, while the best and second-best published methods differ by a median of 0.45.

The two gaps combine in practice. In a hierarchical graph JEPA, partitions from structural entropy (SE), whose hierarchy chooses its own granularities, score higher than a tuned fixed-granularity ladder on all six datasets under the standard protocol. The lead does not survive a corrected test, like-for-like baselines and the untrained control: the naive test over repeated folds inflates its statistic by a factor of 2.56, two baselines had run under a different augmentation regime, and on NCI1 what remains of SE's advantage comes from the architecture rather than from pretraining (Section 4).

We do not propose a new method; we measure what pretraining contributes to a graph JEPA under the protocol the field uses. Our contributions are:

- **The finding and its breadth.** Pretraining a graph JEPA adds no detectable linear-probe accuracy over a randomly initialised copy of its encoder, except on MUTAG, on seven TU benchmarks, across three partitioners, two read-outs and two implementations, including the official Graph-JEPA code, which reproduces its published accuracies (Section 3.1, Figure 1).

- **The mechanism.** Pretraining moves the representation further from its initialisation than two initialisations are from each other, without improving what a linear probe reads, and the partition acts through the architecture: an exact split of the SE-over-METIS gap assigns most of it to the never-trained encoder on NCI1 (Sections 3.2 and 3.3).

- **The boundary.** At the published schedules pretraining helps only on MUTAG, the smallest dataset, and under a corrected test these benchmarks cannot resolve gains below 2.0 to 6.4 points (Section 3.4, Figure 4).
- **A worked failure and a checklist.** How the missing baseline and an uncorrected test produce an apparent win, and the checks that expose it (Sections 4 and 5.1).

This paper does not claim that graph JEPAs cannot learn useful representations, nor that any published number is wrong: the official implementation we re-ran reproduces its published accuracies to within 0.34 points. Nor is the untrained baseline new to graph learning (Section 5.3). Rather, it measures what the frozen-probe protocol can attribute to pretraining on the benchmarks the field uses, and argues for making an inexpensive control routine.

2 SETTING

Hierarchical graph JEPA. Graph-JEPA (Skenderi et al., 2025) splits each graph into patches with METIS (Karypis & Kumar, 1998), following patch-based graph transformers (He et al., 2023; Gao et al., 2022), encodes every patch with a GINE network, and runs a transformer over the patch tokens. A predictor, given the tokens of a context patch, regresses for each target patch a point on a hyperbola computed from the mean of its embedding under an exponential-moving-average copy of the encoder, one scalar per patch, a variant of I-JEPA (Assran et al., 2023). HP-JEPA (Xu et al., 2026) makes the patches hierarchical: each graph is partitioned at several granularities (a *ladder* of cluster counts K), and targets live on a Lorentz hyperboloid. After pretraining the encoder is frozen, patch tokens are mean-pooled into a graph embedding, and an ℓ_2-regularised logistic regression is fitted on the training folds. Our read-out uses the finest rung and regularisation strength 1; HP-JEPA's published read-out mixes all rungs with weights that are three quarters uniform and one quarter learned, under strength 0.01. Appendix A lists the configuration.

Partitioners. Three ways of building the ladder are compared. *METIS* is the backbone's default, at hand-set counts. *SE fixed-K* minimises the structural entropy of the partition (Li & Pan, 2016) at the same counts. *SE levels* reads the ladder off the levels of the graph's own SE hierarchy and appends a node-level rung, so no K is set by hand (Appendix B).

Protocol. Seven datasets are used: MUTAG, ENZYMES, NCI1, PROTEINS, IMDB-BINARY, IMDB-MULTI and DD. Each arm runs five seeds of stratified ten-fold cross-validation, and arms are paired by seed and fold. With $J = 50$ paired differences of mean $\bar{d}$ and variance s_d^2, and a test-to-training ratio of $1/9$, we report the corrected repeated cross-validation statistic of Nadeau & Bengio (2003), recommended by Bouckaert & Frank (2004):

$$t_c = \frac{\bar{d}}{\sqrt{\left(\frac{1}{J} + \frac{1}{9}\right) s_d^2}}, \qquad \text{significant when } |t_c| > 2.01 \text{ (two-sided 5\%, 49 degrees of freedom).} \tag{1}$$

The naive statistic drops the $1/9$. Two defaults were fixed in every arm before the comparisons below: inputs are standardised, and targets are ℓ_2-normalised before the hyperbolic map, without which the loss can overflow, which also confines each target scalar to $\pm 1/\sqrt{512}$; together they moved accuracies by up to 7.2 points (Appendix C).

The untrained arm. The untrained encoder is the trained arm's configuration with zero training epochs: its weights keep the default initialisation drawn from that run's seed, its batch normalisation statistics keep their initial values, and it sees the same partitions and the same probe. It costs one feature extraction per fold.

Tests fixed in advance. The trained-versus-untrained comparisons of Section 3.1 and two held-out tests in Section 4 were specified before they ran, as Rabby & Auer (2026) also do for Graph-JEPA: arms, datasets, statistic and decision rule were recorded first, and later changes were dated notes written before results were read. The other comparisons of Section 4 and the analyses marked exploratory were not. Every number in this paper is generated from the fold records.

Table 1: **Trained minus untrained encoder on our backbone.** Accuracy (%) of the METIS arm, and trained − untrained differences in points over 5 seeds × 10 folds; bold: $|t_c| > 2.01$. *METIS, pub.*: the saved METIS models re-probed under the published read-out, without PROTEINS and IMDB-M, whose trained models cannot be reloaded exactly (Section 3.1). Never-trained SE arms were not run on DD. Intervals are in Figure 5, statistics in Table 7.

	METIS accuracy		trained − untrained (points)			
dataset	untrained	trained	METIS	METIS, pub.	SE levels	SE fixed-K
MUTAG	85.74	86.08	+0.34	−0.08	**+3.49**	+1.98
ENZYMES	57.63	58.37	+0.73	**−7.87**	+1.03	0.00
NCI1	71.80	71.75	−0.05	**−2.44**	+0.37	+0.17
PROTEINS	77.31	76.61	−0.70	–	−0.43	−0.61
IMDB-B	72.72	72.66	−0.06	−0.28	+0.88	+0.24
IMDB-M	50.57	50.35	−0.23	–	−0.35	−0.01
DD	78.27	76.86	−1.41	**−3.48**	–	–
trained significantly better			0/7	0/5	1/6	0/6

3 Would an untrained encoder do as well?

3.1 Pretraining does not beat the encoder's initialisation

The main test. We fixed the METIS arm, the backbone's own default, and changed only the number of training epochs. The decision rule, set before the runs: pretraining *helps* if the trained encoder beats the untrained one with $t_c > 2.01$ on at least four of the seven datasets, and *barely helps* if on at most two. Table 1 gives the result: the trained encoder wins on 0 of 7, and the untrained encoder is numerically ahead on 5. On DD it reaches 78.27%, the best DD accuracy of any arm on our backbone. By this rule, pretraining barely helps.

Is it the read-out? Our read-out uses one rung and strong regularisation, which might hide what the other rungs learned. We therefore re-probed every saved METIS fold model under the read-out HP-JEPA publishes, without retraining; it fits a fixed-strength probe to unscaled features, so part of any difference may reflect feature scale rather than content. Each re-probe also re-fitted our read-out, which had to reproduce the stored accuracy exactly; it did on all 500 folds of the five datasets that could be reloaded. PROTEINS and IMDB-M failed the check because their trained arms re-draw partitions online, and our validity rule excludes them. Under the published read-out the trained encoder wins on 0 of 5 datasets. The untrained encoder is significantly *better* on 3: ENZYMES (−7.87 points, t_c −3.74), NCI1 (−2.44, t_c −3.16) and DD (−3.48, t_c −2.34).

Is it the partitioner? A partitioner that suits the objective might give pretraining something to learn. We repeated the test for both SE partitioners, each against its own like-for-like trained arm, with the same rule scaled to six datasets. The SE-levels encoder wins on 1 of 6 (MUTAG, +3.49 points, t_c 2.15) and the SE fixed-K encoder on 0 of 6. By the rule, pretraining barely helps under either partitioner; the one significant cell is on MUTAG.

Is it our implementation? Every result so far comes from one reproduction of a hierarchical graph JEPA, which runs one configuration on all datasets and lands 0.2 to 5.6 points below HP-JEPA's published accuracies. We therefore ran the official Graph-JEPA code (Skenderi et al., 2025) with each dataset's published configuration, trained and with zero epochs, on the five of its seven TU datasets that our study shares (the two REDDIT sets were not run). Upstream fixes one ten-fold split and repeats it over five seeds; we pair the arms by seed and fold. Small changes let the reference code run zero epochs, call METIS and log each run (Appendix E). A dataset counted only if its trained arm landed within three points of its published accuracy.

Table 2 gives the result. The re-run reproduces every published accuracy to within 0.34 points, far inside the three-point tolerance, so the comparison is made on the published model as published. Pretraining wins significantly on 1 of the 5 datasets, MUTAG, by +3.99 points (t_c 2.03); on the

Table 2: **The official Graph-JEPA code under its published configurations.** Published accuracy, our re-run of the trained arm and the untrained arm (% ± standard deviation over the five seed means), and the paired trained − untrained difference with t_c in brackets. Bold: $|t_c| > 2.01$. *Reproduced*: within three points of the published value, the condition for a dataset to count.

dataset	published	trained (re-run)	untrained	trained − untrained	reproduced
MUTAG	91.25	91.35 ± 0.30	87.36 ± 0.69	**+3.99** (2.03)	yes
PROTEINS	75.67	75.84 ± 0.69	76.52 ± 0.27	−0.68 (−0.55)	yes
DD	78.64	78.98 ± 0.25	78.01 ± 0.51	+0.97 (0.89)	yes
IMDB-B	73.68	73.52 ± 0.28	73.74 ± 0.36	−0.22 (−0.17)	yes
IMDB-M	50.69	50.80 ± 0.35	51.17 ± 0.30	−0.37 (−0.34)	yes

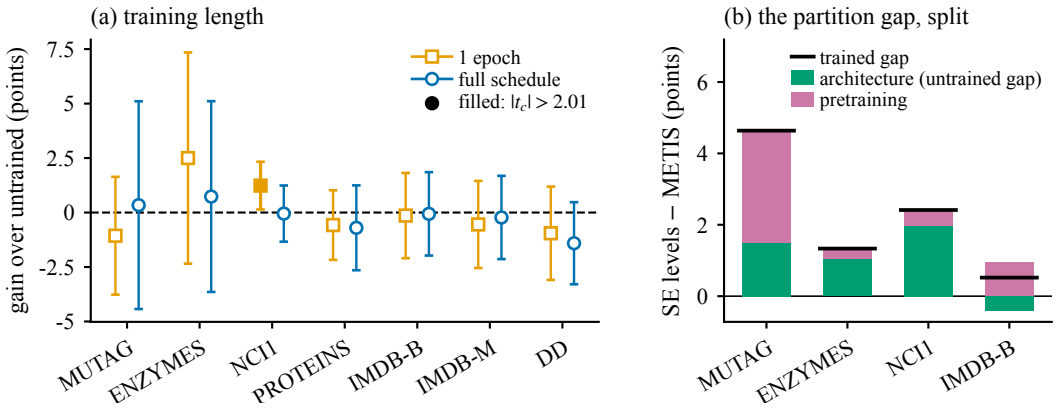

Figure 2: **Pretraining against architecture.** (a) Gain over the untrained encoder after one epoch (squares) and after the full schedule (circles), METIS arm, with corrected 95% intervals; filled markers are significant. (b) The trained gap between SE-level and METIS partitions (black bar) split exactly into the gap between never-trained encoders (architecture) and the difference between the two pretraining gains, on the datasets where all four arms share one partition bank.

others the untrained encoder is within the noise of the trained one. By the same rule, pretraining barely helps in the official code as well.

The untrained encoder against published results. For orientation only, since the comparison is unpaired, the never-trained official encoder scores above the published accuracy of the pretrained Graph-JEPA on PROTEINS, IMDB-B and IMDB-M (Appendix E).

Where pretraining does help. Both significant wins of a trained encoder over its untrained copy are on MUTAG, the smallest dataset: the SE-levels encoder (+3.49 points, t_c 2.15) and the official code (+3.99, t_c 2.03, or 1.81 if its seeds are averaged within each fixed fold). If pretraining did nothing anywhere, about 0.7 one-sided false wins would be expected among the 29 comparisons, so neither survives a multiplicity correction; that both fall on one dataset still suggests a real effect there, which 188 graphs cannot size: its corrected intervals reach up to 7.9 points.

3.2 Pretraining moves the features but not the probed accuracy

Is it too little training? If the schedule were simply too short, more epochs should help more. Figure 2a shows the opposite. A single epoch is significantly better than no training on NCI1 alone (+1.24 points, t_c 2.27), and the full schedule keeps none of that gain. The training loss says little here: with ℓ_2-normalised targets each target scalar is small, so the METIS arm's median final loss of at most 2.7×10^{-5} does not show the task is learned; the official code, whose targets are not normalised, is the control for this choice.

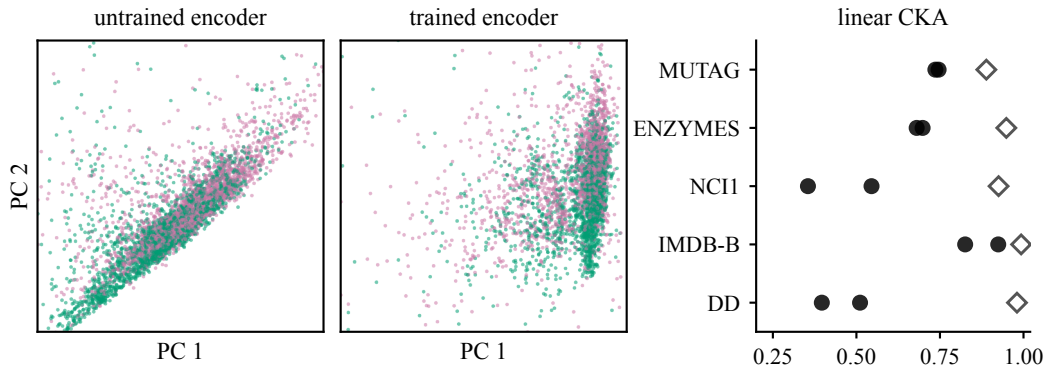

Figure 3: **Training changes the features but not the probed accuracy.** Left, centre: the first two principal components of the probe's features on NCI1, for the untrained encoder and the trained encoder built from the same initialisation (seed 12, fold 0; axes clipped to the 1st–99th percentiles; pink: class 0, green: class 1). Right: linear CKA between a trained encoder and its own initialisation (two seeds, filled) and between two independent initialisations (open). Exploratory analysis.

Representation drift. This analysis is exploratory; it was not planned before the runs. Because the untrained arm is built from the same seed before any update, it is exactly the trained encoder's own initialisation, so the two can be compared directly. We measured their similarity with linear CKA (Kornblith et al., 2019) on the probe's features, for fold 0 of two seeds (Figure 3, right). On all 5 datasets measured, a trained encoder is further from its own initialisation (CKA 0.45 to 0.87, averaged over the two seeds) than two independent initialisations are from each other (0.89 to 0.99). Pretraining is therefore not inert: it reshapes the representation, most of all on NCI1, where the leading components change visibly (Figure 3, left and centre). What a linear probe reads from it does not improve: on that fold the untrained and trained encoders score 67.15% and 68.86%, and over all fifty folds the trained encoder's advantage is -0.05 points. CKA can move without functional change (Davari et al., 2023), so the drift shows only that the arms differ. Random initialisations resemble one another more than any of them resembles its trained descendant, though part of that similarity can come from the inputs themselves (Cui et al., 2022).

3.3 The partition acts through the architecture

Why the untrained encoder is strong here. Under a linear probe, untrained GINs trailed contrastive encoders on NCI1, PROTEINS and IMDB-B in earlier work (Suresh et al., 2021), while our untrained encoders are nowhere significantly behind the fully trained ones there. Our reading points to the architecture: random projections of the computed random-walk features, pooled within the patches of a partitioner, already give a linear probe the structural statistics that simple baselines show to be strong here (Cai & Wang, 2018; Errica et al., 2020). On top of them, each randomly weighted GINE layer adds a random, approximately injective code of rooted subtrees, the quantity Weisfeiler–Leman kernels count (Xu et al., 2019; Shervashidze et al., 2011), and untrained graph models of this kind classify graphs competitively (Navarin et al., 2023; Zambon et al., 2020).

The partition gap splits into architecture and pretraining. The trained gap between two partitions splits exactly into the gap between the never-trained encoders, which the architecture produces, and the difference between the two gains from pretraining (Figure 2b; exploratory). For SE-level against METIS partitions, the first term carries +1.99 of the +2.41-point gap on NCI1, where the never-trained gap alone is significant (t_c 3.36, all five seeds), and the second term carries +3.15 of +4.64 on MUTAG, the dataset where pretraining helps. SE fixed-K partitions behave the same on NCI1 (+1.88 points never-trained, +2.10 trained). With trained encoders (no never-trained random arm was run), much of the effect is the partition's shape: a random partition with the SE ladder's rung counts and cluster sizes beats METIS by +1.74 points (t_c 2.24), and the SE ladder leads it by only +0.68. Partition shape has mattered more than membership in trained patch models (He et al., 2023; Piccoli et al., 2025) and outside self-supervised learning (Zhu et al., 2026), while SE coding trees

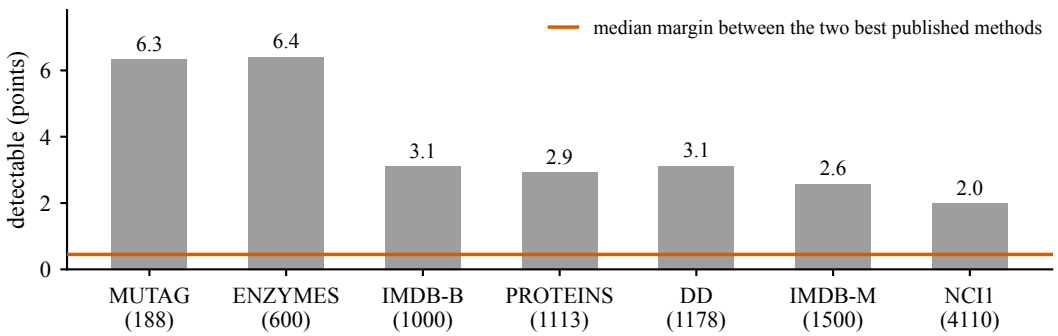

Figure 4: **What these benchmarks can resolve.** The smallest trained − untrained difference a two-sided 5% test detects with 80% power, $(t_{0.975} + t_{0.80})$ corrected standard errors (median over the comparisons on each dataset; graphs per dataset in brackets), against the median margin between the two best self-supervised methods in the table collected by Xu et al. (2026).

Table 3: **What each check did to the SE ladder's lead.** Accuracy differences in points, paired over 5 seeds × 10 folds; bold: significant under the row's statistic (naive t in the first row, corrected t_c elsewhere). *Ahead*: datasets with a positive difference; *sig.*: significant ones. *Like-for-like* replaces the PROTEINS and IMDB-M baselines by re-runs without online re-partitioning and edge dropping. Every statistic is in Table 6.

comparison	MUTAG	ENZYMES	NCI1	PROTEINS	IMDB-B	IMDB-M	ahead	sig.
levels − fixed-K, naive t	**+3.31**	**+3.17**	+0.32	+0.59	+0.52	+0.56	6/6	2/6
same, corrected t_c	+3.31	+3.17	+0.32	+0.59	+0.52	+0.56	6/6	0/6
same, like-for-like	+3.31	+3.17	+0.32	0.00	+0.52	−0.05	4/6	0/6
levels − METIS, like-for-like	**+4.64**	+1.33	**+2.41**	+0.18	+0.52	−0.36	5/6	2/6
levels − random tree	+1.16	+0.23	+0.68	−0.38	−0.18	−0.53	3/6	0/6
levels, trained − untrained	**+3.49**	+1.03	+0.37	−0.43	+0.88	−0.35	4/6	1/6

beat unmatched random trees in trained and training-free models (Wu et al., 2022; 2023). Structural entropy thus acts on a graph JEPA mainly as an architectural choice, the granularity of its patches. The split was computed on runs already made, so it is a hypothesis for held-out benchmarks.

3.4 What the benchmarks can resolve

A null here is a bound, and the bound is wide. Small TU benchmarks are known to separate methods poorly (Dwivedi et al., 2023; Hu et al., 2020a); following Card et al. (2020), Figure 4 measures how poorly. With five seeds of ten-fold cross-validation and the corrected test, a difference must reach 2.0 points on NCI1, the largest benchmark, and 6.4 points on ENZYMES before it is detected with 80% power. The margins at the top of recent comparison tables are far smaller: in the table collected by Xu et al. (2026), the best self-supervised method leads the runner-up by a median of 0.45 points on the 5 datasets it shares with ours, and by at most 0.60. The verdict does not hinge on the correction either: under the naive test, pretraining wins in 5 of the 29 comparisons, and across their seed-level means the trained encoder is ahead in 68 of 145 (4 tied), close to the split a coin would give. Pooled over datasets with inverse-variance weights, the trained advantage is −0.36 [−1.08, +0.37] for METIS, +0.31 [−0.38, +0.99] for SE levels, +0.06 [−0.80, +0.91] for SE fixed-K and +0.31 [−0.80, +1.41] in the official code, so every setting excludes a common gain above 1.4 points. The defensible reading is a bound, not parity.

4 Anatomy of an apparent win

Table 3 follows one claim through the checks, one row per check. The SE ladder was ahead of the tuned fixed-K ladder on all six datasets, by more than three points on MUTAG and ENZYMES, and the naive paired test called both of those significant (first row).

Folds are not independent. Two training sets of ten-fold cross-validation share eight ninths of their graphs, so the naive test understates the variance of the mean difference; under Equation 1 every naive statistic over fifty folds shrinks by $(1 + J/9)^{1/2} = 2.56$. No lead of the SE ladder over the fixed-K ladder survives (second row).

Baselines were not like-for-like. Re-probing saved models (Section 3.1) could not reproduce the PROTEINS and IMDB-M baselines exactly. The cause was that both baselines, SE fixed-K and METIS, had inherited online re-partitioning with thirty per cent edge dropping on these two datasets, while the SE ladder used a fixed bank. Re-run without them, the fixed-K baselines tie with the ladder on both datasets (0.00 and -0.05 points), so "ahead on six of six" becomes "ahead on four" (third row). METIS is then significantly behind only on MUTAG and NCI1 (fourth row).

Randomised and held-out controls. If the SE hierarchy helped because of which vertices it groups, a random hierarchy of the same shape should not. Drawn per graph with the same rung counts and cluster sizes but random membership, it matched the SE hierarchy on every dataset (fifth row), as random patches matched METIS in Graph-JEPA's own ablation (Skenderi et al., 2025). Two tests specified in advance failed the same way. Restricting node-level attention to SE blocks did not beat METIS blocks or random blocks on NCI1 (-0.25 and -0.17 points, t_c -0.52 and -0.40). A node-level rung, which had raised accuracy on 5 of 6 datasets, none significantly, did not replicate on held-out DD (+0.29, t_c 0.31; Appendix D).

One explanation fits every null. Section 3 explains every partition null at once: the probe reads little that pretraining put into the encoder, so no partition can matter through pretraining. The last row of Table 3 applies the control to the SE ladder itself. Its encoder beats its own untrained copy on MUTAG alone, the dataset where its lead was largest, and Figure 2b shows that this lead, unlike the one on NCI1, is mostly the pretraining term (t_c 1.37 on its own).

5 Discussion

5.1 A protocol for graph self-supervised claims

Six checks changed our conclusions, each at little cost next to the runs it checked: run the untrained encoder with the same partitions and probe, paired by seed and fold; correct the repeated cross-validation test and report what it can detect (Figure 4); diff every configuration field against its template, with no exemptions, since an exemption hid two mismatched baselines; re-probe saved models and require an exact reload, which found them; fix the decision rule first and keep a held-out dataset, on which the node-level rung failed; and generate every reported number from fold records.

5.2 What this means for graph self-supervised learning

What a pretraining method is responsible for is its accuracy minus that of its own initialisation under the same probe. In a graph JEPA this zero point is high, because random-walk features, a partition and a transformer describe a graph before any weight is trained, so a table without the untrained row credits all of that to the objective. A claim on small benchmarks should also state the smallest effect it could have detected (Figure 4); where pretraining does beat random initialisation, it pretrains on far more graphs (Khanna et al., 2025; Kulczykowski & Łabędzki, 2026). Which fixed parts of the architecture matter can be measured on the untrained encoder alone, at a fraction of the cost of pretraining; a small training loss is no evidence of learning.

5.3 Related work

Random-weight baselines. Random filters were competitive in early object recognition (Jarrett et al., 2009; Saxe et al., 2011) and random encoders in sentence classification (Wieting & Kiela, 2019). For graphs, untrained reservoir models classify graphs competitively (Gallicchio & Micheli, 2020), and random-initialisation encoders are strong baselines for node-level pretraining (Kipf & Welling, 2017; Veličković et al., 2019).

Untrained encoders in graph self-supervised learning. On TU benchmarks, untrained GINs are competitive with self-supervised encoders on some datasets and behind on others: Suresh et al. (2021) report them level on MUTAG and DD but behind on NCI1, PROTEINS and IMDB-B under a linear probe. Trivedi et al. (2022b) find untrained GINs competitive on several TU datasets and recommend reporting them, Trivedi et al. (2022a) find that seven methods, including the non-contrastive BYOL and SimSiam, do not clearly outperform untrained encoders, and Khanna et al. (2025) find untrained GNNs and simple graph statistics matching or beating four contrastive methods on standard TU datasets. With large external corpora, pretraining does beat random initialisation (Khanna et al., 2025; Wang et al., 2023). Probes read as much from random as from pretrained graph encoders for some molecular properties (Akhondzadeh et al., 2023), and Graph-JEPA's rank signature is already present at initialisation (Rabby & Auer, 2026). We add this test for Graph-JEPA-style encoders, in a reproduction and the official code, with corrected statistics that bound each null, and evidence that pretraining changes the representation without improving what the probe reads.

Evaluating graph learning. Errica et al. (2020) showed that rigorous model selection reorders graph classifiers and that structure-agnostic baselines are strong on the TU benchmarks. Shchur et al. (2018) showed that node-classification rankings change with the split. Closest to our partition nulls, Mesquita et al. (2020) found that randomised pooling does as well as learned local pooling on graph classification. For pretraining, Hu et al. (2020b) found negative transfer without careful strategies, and Sun et al. (2022) found that self-supervised pretraining of molecular GNNs brings small or no gains under fine-tuning. Our finding concerns the frozen-probe protocol, which most graph self-supervised papers use, and graph JEPAs, which neither covers.

Statistics and planned analyses. Dietterich (1998) and Nadeau & Bengio (2003) showed that tests over resampled training sets are anti-conservative, and Bouckaert & Frank (2004) recommended the corrected repeated cross-validation test we use. Card et al. (2020) argue for reporting the minimum detectable effect, da Costa & Modenesi (2026) find gaps between supervised GNNs dissolving under corrected paired tests, Bouthillier et al. (2021) measured how many sources of variance benchmark comparisons ignore, and pre-registration has been proposed to separate confirmatory from exploratory results in machine learning (Bertinetto et al., 2021; van Miltenburg et al., 2021; Hofman et al., 2023).

5.4 Scope

All benchmarks are small TU datasets with in-dataset pretraining, and the conclusions are about the frozen linear probe that the field reports on them. Fine-tuning, transfer or larger corpora may reward pretraining where this protocol does not, and the official code was not run on its two REDDIT datasets. Our backbone's ℓ_2-normalised targets make its prediction task easy; the official code gives the same verdict without them, but another target design could change it. A non-significant difference is not parity: among the non-significant comparisons the highest upper bound, on MUTAG, leaves room for a gain of up to 6.9 points. We tested published schedules and one short one, not a search over recipes. The untrained arm keeps batch-normalisation statistics at their initial values; recomputing them would be a further control.

6 Conclusion

In this work, we asked whether a never-trained encoder would do as well as a pretrained graph JEPA under the protocol the field uses, and found that on the standard benchmarks it largely does: pretraining adds no detectable accuracy over a random initialisation of the same architecture except on MUTAG, in a reproduction and in the official code. Pretraining does move the representation, without improving what a linear probe reads; the probe reads what the architecture computes, and the most robust partition effect, structural entropy on NCI1, lives there too. The boundary is set by the benchmarks, which cannot resolve gains of the size that separate published methods. The consequence is a cheap change of practice: report every graph self-supervised result against its untrained encoder, with the smallest effect the test could detect, and test architectural claims on the untrained encoder first.

REPRODUCIBILITY STATEMENT

The supplementary material contains the analysis plans with their dated notes and outcomes, every fold-level accuracy record (4200 fold-jobs on our backbone, plus the re-probe and the official-code runs), the code of our backbone, the configuration generators, the scripts that turn records into every table, figure and number in this paper, and the patches applied to the official Graph-JEPA code (Appendix E). Appendix A gives the full configuration.

AI USE STATEMENT

We used a generative AI assistant throughout this work. Of the purposes listed on the submission form, we used it for the four below.

To aid or polish writing: editing for readability and concision, formatting references, suggesting the paper's structure, and proposing its title and keywords.

For retrieval and discovery: searching for, identifying and summarising related work, including targeted searches for prior work that would anticipate our claims (Section 5.3); locating the official Graph-JEPA code and its settings; and checking each reference against its publisher or arXiv record.

For research ideation or execution: developing the conceptual framing (the untrained encoder as an evaluation control, and the random-feature reading in Section 3); proposing and refining hypotheses, including the explanation that pretraining contributes little, which was then tested with a decision rule fixed in advance; designing the methodology and experiments (the analysis plans, their decision rules and controls, the decomposition of Section 3.3 and the power analysis of Section 3.4) and suggesting their parameters; implementing the methods (the partitioners, the training and probing harness, the re-probe and the patches to the official code) and running the experiments; and computing the statistics, drawing the figures, building the supplementary artifacts and interpreting the results.

To draft sections of the paper: drafting the text of every section, the abstract, the captions and the appendix, under our direction and revised after our review.

We did not use it to generate synthetic datasets (all data are public benchmarks, read through standard loaders) or to prove mathematical claims (the paper states no theorems or proofs; its one identity, the decomposition of Section 3.3, is listed above). Translation, dataset cleaning, qualitative or thematic analysis, survey design and transcription do not apply to this work.

We have reviewed all AI-assisted work. The analysis plans for the tests of Section 3.1 and the two held-out tests were recorded before their runs; every number in the paper is generated from fold-level records by scripts included in the supplementary material; and an adversarial review of the code, records and draft, whose findings were each verified before acceptance, produced the corrections that Section 4 reports. We take responsibility for the final content of this work, including text, claims and artifacts produced with the aid of generative AI.

REFERENCES

Mohammad Sadegh Akhondzadeh, Vijay Lingam, and Aleksandar Bojchevski. Probing graph representations. In *Proceedings of the 26th International Conference on Artificial Intelligence and Statistics (AISTATS)*, volume 206 of *Proceedings of Machine Learning Research*, pp. 11630–11649, 2023.

Boshra Ariguib, Mathias Niepert, and Andrei Manolache. Learning the neighborhood: Contrast-free multimodal self-supervised molecular graph pretraining. In *Proceedings of the 43rd International Conference on Machine Learning*, 2026.

Mahmoud Assran, Quentin Duval, Ishan Misra, Piotr Bojanowski, Pascal Vincent, Michael Rabbat, Yann LeCun, and Nicolas Ballas. Self-supervised learning from images with a joint-embedding predictive architecture. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 15619–15629, 2023. doi: 10.1109/CVPR52729.2023.01499.

Luca Bertinetto, João F. Henriques, Samuel Albanie, Michela Paganini, and Gül Varol. Preface. In *NeurIPS 2020 Workshop on Pre-registration in Machine Learning*, volume 148 of *Proceedings of Machine Learning Research*, pp. i–i, 2021. URL https://proceedings.mlr.press/v148/bertinetto21a.html.

Remco R. Bouckaert and Eibe Frank. Evaluating the replicability of significance tests for comparing learning algorithms. In *Advances in Knowledge Discovery and Data Mining (PAKDD)*, pp. 3–12, 2004.

Xavier Bouthillier, Pierre Delaunay, Mirko Bronzi, Assya Trofimov, Brennan Nichyporuk, Justin Szeto, Nazanin Mohammadi Sepahvand, Edward Raff, Kanika Madan, Vikram Voleti, Samira Ebrahimi Kahou, Vincent Michalski, Tal Arbel, Chris Pal, Gaël Varoquaux, and Pascal Vincent. Accounting for variance in machine learning benchmarks. In *Proceedings of Machine Learning and Systems (MLSys)*, volume 3, pp. 747–769, 2021.

Chen Cai and Yusu Wang. A simple yet effective baseline for non-attributed graph classification, 2018. Shorter version at the ICLR 2019 Workshop on Representation Learning on Graphs and Manifolds.

Dallas Card, Peter Henderson, Urvashi Khandelwal, Robin Jia, Kyle Mahowald, and Dan Jurafsky. With little power comes great responsibility. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pp. 9263–9274, 2020. doi: 10.18653/v1/2020.emnlp-main.745.

Tianyu Cui, Yogesh Kumar, Pekka Marttinen, and Samuel Kaski. Deconfounded representation similarity for comparison of neural networks. In *Advances in Neural Information Processing Systems*, 2022.

Kleyton da Costa and Bernardo Modenesi. GraphNetz: Statistical benchmarking of graph neural networks with paired tests and rank aggregation, 2026.

MohammadReza Davari, Stefan Horoi, Amine Natik, Guillaume Lajoie, Guy Wolf, and Eugene Belilovsky. Reliability of CKA as a similarity measure in deep learning. In *International Conference on Learning Representations*, 2023.

Thomas G. Dietterich. Approximate statistical tests for comparing supervised classification learning algorithms. *Neural Computation*, 10(7):1895–1923, 1998. doi: 10.1162/089976698300017197.

Vijay Prakash Dwivedi, Chaitanya K. Joshi, Anh Tuan Luu, Thomas Laurent, Yoshua Bengio, and Xavier Bresson. Benchmarking graph neural networks. *Journal of Machine Learning Research*, 24(43):1–48, 2023.

Federico Errica, Marco Podda, Davide Bacciu, and Alessio Micheli. A fair comparison of graph neural networks for graph classification. In *International Conference on Learning Representations*, 2020.

Claudio Gallicchio and Alessio Micheli. Fast and deep graph neural networks. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pp. 3898–3905, 2020. doi: 10.1609/aaai.v34i04.5803.

Han Gao, Xu Han, Jiaoyang Huang, Jian-Xun Wang, and Li-Ping Liu. PatchGT: Transformer over non-trainable clusters for learning graph representations. In *Proceedings of the First Learning on Graphs Conference*, volume 198 of *Proceedings of Machine Learning Research*, pp. 27:1–27:25. PMLR, 2022. URL https://proceedings.mlr.press/v198/gao22a.html.

Xiaoxin He, Bryan Hooi, Thomas Laurent, Adam Perold, Yann LeCun, and Xavier Bresson. A generalization of ViT/MLP-Mixer to graphs. In *Proceedings of the 40th International Conference on Machine Learning*, volume 202 of *Proceedings of Machine Learning Research*, pp. 12724–12745. PMLR, 2023. URL https://proceedings.mlr.press/v202/he23a.html.

Jake M. Hofman, Angelos Chatzimparmpas, Amit Sharma, Duncan J. Watts, and Jessica Hullman. Pre-registration for predictive modeling, 2023.

Zhenyu Hou, Xiao Liu, Yukuo Cen, Yuxiao Dong, Hongxia Yang, Chunjie Wang, and Jie Tang. GraphMAE: Self-supervised masked graph autoencoders. In *Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pp. 594–604, 2022.

Weihua Hu, Matthias Fey, Marinka Zitnik, Yuxiao Dong, Hongyu Ren, Bowen Liu, Michele Catasta, and Jure Leskovec. Open Graph Benchmark: Datasets for machine learning on graphs. In *Advances in Neural Information Processing Systems*, 2020a.

Weihua Hu, Bowen Liu, Joseph Gomes, Marinka Zitnik, Percy Liang, Vijay Pande, and Jure Leskovec. Strategies for pre-training graph neural networks. In *International Conference on Learning Representations*, 2020b.

Kevin Jarrett, Koray Kavukcuoglu, Marc'Aurelio Ranzato, and Yann LeCun. What is the best multi-stage architecture for object recognition? In *Proceedings of the IEEE 12th International Conference on Computer Vision (ICCV)*, pp. 2146–2153, 2009. doi: 10.1109/ICCV.2009.5459469.

George Karypis and Vipin Kumar. A fast and high quality multilevel scheme for partitioning irregular graphs. *SIAM Journal on Scientific Computing*, 20(1):359–392, 1998.

Smayan Khanna, Doruk Efe Gökmen, Risi Kondor, and Vincenzo Vitelli. Graph contrastive learning versus untrained baselines: The role of dataset size, 2025.

Thomas N. Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. In *International Conference on Learning Representations*, 2017.

Simon Kornblith, Mohammad Norouzi, Honglak Lee, and Geoffrey Hinton. Similarity of neural network representations revisited. In *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pp. 3519–3529, 2019.

Michał Kulczykowski and Rafał Łabędzki. Self-supervised pretraining of molecular graph encoders with LeJEPA, 2026.

Angsheng Li and Yicheng Pan. Structural information and dynamical complexity of networks. *IEEE Transactions on Information Theory*, 62(6):3290–3339, 2016.

Diego Mesquita, Amauri H. Souza, and Samuel Kaski. Rethinking pooling in graph neural networks. In *Advances in Neural Information Processing Systems*, volume 33, pp. 2220–2231, 2020.

Christopher Morris, Nils M. Kriege, Franka Bause, Kristian Kersting, Petra Mutzel, and Marion Neumann. TUDataset: A collection of benchmark datasets for learning with graphs, 2020. ICML 2020 Workshop on Graph Representation Learning and Beyond.

Claude Nadeau and Yoshua Bengio. Inference for the generalization error. *Machine Learning*, 52 (3):239–281, 2003.

Nicolò Navarin, Luca Pasa, Claudio Gallicchio, and Alessandro Sperduti. An untrained neural model for fast and accurate graph classification. In *Artificial Neural Networks and Machine Learning – ICANN 2023*, Lecture Notes in Computer Science, pp. 278–290, 2023. doi: 10.1007/978-3-031-44216-2_23.

Francesco Piccoli, Gabriel Vogel, and Jana M. Weber. Joint embedding predictive architecture for self-supervised pretraining on polymer molecular graphs, 2025.

Gollam Rabby and Sören Auer. When Graph-JEPA learns the wrong thing: Diagnosing and repairing category-conditional collapse, 2026.

Andrew M. Saxe, Pang Wei Koh, Zhenghao Chen, Maneesh Bhand, Bipin Suresh, and Andrew Y. Ng. On random weights and unsupervised feature learning. In *Proceedings of the 28th International Conference on Machine Learning (ICML)*, pp. 1089–1096, 2011.

Oleksandr Shchur, Maximilian Mumme, Aleksandar Bojchevski, and Stephan Günnemann. Pitfalls of graph neural network evaluation, 2018. Relational Representation Learning Workshop (R2L), NeurIPS 2018.

Nino Shervashidze, Pascal Schweitzer, Erik Jan van Leeuwen, Kurt Mehlhorn, and Karsten M. Borgwardt. Weisfeiler-Lehman graph kernels. *Journal of Machine Learning Research*, 12:2539–2561, 2011.

Geri Skenderi, Hang Li, Jiliang Tang, and Marco Cristani. Graph-level representation learning with joint-embedding predictive architectures. *Transactions on Machine Learning Research*, 2025. URL https://openreview.net/forum?id=v47f4DwYZb.

Yu Song, Zhigang Hua, Yan Xie, Jingzhe Liu, Bo Long, and Hui Liu. GSTBench: A benchmark study on the transferability of graph self-supervised learning. In *Proceedings of the 34th ACM International Conference on Information and Knowledge Management (CIKM '25)*, pp. 2761–2770, 2025. doi: 10.1145/3746252.3761422.

Fan-Yun Sun, Jordan Hoffmann, Vikas Verma, and Jian Tang. InfoGraph: Unsupervised and semi-supervised graph-level representation learning via mutual information maximization. In *International Conference on Learning Representations*, 2020.

Ruoxi Sun, Hanjun Dai, and Adams Wei Yu. Does GNN pretraining help molecular representation? In *Advances in Neural Information Processing Systems*, volume 35, pp. 12096–12109, 2022. doi: 10.52202/068431-0879.

Susheel Suresh, Pan Li, Cong Hao, and Jennifer Neville. Adversarial graph augmentation to improve graph contrastive learning. In *Advances in Neural Information Processing Systems*, volume 34, pp. 15920–15933, 2021.

Shantanu Thakoor, Corentin Tallec, Mohammad Gheshlaghi Azar, Mehdi Azabou, Eva L. Dyer, Rémi Munos, Petar Veličković, and Michal Valko. Large-scale representation learning on graphs via bootstrapping. In *International Conference on Learning Representations*, 2022.

Puja Trivedi, Ekdeep Singh Lubana, Mark Heimann, Danai Koutra, and Jayaraman J. Thiagarajan. Analyzing data-centric properties for graph contrastive learning. In *Advances in Neural Information Processing Systems*, volume 35, pp. 14030–14043, 2022a. doi: 10.52202/068431-1020.

Puja Trivedi, Ekdeep Singh Lubana, Yujun Yan, Yaoqing Yang, and Danai Koutra. Augmentations in graph contrastive learning: Current methodological flaws & towards better practices. In *Proceedings of the ACM Web Conference 2022 (WWW '22)*, pp. 1538–1549, 2022b. doi: 10.1145/3485447.3512200.

Emiel van Miltenburg, Chris van der Lee, and Emiel Krahmer. Preregistering NLP research. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL-HLT)*, pp. 613–623, 2021. doi: 10.18653/v1/2021.naacl-main.51.

Petar Veličković, William Fedus, William L. Hamilton, Pietro Liò, Yoshua Bengio, and R. Devon Hjelm. Deep graph infomax. In *International Conference on Learning Representations*, 2019.

Hanchen Wang, Jean Kaddour, Shengchao Liu, Jian Tang, Joan Lasenby, and Qi Liu. Evaluating self-supervised learning for molecular graph embeddings. In *Advances in Neural Information Processing Systems (Datasets and Benchmarks Track)*, volume 36, pp. 68028–68060, 2023. doi: 10.52202/075280-2977.

John Wieting and Douwe Kiela. No training required: Exploring random encoders for sentence classification. In *International Conference on Learning Representations*, 2019.

Junran Wu, Shangzhe Li, Jianhao Li, Yicheng Pan, and Ke Xu. A simple yet effective method for graph classification. In *Proceedings of the Thirty-First International Joint Conference on Artificial Intelligence (IJCAI)*, pp. 3580–3586, 2022. doi: 10.24963/ijcai.2022/497.

Junran Wu, Xueyuan Chen, Bowen Shi, Shangzhe Li, and Ke Xu. SEGA: Structural entropy guided anchor view for graph contrastive learning. In *Proceedings of the 40th International Conference on Machine Learning*, 2023.

Keyulu Xu, Weihua Hu, Jure Leskovec, and Stefanie Jegelka. How powerful are graph neural networks? In *International Conference on Learning Representations*, 2019. URL https://openreview.net/forum?id=ryGs6iA5Km.

Ruichen Xu, Jingxiang Qu, Wenhan Gao, Jiaxing Zhang, Linsey Pang, Ravid Shwartz-Ziv, Yann LeCun, and Yuefan Deng. HP-JEPA: Hierarchical partitioning for multi-resolution graph joint-embedding predictive learning, 2026.

Yuning You, Tianlong Chen, Yongduo Sui, Ting Chen, Zhangyang Wang, and Yang Shen. Graph contrastive learning with augmentations. In *Advances in Neural Information Processing Systems*, volume 33, pp. 5812–5823, 2020.

Daniele Zambon, Cesare Alippi, and Lorenzo Livi. Graph random neural features for distance-preserving graph representations. In *Proceedings of the 37th International Conference on Machine Learning*, volume 119 of *Proceedings of Machine Learning Research*, pp. 10968–10977, 2020.

Junru Zhu, Yixin Yang, Xiaoqing Ding, and Ruoyu Qi. Partition-matched evaluation of community features under distribution shift in android malware function-call graphs, 2026.

A Backbone and configuration

All arms of our backbone share one configuration and differ only in the fields each comparison names. Patches are the clusters of a partition expanded by one hop; each patch is encoded by a two-layer GINE network of width 512 with random-walk positional features of dimension 20, and a four-layer transformer with eight heads runs over the patch tokens of every rung. The target encoder is an exponential moving average of the context encoder, with momentum rising from 0.996 to 0.998. Targets are ℓ_2-normalised and mapped to the Lorentz hyperboloid; the loss is a smooth-ℓ_1 distance. Training uses Adam at a constant learning rate of 5×10^{-4}, gradient clipping at 1, batches of 128 graphs and 30 epochs; IMDB-MULTI uses batches of 32 and 5 epochs in every arm. The ladder has five rungs: cluster counts $[2, 4, 6, 8, 12]$, and $[2, 3, 4, 5, 6]$ on IMDB-MULTI. Node attributes are standardised with statistics of the training folds. The probe is an ℓ_2-regularised logistic regression fitted on the training folds of each split; our read-out uses the finest rung with regularisation strength 1 (inverse strength $C = 1$), and the published read-out combines all rungs with weights $0.75 \cdot \text{uniform} + 0.25 \cdot \text{learned}$ at strength 0.01. Seeds are 12, 21, 35, 42 and 95; each seed draws its own stratified ten-fold split. DD's largest graphs need micro-batching of the attention for the node-level rung only; this is exact at feature extraction.

B Partitions and ladders

METIS computes a balanced K-way cut at each count of the ladder (Karypis & Kumar, 1998). *SE fixed-K* minimises two-level structural entropy (Li & Pan, 2016) directly over partitions with exactly K clusters. *SE levels* runs a Louvain-style minimisation of structural entropy that produces a hierarchy of nested partitions, keeps the levels that are strictly finer than their predecessor, anchors the coarse end at $K = 2$ by an SE-greedy merge, appends the node-level partition, and fits the result to the ladder's five slots by even subsampling or by SE-greedy merges at geometrically spaced counts. The ladder is computed once per graph and seed and cached. *Shape-matched random* trees keep, for every graph, the SE ladder's number of rungs and cluster sizes and assign vertices to clusters at random, nested across rungs.

C The two instrument fixes

The backbone as first reproduced read raw node attributes, and its Lorentz target applies $\cosh$ to the mean of the target encoder's output. When attributes are unnormalised, the target encoder's output norm grows during training, and $\cosh$ overflows in single precision once its argument passes 89.4; the loss becomes infinite, but the target branch carries no gradient and the smooth-ℓ_1 loss has bounded slope, so the gradients stay finite and training continues silently. Standardising the

inputs and ℓ_2-normalising the target before the hyperbolic map removes both failure modes. Every comparison in this paper uses both fixes. Table 4 shows how far re-measuring the same arms moved them.

Table 4: Accuracy of the same arms before and after the two instrument fixes (points; t_c in brackets), paired over the shared seeds and folds.

dataset	arm	pre-fix	corrected	corrected − pre-fix
MUTAG	SE levels	91.48	90.72	−0.76 (−0.59)
MUTAG	SE fixed-K	89.44	87.41	−2.04 (−1.32)
MUTAG	METIS	87.44	86.08	−1.36 (−0.73)
ENZYMES	SE levels	52.47	59.70	**+7.23** (3.25)
ENZYMES	SE fixed-K	52.97	56.53	+3.57 (1.55)
ENZYMES	METIS	54.53	58.37	+3.83 (1.53)
NCI1	SE levels	74.62	74.16	−0.46 (−0.48)
NCI1	SE fixed-K	72.76	73.84	+1.08 (1.26)
NCI1	METIS	70.46	71.75	+1.29 (1.45)
PROTEINS	SE fixed-K	76.14	75.85	−0.29 (−0.29)
PROTEINS	METIS	76.28	76.61	+0.32 (0.31)
IMDB-BINARY	SE fixed-K	72.70	72.66	−0.04 (−0.03)
IMDB-BINARY	METIS	72.80	72.66	−0.14 (−0.11)
IMDB-MULTI	SE fixed-K	49.76	49.93	+0.17 (0.20)
IMDB-MULTI	METIS	49.67	50.35	+0.68 (0.85)

D FULL TABLES

Table 5: The partition with and without pretraining (exploratory): structural-entropy minus METIS partitions, paired over 5 seeds × 10 folds, in points with t_c in brackets; bold: $|t_c| > 2.01$. Never-trained arms share seeds and folds with each other; trained arms are the like-for-like ones of Table 3.

	never-trained encoders		trained encoders	
dataset	SE fixed-K − METIS	SE levels − METIS	SE fixed-K − METIS	SE levels − METIS
MUTAG	−0.32 (−0.14)	+1.49 (0.91)	+1.33 (0.48)	**+4.64** (2.28)
ENZYMES	−1.10 (−0.54)	+1.03 (0.66)	−1.83 (−1.03)	+1.33 (0.77)
NCI1	**+1.88** (3.31)	**+1.99** (3.36)	**+2.10** (2.27)	**+2.41** (3.30)
PROTEINS	−0.25 (−0.31)	−0.43 (−0.57)	+0.18 (0.19)	+0.18 (0.22)
IMDB-B	−0.30 (−0.38)	−0.42 (−0.71)	0.00 (0.00)	+0.52 (0.52)
IMDB-M	−0.01 (−0.02)	+0.27 (0.60)	−0.31 (−0.40)	−0.36 (−0.53)

Table 6: Table 3 with every statistic: differences in points, statistic in brackets (naive t in the first row, corrected t_c elsewhere); bold: $|t| > 2.01$.

comparison	MUTAG	ENZYMES	NCI1	PROTEINS	IMDB-B	IMDB-M	ahead	sig.
SE levels − fixed-K, naive t	**+3.31** (4.55)	**+3.17** (3.97)	+0.32 (1.07)	+0.59 (1.93)	+0.52 (1.18)	+0.56 (1.76)	6/6	2/6
same, corrected t_c	+3.31 (1.78)	+3.17 (1.55)	+0.32 (0.42)	+0.59 (0.75)	+0.52 (0.46)	+0.56 (0.69)	6/6	0/6
same, like-for-like baseline	+3.31 (1.78)	+3.17 (1.55)	+0.32 (0.42)	0.00 (0.00)	+0.52 (0.46)	−0.05 (−0.06)	4/6	0/6
SE levels − METIS, like-for-like	**+4.64** (2.28)	+1.33 (0.77)	**+2.41** (3.30)	+0.18 (0.22)	+0.52 (0.52)	−0.36 (−0.53)	5/6	2/6
SE levels − shape-matched random tree	+1.16 (1.12)	+0.23 (0.16)	+0.68 (0.87)	−0.38 (−0.60)	−0.18 (−0.21)	−0.53 (−0.81)	3/6	0/6
SE levels, trained − untrained	**+3.49** (2.15)	+1.03 (0.45)	+0.37 (0.62)	−0.43 (−0.54)	+0.88 (1.05)	−0.35 (−0.47)	4/6	1/6

E THE OFFICIAL GRAPH-JEPA RUNS

We ran the official implementation, as released by its authors, with the configuration it publishes for each dataset, and made six changes, none of which touches the model, the loss, the optimiser, the schedule or the probe. (i) Two variables are initialised before the training loop, because upstream logs values that exist only after one epoch; the trained arm is unaffected. (ii) Each run writes its own

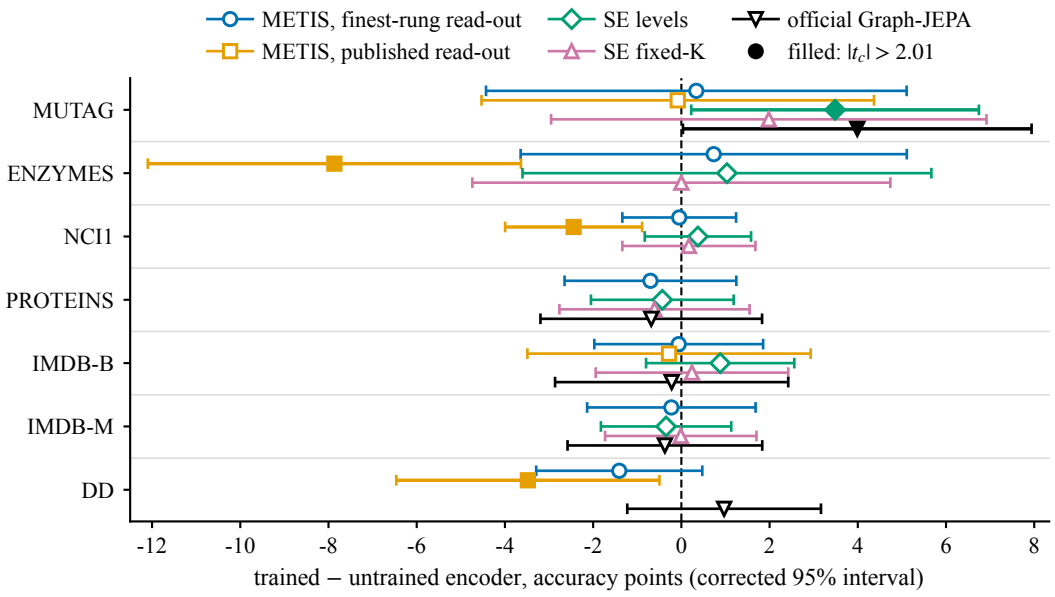

Figure 5: Every comparison of Figure 1a as a difference: trained minus untrained accuracy with the corrected 95% interval of Equation 1, paired over 5 seeds × 10 folds. Colours and shapes mark the five settings; filled markers are significant.

Table 7: Table 1 with every statistic: trained − untrained differences in points, t_c in brackets; bold: $|t_c| > 2.01$.

	METIS, finest-rung read-out			METIS	SE levels	SE fixed-K
dataset	untrained	trained	trained − untr.	published read-out	trained − untr.	trained − untr.
MUTAG	85.74	86.08	+0.34 (0.14)	−0.08 (−0.04)	**+3.49** (2.15)	+1.98 (0.81)
ENZYMES	57.63	58.37	+0.73 (0.34)	**−7.87** (−3.74)	+1.03 (0.45)	0.00 (0.00)
NCI1	71.80	71.75	−0.05 (−0.08)	**−2.44** (−3.16)	+0.37 (0.62)	+0.17 (0.23)
PROTEINS	77.31	76.61	−0.70 (−0.72)	–	−0.43 (−0.54)	−0.61 (−0.57)
IMDB-BINARY	72.72	72.66	−0.06 (−0.06)	−0.28 (−0.18)	+0.88 (1.05)	+0.24 (0.22)
IMDB-MULTI	50.57	50.35	−0.23 (−0.24)	–	−0.35 (−0.47)	−0.01 (−0.02)
DD	78.27	76.86	−1.41 (−1.50)	**−3.48** (−2.34)	–	–
trained significantly better			0/7	0/5	1/6	0/6

log file, because upstream names logs by timestamp and two arms started in the same second shared one. (iii) Import-only packages were installed. (iv) Graph sorting uses `torch.sort`, because the cluster's build of a PyTorch Geometric extension has no CPU kernel for the operation it would otherwise call. (v) The GPU is selected by the environment, because two configurations name a second device. (vi) Data loading uses two worker processes instead of eight. METIS is provided through the `pymetis` binding. Upstream repeats one fixed ten-fold split over five seeds and prints one accuracy per fold; we pair the arms by seed and fold. On DD, to finish before the cut-off fixed in advance, the five seeds of the trained arm ran as separate processes through upstream's own single-seed option; upstream seeds each run at its start, so this computes the same runs. For orientation only, since the comparison is unpaired and protocols differ: in the table collected by Xu et al. (2026), the never-trained official encoder would rank above 9 of 10 self-supervised methods on PROTEINS, 6 of 9 on DD, 7 of 11 on IMDB-B and 7 of 10 on IMDB-M, and above only 1 of 10 on MUTAG, where pretraining helps. One operational fault occurred and is recorded in a dated note to the plan: a wrapper script edited while it ran re-executed DD's first trained run after it had finished, and upstream's log, opened for writing, was overwritten. The run was repeated in full with the same seed. The first eight folds of the lost log, recovered from an earlier copy, are archived in the supplementary material and are not used. As a determinism check they match the repeated run

Table 8: The partition comparison. Accuracy (%) ± standard deviation over the five seed means, and paired differences in points with t_c in brackets. Bold: $|t_c| > 2.01$. These are the original arms; Table 3 gives the like-for-like re-runs for PROTEINS and IMDB-M.

dataset	SE levels	SE fixed-K	METIS	levels − fixed-K	levels − METIS	fixed-K − METIS
MUTAG	90.72 ± 1.10	87.41 ± 0.30	86.08 ± 0.66	+3.31 (1.78)	**+4.64** (2.28)	+1.33 (0.48)
ENZYMES	59.70 ± 1.36	56.53 ± 1.90	58.37 ± 1.22	+3.17 (1.55)	+1.33 (0.77)	−1.83 (−1.03)
NCI1	74.16 ± 0.43	73.84 ± 0.42	71.75 ± 0.59	+0.32 (0.42)	**+2.41** (3.30)	**+2.10** (2.27)
PROTEINS	76.44 ± 0.36	75.85 ± 0.54	76.61 ± 0.73	+0.59 (0.75)	−0.16 (−0.17)	−0.75 (−1.14)
IMDB-BINARY	73.18 ± 0.50	72.66 ± 1.10	72.66 ± 0.67	+0.52 (0.46)	+0.52 (0.52)	0.00 (0.00)
IMDB-MULTI	50.49 ± 0.55	49.93 ± 0.79	50.35 ± 0.37	+0.56 (0.69)	+0.15 (0.21)	−0.41 (−0.48)

Table 9: Ablations of the SE ladder: removing the node-level rung, and replacing the SE hierarchy by a shape-matched random tree (points; t_c in brackets).

dataset	SE levels	no node rung	shape-matched random	levels − no-node-rung	levels − random
MUTAG	90.72	88.71	89.56	+2.01 (1.19)	+1.16 (1.12)
ENZYMES	59.70	57.83	59.47	+1.87 (1.00)	+0.23 (0.16)
NCI1	74.16	72.99	73.48	+1.17 (1.75)	+0.68 (0.87)
PROTEINS	76.44	76.46	76.82	−0.02 (−0.02)	−0.38 (−0.60)
IMDB-BINARY	73.18	72.54	73.36	+0.64 (0.67)	−0.18 (−0.21)
IMDB-MULTI	50.49	49.99	51.03	+0.51 (0.73)	−0.53 (−0.81)

exactly on 2 of 8 folds and differ by up to 3.39 points on the others: upstream training on the GPU is not bit-reproducible, which is one more reason to pair arms and to report corrected intervals.

F PLANNED TESTS AND OUTCOMES

Each test below was specified, with its decision rule, before its first run, and each outcome is the one its rule gives:

- *SE-block attention* (NCI1; SE blocks must beat METIS and random blocks): not met.
- *Node-level rung on held-out DD* (must beat METIS with $t_c > 2.01$): not met.
- *Does pretraining help* (METIS arm, seven datasets; helps if ≥ 4 wins): barely helps, 0 of 7.
- *Published read-out* (re-probe of saved models; same rule over the datasets that reload exactly): barely helps, 0 of 5, untrained significantly better on 3.
- *Other partitioners* (SE levels and SE fixed-K, six datasets each; helps if ≥ 4): barely helps, 1 and 0 of six.
- *Official implementation* (five datasets; reproduction within three points first; helps if ≥ 3 reproduced wins): barely helps, 1 of 5 (all 5 reproduced, within 0.34 points) .

Table 10: Read-out-matched comparison: both baselines with a node-level rung appended, so that every arm reads the same finest rung.

dataset	SE levels	fixed-K + node	METIS + node	levels − fixed-K + node	levels − METIS + node
MUTAG	90.72	89.35	89.64	+1.37 (1.14)	+1.08 (0.65)
ENZYMES	59.70	59.90	60.83	−0.20 (−0.11)	−1.13 (−0.62)
NCI1	74.16	74.45	72.98	−0.29 (−0.40)	+1.18 (1.60)
PROTEINS	76.44	76.48	76.66	−0.04 (−0.05)	−0.22 (−0.30)
IMDB-BINARY	73.18	72.98	73.62	+0.20 (0.23)	−0.44 (−0.58)
IMDB-MULTI	50.49	49.96	50.01	+0.53 (0.66)	+0.48 (0.56)

Table 11: Held-out tests specified in advance: SE-block attention on NCI1 (against METIS blocks, random blocks and dense attention) and a node-level rung on DD.

pre-registered comparison	dataset	difference (t_c)
SE-block − METIS-block	NCI1	−0.25 (−0.52)
SE-block − random-block	NCI1	−0.17 (−0.40)
SE-block − dense	NCI1	−0.24 (−0.45)
METIS + node rung − METIS	DD	+0.29 (0.31)

Table 12: The first untrained-encoder test (METIS arm, our read-out), with the one-epoch arm, which the plan reported as descriptive.

dataset	untrained	1 epoch	trained	trained − untrained	1 epoch − untrained
MUTAG	85.74	84.68	86.08	+0.34 (0.14)	−1.06 (−0.79)
ENZYMES	57.63	60.13	58.37	+0.73 (0.34)	+2.50 (1.04)
NCI1	71.80	73.03	71.75	−0.05 (−0.08)	**+1.24** (2.27)
PROTEINS	77.31	76.73	76.61	−0.70 (−0.72)	−0.57 (−0.72)
IMDB-BINARY	72.72	72.58	72.66	−0.06 (−0.06)	−0.14 (−0.14)
IMDB-MULTI	50.57	50.03	50.35	−0.23 (−0.24)	−0.55 (−0.55)
DD	78.27	77.32	76.86	−1.41 (−1.50)	−0.95 (−0.89)

Table 13: Published graph-level accuracies collected in HP-JEPA's comparison table, for orientation only: they are unpaired with ours and were obtained under other instruments.

	model	PROTEINS	MUTAG	DD	IMDB-BINARY	IMDB-MULTI
Supervised	F-GIN (supervised)	72.39 ± 2.76	90.41 ± 4.61	74.87 ± 3.56	71.83 ± 1.93	48.46 ± 2.31
Contrastive	InfoGraph	72.57 ± 0.65	87.71 ± 1.77	75.23 ± 0.39	71.11 ± 0.88	48.66 ± 0.67
	GraphCL	72.86 ± 1.01	88.29 ± 1.31	74.70 ± 0.70	70.80 ± 0.77	48.49 ± 0.63
	MVGRL	–	–	–	74.20 ± 0.70	51.20 ± 0.50
	AD-GCL-FIX	73.59 ± 0.65	89.25 ± 1.45	74.49 ± 0.52	71.57 ± 1.01	49.04 ± 0.53
	AD-GCL-OPT	73.81 ± 0.46	89.70 ± 1.03	75.10 ± 0.39	72.33 ± 0.56	49.89 ± 0.66
Generative	GraphMAE	75.30 ± 0.39	88.19 ± 1.26	74.27 ± 1.07	75.52 ± 0.66	51.63 ± 0.52
	S2GAE	76.37 ± 0.43	88.26 ± 0.76	–	75.76 ± 0.62	51.79 ± 0.36
Self-predictive	BGRL	70.99 ± 3.86	74.99 ± 8.83	71.52 ± 2.97	0.50 ± 0.00	0.33 ± 0.00
	LaGraph	75.20 ± 0.40	90.20 ± 1.10	78.10 ± 0.40	73.70 ± 0.90	–
JEPA	Graph-JEPA	75.67 ± 3.78	91.25 ± 5.75	78.64 ± 2.35	73.68 ± 3.24	50.69 ± 2.91
	HP-JEPA	76.97 ± 0.50	91.70 ± 5.31	79.20 ± 0.38	74.08 ± 0.40	50.51 ± 1.81
This work	SE levels (ours)	76.44 ± 0.36	90.72 ± 1.10	–	73.18 ± 0.50	50.49 ± 0.55
	SE shape + random (ours)	76.82 ± 0.43	89.56 ± 0.99	–	73.36 ± 0.64	51.03 ± 0.56
	SE fixed-K (ours)	75.85 ± 0.54	87.41 ± 0.30	77.15 ± 0.77	72.66 ± 1.10	49.93 ± 0.79
	METIS (ours)	76.61 ± 0.73	86.08 ± 0.66	76.86 ± 0.76	72.66 ± 0.67	50.35 ± 0.37