ANYGRAM: ROTATION-INVARIANT GLOBAL OPERATORS FOR FULLY-INDUCTIVE NODE CLASSIFICATION

Anonymous authors
Paper under double-blind review

ABSTRACT

A fully-inductive node classifier is trained on some graphs and then applied, frozen and without gradient steps, to a new graph whose features, label space, and structure it did not see during training. Existing models combine a fixed bank of local propagation operators, a closed-form least-squares readout on the labeled support nodes, and a head, learned on the source graphs, that weighs the operators. Two parts of this design remain open: the operators see only local neighborhoods, and the choice of how to read the bank, which depends on the new graph, is learned in advance on other graphs. In this paper, we ask which parts of a fully-inductive classifier should be learned across graphs and which should be decided on the target graph from its support labels. We introduce **AnyGRAM**, which learns encoders whose inputs are invariant to rotations of the random feature projection and uses them to add global covariance operators to the local bank, applied in factored form without forming a node-by-node matrix. The ridge regularization of each branch and the branch used on each graph are selected by cross-validation on the support set of the target graph. Trained on a single source graph and applied frozen to 26 target graphs, AnyGRAM reaches 71.98% average accuracy, 6.16 percentage points above the strongest zero-shot baseline and above end-to-end models trained on each target graph. Our empirical results show that the covariance branch family and the decisions made on the target graph contribute to this gain, and accuracy changes little with the choice or number of source graphs.

1 INTRODUCTION

Graph foundation models classify nodes in graphs they were not trained on (Mao et al., 2024). Unlike images or text, every new graph brings its own input space. Its node features can differ in dimension, meaning, and range; target label sets can differ in size and meaning, and the topological relations between labels can range from homophilous citation networks to heterophilous web and road graphs. The fully-inductive setting of Zhao et al. (2025) asks a model trained on a few source graphs to handle all of these differences at once, with frozen parameters, no gradient steps on the new graph, and only the labeled support nodes of that graph as guidance.

Recent works have taken important steps toward this goal. GraphAny (Zhao et al., 2025) propagates the features with a fixed bank of low-pass and high-pass operators, maps each propagated representation to class logits with a closed-form least-squares fit on the support labels, and learns an attention head that combines the resulting predictions. Because no learned parameter depends on the feature dimension or the number of classes, the same model can be applied to graphs with any feature dimension and label set. Finkelshtein et al. (2025) derive the node, feature, and label symmetries that such a model should respect and build triple-symmetry networks that satisfy them. These works show that a single frozen model can compete with networks trained on each target graph. Nonetheless, two parts of their design remain open. First, their operators are local: a prediction depends on a bounded neighborhood, and distant nodes whose features co-vary across the graph influence it only through repeated smoothing. Second, the choice of how to use the operator bank on a new graph, which depends on properties of that graph such as its homophily, is made by a head learned on the source graphs before the new graph is seen.

This raises the question of which parts of a fully-inductive classifier should be learned across graphs, and which should be decided on the target graph itself? We argue for a division. A component

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.

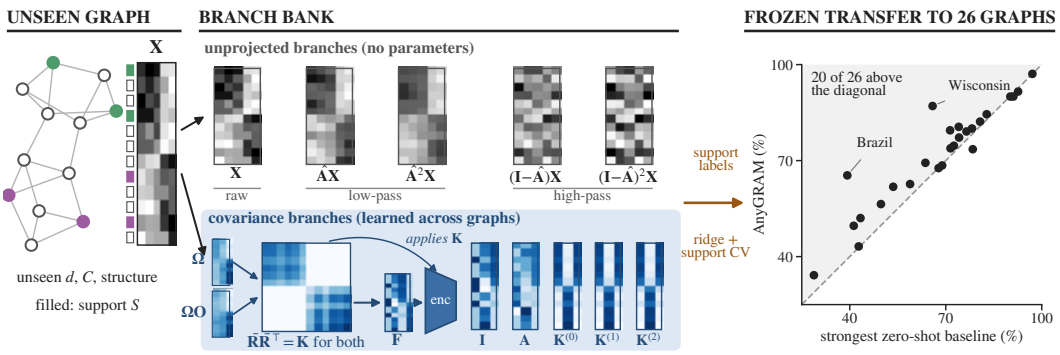


Figure 1: **AnyGRAM learns its covariance branches across graphs and decides the readout on each target graph.** Unprojected branches (gray) are the raw features and their low-pass and high-pass propagations. Covariance branches (blue) are learned across source graphs: a projection and its rotation, $\mathbf{\Omega}$ and $\mathbf{\Omega O}$, give different projected features $\bar{\mathbf{R}}$ but the same covariance operator $\mathbf{K}$, and the encoders read only rotation-invariant inputs $\mathbf{F}$ and apply $\mathbf{K}$ (matrices computed on a toy graph). On the target graph (orange), each branch is read out by ridge regression on the support labels, and support cross-validation selects the branch. Right: trained on Cora and applied frozen, AnyGRAM is above the strongest zero-shot baseline on 20 of the 26 target graphs (Table 1).

learned across graphs should read inputs that mean the same thing on every graph, and choices that depend on the target graph can be estimated from its support labels, the only labels available at inference. The first half of this division determines which global operators a learned model can use. Covariance operators built from randomly projected node features (Eliasof et al., 2026; Sihag et al., 2022) are a natural candidate, because they relate every pair of nodes through their features, their size does not depend on the feature dimension, and they are invariant *in distribution* to permutations of the features. However, the projected features themselves live in an arbitrary basis, so a learned encoder that reads them sees its input in a new, arbitrary basis whenever the projection is redrawn. The second half turns the readout into a model selection problem on the support set, as generalized cross-validation does for ridge regression (Golub et al., 1979) and ensemble selection does for an ensemble of models (Caruana et al., 2004).

Our approach. We introduce AnyGRAM, a fully-inductive node classifier built on this division (Figure 1). Its node embedding bank has ten branches. Five are the unprojected branches of GraphAny, which have no parameters, and five, which we call covariance branches, are learned encoders that read only quantities invariant to rotations of the projection: random-walk structural encodings, propagated feature norms, similarities to landmark nodes, and label-propagation statistics. Three covariance branches also propagate with global covariance operators $\mathbf{K}^{(p)} = \bar{\mathbf{R}}^{(p)}\bar{\mathbf{R}}^{(p)\top}$ of the centered projected features $\bar{\mathbf{R}}^{(p)}$, applied as attention with queries and keys fixed to $\bar{\mathbf{R}}^{(p)}$ and evaluated as $\bar{\mathbf{R}}^{(p)}(\bar{\mathbf{R}}^{(p)\top}\mathbf{V})$ without forming an $n \times n$ matrix; the encoder inputs do not change under a rotation of the projection. Each branch is classified by ridge regression on the support labels, and one eigendecomposition gives the solutions for a grid of regularization strengths, the choice among them by generalized cross-validation (GCV), and exact leave-one-out predictions. A learned soft fusion trains the encoders through the ridge solve; at inference, cross-validation on the support set (support CV) picks the branch for each graph, and a query node switches to another branch when its most similar support nodes favor it by a margin.

Trained on Cora alone and applied frozen to 26 graphs, AnyGRAM improves on the average accuracy of the strongest zero-shot baseline by about six points and exceeds end-to-end models trained on each target graph. The covariance branches are selected on a subset of the graphs and bring their largest gains there, support-validated selection outperforms the learned readouts we evaluate on the same branch predictions, and accuracy changes little when the source graph or the number of source graphs changes.

Main contributions. The contributions of this paper are as follows:

- We formulate the design of a fully-inductive node classifier as a division between components learned across graphs, which read inputs that mean the same thing on every graph, and decisions made on the target graph from its support labels.

- We introduce AnyGRAM, which makes global covariance operators usable inside a learned fully-inductive model through encoders that read only rotation-invariant quantities and an attention with fixed queries and keys, and which selects the ridge regularization and the branch for each target graph by cross-validation on its support set.

We conduct an extensive empirical study on 26 graphs, showing that AnyGRAM trained on a single source graph outperforms zero-shot baselines and end-to-end models on average, that the global operators and the decisions made on the target graph both contribute, and that the results change little with the choice and number of source graphs.

2 Related Work

Fully-inductive node classification. Graph foundation models built on text representations transfer well across text-attributed graphs (Zhao et al., 2023; Liu et al., 2024; Tang et al., 2024) but do not apply when attributes are numerical or categorical. Models that align non-textual feature spaces through projections, prompts, or perceiver encoders (Yu et al., 2024; Zhao et al., 2024; Lachi et al., 2025; Fang et al., 2023) generally still fine-tune on each dataset, while AnyGraph and OpenGraph (Xia & Huang, 2026; Xia et al., 2024) transfer without tuning but are trained with link-level objectives. Zhao et al. (2025) define the fully-inductive setting and solve it with parameter-free propagation, a least-squares readout, and a learned attention over the resulting predictions. Finkelshtein et al. (2025) derive the symmetries of this setting from first principles and show that GraphAny covers a restricted part of the resulting design space. Lee et al. (2026) learn feature-equivariant maps in a view space induced by the graph structure, and tabular foundation models have also been adapted to graphs (Eremeev et al., 2025; 2026; Hayler et al., 2025). These works design the invariant or equivariant layers that make a frozen model applicable to a new graph. In this work, we keep an invariant construction, extend the operator bank with global operators, and ask which decisions such a model should make on the target graph instead of learning them in advance.

Random projections, covariance operators, and global attention. ALL-IN (Eliasof et al., 2026) bridges feature spaces with an isotropic Gaussian projection to a fixed width and builds covariance operators that are invariant in distribution to feature permutations and, in expectation, to orthogonal changes of the feature basis, building on random node features (Sato et al., 2021; Abboud et al., 2021) and covariance matrices as graph shift operators (Sihag et al., 2022). ALL-IN feeds the projected features directly to its downstream layers, so every redraw of the projection presents the encoder input in a new, arbitrarily rotated basis; AnyGRAM instead supplies its learned encoders with inputs invariant to this rotation. Graph transformers provide dense attention between all node pairs (Rampášek et al., 2022), and scalable variants use linear attention (Katharopoulos et al., 2020; Wu et al., 2022; 2023; Deng et al., 2024); built on projected features, they would inherit the same rotation. STAGE (Shen et al., 2025) shows that statistical dependencies between attributes transfer across attribute domains where raw values do not. Differently, our encoders read only functionals of the covariance operator of the projected features, and their global term is an attention block, as in GPS (Rampášek et al., 2022), whose queries and keys are fixed to the centered projection, so that the attention matrix is exactly the covariance operator and only the values are learned.

Operator banks, closed-form readouts, and model selection. Banks of fixed propagation operators combined by a learned head predate graph foundation models (Frasca et al., 2020; Wu et al., 2019), with high-pass and identity channels added to handle heterophily (Zhu et al., 2020; Bo et al., 2021; Chien et al., 2021), and mixtures of experts learn such combinations per input (Jacobs et al., 1991; Shazeer et al., 2017). Differentiable closed-form ridge solvers have been used as meta-learning heads for few-shot classification (Bertinetto et al., 2019; Lee et al., 2019), and GraphAny uses a least-squares readout in the fully-inductive setting. We adopt a ridge readout that is trained through, choose its regularization per branch and per graph by generalized cross-validation (Hoerl & Kennard, 1970; Golub et al., 1979), and replace the learned combination at inference with a selection validated on the support set, in the spirit of ensemble selection from a library of models (Caruana et al., 2004).

3 Learning Invariant Branches and Selecting on Target Graphs

Problem setting. We consider node classification in the fully-inductive setting (Zhao et al., 2025). A graph $G = (\mathcal{V}, \mathcal{E}, \mathbf{X})$ has $n = |\mathcal{V}|$ nodes and node features $\mathbf{X} \in \mathbb{R}^{n \times d}$. On each graph we observe

a support set $S \subset \mathcal{V}$ of $m = |S|$ labeled nodes with one-hot labels $\mathbf{Y}_S \in \{0,1\}^{m \times C}$, and we predict the labels of a disjoint query set Q. A model is trained on a pool of source graphs and then applied to target graphs with frozen parameters and no gradient steps, where d, C, and the graph structure may all differ from those seen in training. Therefore, no learned parameter may have a shape that depends on d or C.

Overview. Following the division of Section 1, AnyGRAM builds a bank of B branches, each a node representation $\mathbf{H}_b \in \mathbb{R}^{n \times D_b}$, of which five are parameter-free and five are learned encoders of rotation-invariant inputs (Section 3.1). It classifies each branch by ridge regression with a regularization chosen on the support set (Section 3.2), trains the learned branches across source graphs, and selects among the branches on the target graph (Section 3.3), as summarized in Figure 1.

3.1 A Bank of Local and Global Branches

The bank combines the local operators of GraphAny, which need no learning, with learned branches that add global covariance operators. We use $B = 10$ branches and propagation depth $k = 2$. Let $\mathbf{A}$ be the adjacency matrix, $\hat{\mathbf{A}}$ its symmetric normalization without self-loops, and $\hat{\mathbf{A}}_{\mathrm{sl}}$ the symmetric normalization with self-loops.

Unprojected branches. The first five branches are the raw and propagated features used by GraphAny (Zhao et al., 2025),

$$\mathbf{X}, \quad \hat{\mathbf{A}}\mathbf{X}, \quad \hat{\mathbf{A}}^2\mathbf{X}, \quad (\mathbf{I} - \hat{\mathbf{A}})\mathbf{X}, \quad (\mathbf{I} - \hat{\mathbf{A}})^2\mathbf{X}. \tag{1}$$

The raw features $\mathbf{X}$ use no propagation, $\hat{\mathbf{A}}\mathbf{X}$ and $\hat{\mathbf{A}}^2\mathbf{X}$ are low-pass channels suited to homophilous graphs, and the last two are high-pass channels suited to heterophilous graphs (Zhu et al., 2020; Bo et al., 2021). They have no parameters, so their width is $D_b = d$.

Global covariance operators. To relate nodes that are far apart in the graph, we use an operator that compares nodes through their features and is defined for any feature dimension. Following ALL-IN (Eliasof et al., 2026), we draw a random projection $\mathbf{\Omega} \in \mathbb{R}^{d \times h}$ with i.i.d. standard normal entries, set $\mathbf{R}^{(0)} = \mathbf{X}\mathbf{\Omega}$, propagate $\mathbf{R}^{(p)} = \hat{\mathbf{A}}_{\mathrm{sl}}\mathbf{R}^{(p-1)}$, and center each $\mathbf{R}^{(p)}$ over the nodes of the graph to obtain $\bar{\mathbf{R}}^{(p)}$. The covariance operator of step p is

$$\mathbf{K}^{(p)} = \bar{\mathbf{R}}^{(p)}\bar{\mathbf{R}}^{(p)\top} \in \mathbb{R}^{n \times n}, \qquad p = 0, \ldots, k. \tag{2}$$

The entry $\mathbf{K}^{(p)}_{ij}$ is the inner product of the centered propagated projected features of nodes i and j, so $\mathbf{K}^{(p)}$ is in general dense and relates nodes that are many hops apart. Because $\mathbf{\Omega}$ is isotropic, $\mathbf{K}^{(p)}$ is invariant in distribution to permutations of the input features, and $\mathbf{K}^{(p)}/h$ concentrates, as h grows, around an operator that is invariant to orthogonal changes of the feature basis (Eliasof et al., 2026).

Rotation-invariant inputs. The projected features $\bar{\mathbf{R}}^{(p)}$ themselves live in an arbitrary basis: replacing $\mathbf{\Omega}$ by $\mathbf{\Omega}\mathbf{O}$ for an orthogonal $\mathbf{O} \in \mathbb{R}^{h \times h}$ rotates them to $\bar{\mathbf{R}}^{(p)}\mathbf{O}$. Because $\mathbf{\Omega}$ and $\mathbf{\Omega}\mathbf{O}$ have the same distribution, no coordinate of $\bar{\mathbf{R}}^{(p)}$ has a fixed meaning, and an encoder that reads $\bar{\mathbf{R}}^{(p)}$ sees a new basis at every redraw. Instead, the learned branches read a feature vector $\mathbf{F}_i$ per node, stacked into a matrix $\mathbf{F} \in \mathbb{R}^{n \times f}$, that depends on $\mathbf{\Omega}$ only through entries of $\mathbf{K}^{(p)}$:

- random-walk structural encodings (RWSE) of walk length ℓ (Dwivedi et al., 2022) and the log-degree $\log(1 + \deg_i)$ of node i;
- propagated norms $\|\bar{\mathbf{r}}^{(p)}_i\|/\sqrt{h} = (\mathbf{K}^{(p)}_{ii}/h)^{1/2}$, with $\bar{\mathbf{r}}^{(p)}_i$ the i-th row of $\bar{\mathbf{R}}^{(p)}$;
- landmark similarities $\mathbf{K}^{(0)}_{iu}/h$ to the L highest-degree nodes u (You et al., 2019);
- label-propagation statistics: for $t = 1, \ldots, T$, we propagate the support labels, padded with zeros on the other nodes, through $\hat{\mathbf{A}}$ for t steps, normalize each row, and record its normalized entropy, top-1 mass, and top-1 minus top-2 margin.

The width f depends on neither d nor C. Because the norms and landmark similarities are entries of $\mathbf{K}^{(p)}$, each learned branch reads covariance information, and we call these five branches the covariance branches.

Covariance branches. Each covariance branch b applies exactly one operator from $(\mathbf{I}, \mathbf{A}, \mathbf{K}^{(0)}, \ldots, \mathbf{K}^{(k)})$, giving $k+3$ branches, and has its own encoder with input $\mathbf{X}_{\mathrm{se},b} = \phi_b(\mathbf{F}) \in$

$\mathbb{R}^{n\times h'}$, where ϕ_b is affine. The sparse operators $\mathbf{I}$ and $\mathbf{A}$ are applied by a GIN aggregation (Xu et al., 2019). For a global operator, the encoder computes values $\mathbf{V}_b = \phi_{V,b}(\mathbf{X}_{\mathrm{se},b}) \in \mathbb{R}^{n\times h'}$ with a learned affine map $\phi_{V,b}$ and propagates them as

$$\mathbf{K}^{(p)}\mathbf{V}_b = \bar{\mathbf{R}}^{(p)}\big(\bar{\mathbf{R}}^{(p)\top}\mathbf{V}_b\big). \tag{3}$$

Equation (3) is equivalent to a graph attention whose queries and keys are fixed to $\bar{\mathbf{R}}^{(p)}$ and whose values are learned, so the unnormalized attention matrix is exactly $\mathbf{K}^{(p)}$. Evaluating the product from right to left does not form the $n \times n$ matrix. Each encoder adds an MLP, a residual connection, LayerNorm, and GELU, and outputs $\mathbf{H}_b \in \mathbb{R}^{n\times h'}$ (Appendix A.2). Unlike a GPS layer (Rampášek et al., 2022), which sums a local and a global term inside every layer, each branch keeps a single operator, so that each ridge fit and each selection decision in Section 3.3 concerns one operator.

Invariance to the projection basis. Centering and propagation act on $\mathbf{R}^{(p)}$ from the left, so replacing $\mathbf{\Omega}$ by $\mathbf{\Omega O}$ maps $\bar{\mathbf{R}}^{(p)}$ to $\bar{\mathbf{R}}^{(p)}\mathbf{O}$ and leaves $\mathbf{K}^{(p)} = \bar{\mathbf{R}}^{(p)}\mathbf{O}\mathbf{O}^\top\bar{\mathbf{R}}^{(p)\top} = \bar{\mathbf{R}}^{(p)}\bar{\mathbf{R}}^{(p)\top}$ unchanged. Hence $\mathbf{F}$, $\mathbf{V}_b$, the product in Equation (3), and every representation $\mathbf{H}_b$ are unchanged, so no learned input changes under a rotation of the projection. A fresh draw of $\mathbf{\Omega}$ is not a rotation of the previous one, and $\mathbf{K}^{(p)}$ varies between draws. Its expectation is $h\,\bar{\mathbf{X}}^{(p)}\bar{\mathbf{X}}^{(p)\top}$, where $\bar{\mathbf{X}}^{(p)}$ is the centered $\hat{\mathbf{A}}_{\mathrm{sl}}^{p}\mathbf{X}$, because $\mathbb{E}[\mathbf{\Omega}\mathbf{\Omega}^\top] = h\mathbf{I}_d$; we further average predictions over several draws at inference.

Computational complexity. Applying a global covariance operator by Equation (3) costs $O(nhh')$ time and $O(n(h + h'))$ memory, instead of $O(n^2h)$ time and $O(n^2)$ memory for forming $\mathbf{K}^{(p)}$. Applying a sparse operator costs $O(|\mathcal{E}|h')$, each encoder MLP adds $O(nh'^2)$, and the unprojected branches cost $O(k|\mathcal{E}|d)$ and are computed once; RWSE is estimated with Hutchinson probes on large graphs (Appendix A.1).

3.2 A Closed-Form Readout Chosen on the Support Set

A classifier for a new graph needs logits for classes that were not seen in training. Following GraphAny, AnyGRAM obtains them in closed form from the support labels, and it also chooses the regularization strength on the target graph. For branch b, ridge regression onto the one-hot support labels gives

$$\mathbf{W}_b^\star = \big(\mathbf{H}_{b,S}^\top\mathbf{H}_{b,S} + \lambda\mathbf{I}\big)^{-1}\mathbf{H}_{b,S}^\top\mathbf{Y}_S, \qquad \hat{\mathbf{Y}}_{b,Q} = \mathbf{H}_{b,Q}\,\mathbf{W}_b^\star, \tag{4}$$

where $\mathbf{W}_b^\star \in \mathbb{R}^{D_b\times C}$, $\mathbf{H}_{b,S}$ and $\mathbf{H}_{b,Q}$ are the rows of $\mathbf{H}_b$ indexed by S and Q, and $\hat{\mathbf{Y}}_{b,S} = \mathbf{H}_{b,S}\mathbf{W}_b^\star$ are the fitted support values. The number of classes enters only through $\mathbf{Y}_S$, so no learned parameter has a shape that depends on C. Using a differentiable closed-form solver for Equation (4) (Bertinetto et al., 2019) makes it differentiable in $\mathbf{H}_b$, so during training the task loss passes through it into the encoders.

Regularization chosen by generalized cross-validation. The branches differ in scale and conditioning, and a suitable λ also changes with the graph and the size of its support set. We therefore choose λ for each branch and each graph by generalized cross-validation (GCV) (Golub et al., 1979), minimizing

$$\mathrm{GCV}(\lambda) = \frac{\|\mathbf{Y}_S - \hat{\mathbf{Y}}_{b,S}\|_F^2/m}{\big(1 - \operatorname{tr}\mathbf{\Psi}_\lambda/m\big)^2}, \qquad \operatorname{tr}\mathbf{\Psi}_\lambda = \sum_j \frac{s_j}{s_j + \lambda}, \tag{5}$$

over 20 log-spaced values between $10^{-4}\bar{s}$ and $10^{4}\bar{s}$, where $\mathbf{\Psi}_\lambda = \mathbf{H}_{b,S}(\mathbf{H}_{b,S}^\top\mathbf{H}_{b,S} + \lambda\mathbf{I})^{-1}\mathbf{H}_{b,S}^\top$ is the hat matrix, s_j are the $r = \min(m, D_b)$ eigenvalues of the smaller of the Gram matrices $\mathbf{H}_{b,S}^\top\mathbf{H}_{b,S}$ and $\mathbf{H}_{b,S}\mathbf{H}_{b,S}^\top$, and $\bar{s}$ is their mean.

One eigendecomposition for all candidate values of λ. For a fixed $\mathbf{H}_{b,S}$, λ enters only through $\mathbf{H}_{b,S}^\top\mathbf{H}_{b,S} + \lambda\mathbf{I}$, which has the same eigenvectors as $\mathbf{H}_{b,S}^\top\mathbf{H}_{b,S}$ and eigenvalues shifted by λ. One eigendecomposition of the smaller Gram matrix therefore serves all 20 candidate values: the ridge solutions, GCV scores, and hat diagonals ψ_{ii} differ only by diagonal factors $1/(s_j + \lambda)$ (Appendix B). This costs $O(r^3 + mr(D_b + C))$ once per branch and support set, plus $O(r)$ per candidate. At the selected λ, ψ_{ii} also gives the exact leave-one-out predictions $\hat{\mathbf{y}}_i^{\mathrm{LOO}} = \mathbf{y}_i - (\mathbf{y}_i - \hat{\mathbf{y}}_i)/(1 - \psi_{ii})$, where $\mathbf{y}_i$ and $\hat{\mathbf{y}}_i$ are the rows of $\mathbf{Y}_S$ and $\hat{\mathbf{Y}}_{b,S}$ for support node i; their accuracy $\rho_b \in [0, 1]$ is read by the training fusion as a reliability feature.

3.3 Training Across Graphs and Selecting on the Target Graph

The encoders of the covariance branches are the only learned components that AnyGRAM uses on a target graph. They are trained across the source graphs through a learned soft fusion of the branch predictions, which is then replaced on the target graph by a selection validated on its support set.

Training through a soft fusion. For each query node n, a one hidden-layer MLP reads features of the gradient-detached branch predictions whose size depends only on B: the pairwise distance features of GraphAny, per-branch confidence, and the reliabilities ρ_b (Appendix C). It outputs attention weights $\boldsymbol{\alpha}_n$ over the branches, and the fused prediction $\hat{\mathbf{y}}_n^{\text{fused}} = \sum_b \alpha_{nb} \hat{\mathbf{y}}_{nb}$, where $\hat{\mathbf{y}}_{nb}$ is the row of $\hat{\mathbf{Y}}_{b,Q}$ for node n, uses the live branch logits, so the task loss reaches the encoders through the ridge solve but not through the routing input. The loss on each source graph is

$$\mathcal{L} = \frac{\text{CE}(\hat{\mathbf{Y}}_Q^{\text{fused}}, \mathbf{Y}_Q)}{\log C} + w_{\text{proto}}\mathcal{L}_{\text{proto}} + w_{\text{ent}}\mathcal{L}_{\text{ent}} + w_{\text{oracle}}\mathcal{L}_{\text{oracle}}, \tag{6}$$

where $\mathbf{Y}_Q$ are the one-hot query labels of the source graph and dividing by $\log C$ puts source graphs with different numbers of classes on a common scale. The prototypical loss $\mathcal{L}_{\text{proto}}$ (Snell et al., 2017) gives each covariance encoder a direct gradient, including encoders to which the fusion assigns little weight, $\mathcal{L}_{\text{ent}}$ discourages the attention from collapsing onto one branch, and $\mathcal{L}_{\text{oracle}}$ rewards attention on branches that classify a query node correctly (Appendix C).

Projection ensembling. On a target graph, the support set is its training split and the query set is its test split. We repeat the forward pass for E independent draws of $\boldsymbol{\Omega}$ and average the query logits of each branch, $\bar{\mathbf{y}}_{nb} = \frac{1}{E}\sum_{e=1}^{E} \hat{\mathbf{y}}_{nb}^{(e)}$.

Support-validated branch selection. We estimate how well each branch classifies the target graph by cross-validation on its support set (support CV). We split S into stratified folds and, for each fold, recompute every quantity that depends on labels, namely the label-propagation statistics and the branch representations built from them, the ridge fits, and the GCV choice of λ, using only the labels of the other folds. For a support node i in the scored set $S_{\text{cv}} \subseteq S$ (Appendix D), let $\chi_{ib} \in \{0, 1\}$ indicate that the out-of-fold prediction of branch b, averaged over the E draws, is correct. The support-CV accuracy and the global branch are

$$a_b = \frac{1}{|S_{\text{cv}}|} \sum_{i \in S_{\text{cv}}} \chi_{ib}, \qquad b_{\text{g}} = \arg\max_b a_b. \tag{7}$$

We use five folds, reduced when necessary to the smallest class count among classes with at least two support nodes (Appendix D). Unlike ρ_b, this estimate reselects λ and rebuilds the label-dependent inputs without each held-out label.

Node-level overrides. Different branches can succeed in different regions of one graph. For a query node n, let $\mathcal{N}(n)$ be the κ nodes of S_{cv} most similar to n by cosine similarity between the rows $\mathbf{x}_n$ and $\mathbf{x}_i$ of $\mathbf{X}$, with weights $w_{ni} = \max\{0, \cos(\mathbf{x}_n, \mathbf{x}_i)\}^2$ and total weight $\omega_n = \sum_{i \in \mathcal{N}(n)} w_{ni}$. The local reliability of branch b shrinks the weighted local accuracy toward a_b,

$$q_{nb} = \frac{\eta\, a_b + \sum_{i \in \mathcal{N}(n)} w_{ni}\, \chi_{ib}}{\eta + \omega_n}. \tag{8}$$

Let $b_{\text{loc}}(n) = \arg\max_b q_{nb}$. Node n uses $b_n^\star = b_{\text{loc}}(n)$ if $q_{n,b_{\text{loc}}(n)} - q_{n,b_{\text{g}}} > \delta$ and $\omega_n \geq \mu$, and $b_n^\star = b_{\text{g}}$ otherwise, and predicts $\hat{y}_n = \arg\max_c \bar{y}_{n,b_n^\star,c}$. We use $\kappa = 32$, $\eta = 10$, $\delta = 0.05$, and $\mu = 2$. Query labels do not enter these estimates, and no gradient step is taken on the target graph.

4 Experiments

Section 3 separates what AnyGRAM learns across graphs, the covariance encoders, from what it decides on the target graph, the ridge regularization and the branch. This separation lets us remove one component at a time while keeping the rest of the model fixed, and compare different readouts on exactly the same branch predictions. Our experiments seek to address the following questions:

(Q1) Does a frozen AnyGRAM trained on a single source graph transfer to unseen graphs?

Table 1: **Trained once on Cora and applied frozen, AnyGRAM reaches the highest average accuracy.** Accuracy (%) ↑, mean ± sample standard deviation over five seeds; end-to-end models are trained per graph and zero-shot models on Cora only. Graphs are grouped by adjusted homophily (Platonov et al., 2023). The Average row averages the per-graph means and standard deviations. The top three fully-inductive methods are marked by **First**, **Second**, **Third**.

	End-to-end		Zero-shot (fully-inductive)			
Graph	MeanGNN	GAT	GraphAny	TS-Mean	TS-GAT	AnyGRAM
Homophilic						
arxiv	50.94 ± 0.38	57.93 ± 3.44	**58.85** ± 0.03	**56.33** ± 2.58	53.13 ± 2.04	**62.66** ± 0.02
citeseer	65.06 ± 1.30	63.92 ± 0.84	**68.88** ± 0.10	**68.66** ± 0.19	**68.70** ± 0.48	68.48 ± 3.73
co-cs	80.87 ± 0.69	82.28 ± 0.86	**90.06** ± 0.80	**90.92** ± 0.47	**91.13** ± 0.43	90.02 ± 1.53
co-physics	79.05 ± 1.13	85.92 ± 1.10	**91.85** ± 0.34	**92.61** ± 0.61	**92.23** ± 0.61	91.57 ± 1.73
computers	73.88 ± 0.88	70.94 ± 3.40	**82.79** ± 1.13	**81.37** ± 1.25	80.23 ± 2.44	**84.48** ± 1.00
full-DBLP	65.01 ± 2.40	67.34 ± 2.75	**71.48** ± 1.44	**66.42** ± 3.65	64.30 ± 3.75	**73.84** ± 1.48
full-cora	55.85 ± 1.04	59.95 ± 0.88	51.18 ± 0.78	**53.58** ± 0.73	**52.82** ± 0.90	**61.81** ± 0.47
photo	88.95 ± 1.08	80.78 ± 3.59	**89.91** ± 0.88	**90.18** ± 1.30	89.66 ± 1.23	**90.01** ± 0.77
pubmed	74.56 ± 0.13	75.12 ± 0.89	**76.46** ± 0.08	74.98 ± 0.56	**75.46** ± 0.86	**79.10** ± 0.00
wiki-attr	74.23 ± 0.89	68.91 ± 9.50	60.27 ± 3.06	**69.89** ± 1.31	**72.53** ± 1.59	**74.64** ± 2.45
wiki-cs	71.97 ± 1.70	74.99 ± 0.59	**74.11** ± 0.60	**74.16** ± 2.07	73.24 ± 2.49	**77.19** ± 0.37
Heterophilic						
actor	32.03 ± 0.29	32.59 ± 0.83	27.54 ± 0.20	**28.09** ± 0.93	**28.80** ± 2.12	**34.18** ± 1.06
amazon-ratings	40.74 ± 0.13	40.63 ± 0.66	**42.80** ± 0.09	**42.27** ± 1.40	41.92 ± 0.47	**43.17** ± 0.19
blogcatalog	84.48 ± 0.74	78.20 ± 7.23	71.54 ± 3.04	**76.30** ± 2.92	**78.44** ± 5.18	**73.55** ± 5.29
brazil	32.31 ± 7.50	35.38 ± 4.21	33.84 ± 15.65	**39.23** ± 5.70	**36.92** ± 10.03	**65.38** ± 4.71
chameleon	61.75 ± 0.94	58.46 ± 6.23	**63.64** ± 1.48	**60.83** ± 5.41	54.65 ± 4.58	**69.30** ± 1.20
cornell	63.78 ± 1.48	69.73 ± 2.26	63.24 ± 1.32	**68.65** ± 2.42	**71.35** ± 4.52	**79.46** ± 2.42
europe	39.12 ± 6.44	39.00 ± 4.30	**41.25** ± 7.25	35.88 ± 6.91	**38.38** ± 6.32	**49.63** ± 3.44
minesweeper	84.06 ± 0.18	84.15 ± 0.24	80.46 ± 0.11	**80.68** ± 0.38	**80.72** ± 0.78	**82.18** ± 0.24
questions	97.16 ± 0.06	97.13 ± 0.05	**97.07** ± 0.03	**97.02** ± 0.01	**97.02** ± 0.01	**97.10** ± 0.04
roman-empire	69.37 ± 0.66	69.80 ± 4.18	63.34 ± 0.58	**66.36** ± 1.02	**67.76** ± 0.60	**67.65** ± 0.23
squirrel	43.32 ± 0.66	38.16 ± 1.04	**49.74** ± 0.47	**41.81** ± 0.80	38.29 ± 4.29	**56.41** ± 2.11
texas	76.76 ± 1.48	81.62 ± 6.45	71.35 ± 2.16	**73.51** ± 4.01	**74.05** ± 4.10	**80.54** ± 2.96
tolokers	78.59 ± 0.66	78.22 ± 0.37	**78.20** ± 0.03	**78.12** ± 0.09	78.01 ± 0.23	**80.03** ± 0.17
usa	43.93 ± 1.16	43.03 ± 2.08	**43.35** ± 1.62	**42.34** ± 2.12	41.69 ± 2.59	**52.04** ± 2.38
wisconsin	74.12 ± 12.20	73.33 ± 8.27	59.61 ± 5.77	**61.18** ± 11.38	**65.88** ± 9.86	**87.06** ± 1.07
Average	65.46 ± 1.78	65.67 ± 2.93	65.49 ± 1.89	**65.82** ± 2.32	**65.67** ± 2.79	**71.98** ± 1.58

(Q2) How do the covariance branches and global covariance propagation affect transfer?
(Q3) Which decisions benefit from being made on the target graph?
(Q4) How much does transfer depend on the choice and the number of source graphs?

4.1 Experimental Setup

All experiments follow the evaluation protocol of Finkelshtein et al. (2025) on 26 target graphs (Appendix E) that differ in domain, size, feature dimension, number of classes, and homophily (Table 1). Deezer and LastFM-Asia were omitted due to availability issues. Unless stated otherwise, AnyGRAM is trained on Cora alone. All runs use five seeds; we report the mean and sample standard deviation of the accuracy on each graph, and averages are unweighted means over graphs. On each target graph, the support set is the training split and the query set is the test split; no other label is used by any deployed readout, and no parameter is updated on the target graph. One hyperparameter configuration, shared by all experiments, was selected on the validation split of Cora (Appendix E).

Comparisons. The end-to-end models MeanGNN (Finkelshtein et al., 2025) and GAT (Veličković et al., 2018) are trained on each target graph and show what per-graph training achieves. GraphAny (Zhao et al., 2025) uses the same five unprojected branches as AnyGRAM with a learned attention readout, so it isolates what the covariance branches and the decisions made on the target graph add. The triple-symmetry networks TS-Mean and TS-GAT (Finkelshtein et al., 2025) are

Table 2: **Both branch families and the decisions made on the target graph contribute to transfer.** Average accuracy (%) ↑ over the 26 target graphs after training on Cora, mean ± sample standard deviation over five seed-wise averages. Δ is the paired difference from the full model in percentage points; * marks $p < 0.05$ in an uncorrected paired t-test (retrained rows only). #B is the number of branches.

Variant	#B	Accuracy (%)	Δ
FULL MODEL			
AnyGRAM (support-validated selection)	10	72.0 ± 0.5	–
RETRAINED WITH ONE COMPONENT CHANGED			
without the raw and low-pass branches ($\mathbf{X}$, $\hat{\mathbf{A}}\mathbf{X}$, $\hat{\mathbf{A}}^2\mathbf{X}$)	7	66.5 ± 0.7	-5.5^*
without covariance branches	5	69.8 ± 0.6	-2.2^*
with a fixed ridge multiplier instead of GCV	10	70.8 ± 0.7	-1.1^*
without global covariance propagation	10	71.7 ± 0.6	-0.3
without high-pass branches	8	71.9 ± 0.5	-0.1
without the local aggregation term	10	71.9 ± 0.5	-0.1
with uniform instead of learned fusion in training	10	72.0 ± 0.4	+0.1
OTHER READOUTS OF THE SAME BRANCH PREDICTIONS			
Global selection (no node-level overrides)	–	71.9 ± 0.5	-0.1
Per-node selection	–	71.7 ± 0.4	-0.2
Learned soft fusion	–	69.5 ± 0.6	-2.5
Uniform branch average	–	70.0 ± 0.6	-2.0
Hard attention routing (top-1)	–	66.9 ± 0.8	-5.1
QUERY LABEL ORACLES			
Per-graph oracle	–	72.7 ± 0.3	+0.7
Per-node oracle	–	90.1 ± 0.1	+18.1

the strongest zero-shot models reported under this protocol; all baseline numbers are taken from Finkelshtein et al. (2025). Retrained ablations change one component of AnyGRAM at a time, and alternative readouts are applied to the branch predictions of the same trained model. Finally, two query label oracles choose the branch per graph or per query node using query labels; they are reference bounds and are not used by any deployed readout.

4.2 (A1) AnyGRAM Transfers from a Single Source Graph

AnyGRAM trained on Cora alone outperforms the zero-shot baselines and both end-to-end models on average. It reaches an average accuracy of 71.98% (Table 1), compared with 65.82% for TS-Mean, the strongest zero-shot baseline, and 65.67% and 65.46% for GAT and MeanGNN trained on each target graph. It has the highest zero-shot mean on 20 of the 26 graphs, and on 10 of them its margin over the strongest zero-shot baseline exceeds two combined standard deviations. The largest gains appear on the airport graphs Brazil ($+26.15$), USA ($+8.69$), and Europe ($+8.38$), on the heterophilous web graphs Wisconsin ($+21.18$) and Cornell ($+8.11$), and on full-Cora ($+8.23$). The improvement is not confined to these graphs: without the five largest gains, AnyGRAM still exceeds the strongest zero-shot baseline by 2.09 points per graph on average. Of the six graphs where a baseline is ahead, the gap exceeds two points only on BlogCatalog (-4.89 against TS-GAT); the other five gaps are at most 1.11 points. AnyGRAM also has the lowest average seed standard deviation among the zero-shot methods, 1.58 compared with 1.89 to 2.79.

4.3 (A2) Covariance Branches Improve Transfer

Removing all five covariance branches lowers the average accuracy from 72.0% to 69.8% (Table 2). This ablation measures the contribution of the complete branch family, including its invariant inputs and learned encoders. Keeping all ten branches and their invariant inputs while removing only the global product of Equation (3) lowers accuracy to 71.7%. In contrast, removing the local aggregation term of the encoders or the two high-pass unprojected branches changes the average by at most 0.1 points. Removing the raw and low-pass unprojected branches $\mathbf{X}$, $\hat{\mathbf{A}}\mathbf{X}$, and $\hat{\mathbf{A}}^2\mathbf{X}$ costs the most, 5.5 points.

The covariance branches matter on a specific group of graphs, and support cross-validation finds them there. On Brazil, Europe, USA, full-DBLP, CiteSeer, and Tolokers, it assigns at least 80% of the query nodes to covariance branches (Figure 3a in Appendix F), and these six graphs show the largest paired gains from adding the covariance branches, from $+25.38$ points on Brazil and $+10.25$ on Europe to $+1.9$ on Tolokers (Figure 3b). On none of the 26 graphs does adding them lower accuracy by more than 1.2 points. On the three airport graphs, whose labels reflect airport activity (Ribeiro et al., 2017), the selected branches include the identity branch $\mathbf{I}$, so part of these gains may come from the structural encodings in $\mathbf{F}$ rather than from global propagation, which the ablation above isolates.

4.4 (A3) Decisions Made on the Target Graph Outperform Learned Readouts

Choosing the readout on the target graph improves transfer at both levels at which AnyGRAM makes a choice. For the regularization, replacing GCV by a fixed ridge multiplier lowers the average accuracy by 1.1 points (Table 2). For the branch, on the same predictions of the full model, global selection reaches 71.9%, compared with 69.5% for the learned soft fusion used in training, 70.0% for a uniform average of the branches, and 66.9% for hard top-1 attention routing. Support-validated selection also exceeds the learned soft fusion for each of the four source graphs of Table 4 in Appendix F, by 1.9 to 3.3 points. Training with a uniform fusion instead of the learned one gives 72.0%, equal to 72.0% for the full model, so the learned fusion contributes little to what transfers: it trains the encoders, and the support labels of the target graph make the final choice better than a head learned on other graphs.

Decisions at the level of single nodes are harder to make from the support set. Node-level overrides add 0.1 points over global selection, and per-node selection is 0.2 points below it. AnyGRAM comes within 0.7 points of the per-graph oracle, which reaches 72.7%, whereas the per-node oracle reaches 90.1%.

4.5 (A4) Transfer Is Stable Across Source Graphs

AnyGRAM reaches its accuracy with a single source graph, and its accuracy changes little with which graph it is. Training on Cora, Chameleon, Roman-empire, or Wiki-CS gives average accuracies between 71.95% and 72.16% on the 23 graphs outside all four source graphs (Table 4 in Appendix F). Adding source graphs does not improve the average either (Figure 2 in Appendix F). On a fixed set of 18 target graphs, AnyGRAM reaches 74.05% with one source graph and 74.20% with nine, and every value in between lies within 0.28 points of the one-source value, while TS-Mean improves from 67.23% to 68.57% and GraphAny stays between 67.15% and 67.35%. We attribute this stability to the division of Section 3: the choices that depend on the target graph are made on its support set, and on 9 of the 26 graphs the accuracy is identical for every source graph we trained on, consistent with parameter-free branches being selected there regardless of the source graph (Figure 3a).

5 Conclusion

We asked which parts of a fully-inductive node classifier should be learned across graphs and which should be decided on the target graph. AnyGRAM learns encoders whose inputs are invariant to rotations of the random feature projection, which makes global covariance operators usable after transfer: they bring large gains on the graphs where cross-validation selects them and cost at most 1.2 points elsewhere. It decides the ridge regularization and the branch of each target graph by cross-validation on its support labels, which outperforms the learned readouts we evaluate. Trained on one source graph and applied frozen to 26 target graphs, AnyGRAM outperforms zero-shot baselines and end-to-end models on average, largely independently of the source graph.

Limitations. The gap between the per-graph and per-node oracles reveals a node-level complementarity that support cross-validation does not recover; a support set may be large enough to rank ten branches for a whole graph but not for each region of it. The flat scaling curve also suggests that the current encoders saturate with one source graph, so richer invariant inputs may be needed to benefit from more source graphs.

AI use statement. In this work, we used generative AI tools for polishing the paper writing, aiding coding, and related literature search. Method and experiment design did not use AI. We have reviewed all AI-assisted work. We take responsibility for the final content of this work, including text, claims or artifacts produced with the aid of generative AI.

Ethics statement. This work uses publicly available benchmark graphs and does not involve human subjects or personal data. We are not aware of ethical concerns specific to this work.

Reproducibility statement. All graphs are public node classification benchmarks, and we follow the evaluation protocol and splits of Finkelshtein et al. (2025) (Section 4.1). AnyGRAM is trained on source graphs only and applied with frozen parameters: no gradient step is taken on any target graph, and at inference target labels enter only through the support set. A single hyperparameter configuration was selected on the validation split of Cora, and no hyperparameter was tuned on the evaluation graphs (Appendix E). Section 3 and Appendices A to D specify every component, including the invariant features, the encoder equations, the ridge readout with its numerical safeguards, the training losses, and the inference procedure, and Appendix E lists every hyperparameter. All results are means and sample standard deviations over five seeds, and every table reports its per-graph results in full. Baseline numbers are taken from Finkelshtein et al. (2025). Experiments ran on NVIDIA A100 80GB GPUs; a full training run takes about 2.3 minutes excluding data loading, and zero-shot inference over all 26 target graphs takes about 2 minutes. An anonymized implementation of AnyGRAM, including data preprocessing and evaluation scripts will be released upon publication.

REFERENCES

Ralph Abboud, İsmail İlkan Ceylan, Martin Grohe, and Thomas Lukasiewicz. The surprising power of graph neural networks with random node initialization. In *International Joint Conference on Artificial Intelligence (IJCAI)*, pp. 2112–2118, 2021.

Costas Bekas, Effrosyni Kokiopoulou, and Yousef Saad. An estimator for the diagonal of a matrix. *Applied Numerical Mathematics*, 57(11–12):1214–1229, 2007.

Luca Bertinetto, João F. Henriques, Philip H. S. Torr, and Andrea Vedaldi. Meta-learning with differentiable closed-form solvers. In *International Conference on Learning Representations (ICLR)*, 2019.

Deyu Bo, Xiao Wang, Chuan Shi, and Huawei Shen. Beyond low-frequency information in graph convolutional networks. In *AAAI Conference on Artificial Intelligence (AAAI)*, pp. 3950–3957, 2021.

Rich Caruana, Alexandru Niculescu-Mizil, Geoff Crew, and Alex Ksikes. Ensemble selection from libraries of models. In *International Conference on Machine Learning (ICML)*, 2004.

Eli Chien, Jianhao Peng, Pan Li, and Olgica Milenkovic. Adaptive universal generalized PageRank graph neural network. In *International Conference on Learning Representations (ICLR)*, 2021.

Chenhui Deng, Zichao Yue, and Zhiru Zhang. Polynormer: Polynomial-expressive graph transformer in linear time. In *International Conference on Learning Representations (ICLR)*, 2024.

Vijay Prakash Dwivedi, Anh Tuan Luu, Thomas Laurent, Yoshua Bengio, and Xavier Bresson. Graph neural networks with learnable structural and positional representations. In *International Conference on Learning Representations (ICLR)*, 2022.

Moshe Eliasof, Krishna Sri Ipsit Mantri, Beatrice Bevilacqua, Bruno Ribeiro, and Carola-Bibiane Schönlieb. Bridging input feature spaces towards graph foundation models. In *International Conference on Learning Representations (ICLR)*, 2026. arXiv:2605.04834.

Dmitry Eremeev, Gleb Bazhenov, Oleg Platonov, Artem Babenko, and Liudmila Prokhorenkova. Turning tabular foundation models into graph foundation models. *arXiv preprint arXiv:2508.20906*, 2025.

Dmitry Eremeev, Oleg Platonov, Gleb Bazhenov, Artem Babenko, and Liudmila Prokhorenkova. GraphPFN: A prior-data fitted graph foundation model. In *International Conference on Machine Learning (ICML)*, 2026. arXiv:2509.21489.

Taoran Fang, Yunchao Zhang, Yang Yang, Chunping Wang, and Lei Chen. Universal prompt tuning for graph neural networks. In *Advances in Neural Information Processing Systems (NeurIPS)*, pp. 52464–52489, 2023.

Ben Finkelshtein, İsmail İlkan Ceylan, Michael Bronstein, and Ron Levie. Equivariance everywhere all at once: A recipe for graph foundation models. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2025. arXiv:2506.14291.

Fabrizio Frasca, Emanuele Rossi, Davide Eynard, Ben Chamberlain, Michael Bronstein, and Federico Monti. SIGN: Scalable inception graph neural networks. *arXiv preprint arXiv:2004.11198*, 2020.

Gene H. Golub, Michael Heath, and Grace Wahba. Generalized cross-validation as a method for choosing a good ridge parameter. *Technometrics*, 21(2):215–223, 1979.

Adrian Hayler, Xingyue Huang, İsmail İlkan Ceylan, Michael Bronstein, and Ben Finkelshtein. Bringing graphs to the table: Zero-shot node classification via tabular foundation models. *arXiv preprint arXiv:2509.07143*, 2025.

Arthur E. Hoerl and Robert W. Kennard. Ridge regression: Biased estimation for nonorthogonal problems. *Technometrics*, 12(1):55–67, 1970.

Michael F. Hutchinson. A stochastic estimator of the trace of the influence matrix for Laplacian smoothing splines. *Communications in Statistics – Simulation and Computation*, 18(3):1059–1076, 1989.

Robert A. Jacobs, Michael I. Jordan, Steven J. Nowlan, and Geoffrey E. Hinton. Adaptive mixtures of local experts. *Neural Computation*, 3(1):79–87, 1991.

Angelos Katharopoulos, Apoorv Vyas, Nikolaos Pappas, and François Fleuret. Transformers are RNNs: Fast autoregressive transformers with linear attention. In *International Conference on Machine Learning (ICML)*, pp. 5156–5165, 2020.

Divyansha Lachi, Mehdi Azabou, Vinam Arora, and Eva L. Dyer. GraphFM: A generalist graph transformer that learns transferable representations across diverse domains. *Transactions on Machine Learning Research*, 2025.

Balaji Lakshminarayanan, Alexander Pritzel, and Charles Blundell. Simple and scalable predictive uncertainty estimation using deep ensembles. In *Advances in Neural Information Processing Systems (NIPS)*, 2017.

Dooho Lee, Myeong Kong, Minho Jeong, and Jaemin Yoo. View space: Learning representation across arbitrary graphs. In *International Conference on Machine Learning (ICML)*, 2026. URL `https://openreview.net/forum?id=9klc62KROH`.

Kwonjoon Lee, Subhransu Maji, Avinash Ravichandran, and Stefano Soatto. Meta-learning with differentiable convex optimization. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019.

Hao Liu, Jiarui Feng, Lecheng Kong, Ningyue Liang, Dacheng Tao, Yixin Chen, and Muhan Zhang. One for all: Towards training one graph model for all classification tasks. In *International Conference on Learning Representations (ICLR)*, 2024.

Haitao Mao, Zhikai Chen, Wenzhuo Tang, Jianan Zhao, Yao Ma, Tong Zhao, Neil Shah, Mikhail Galkin, and Jiliang Tang. Position: Graph foundation models are already here. In *International Conference on Machine Learning (ICML)*, pp. 34670–34692, 2024.

Oleg Platonov, Denis Kuznedelev, Artem Babenko, and Liudmila Prokhorenkova. Characterizing graph datasets for node classification: Homophily-heterophily dichotomy and beyond. In *Advances in Neural Information Processing Systems*, volume 36, 2023.

Ladislav Rampášek, Mikhail Galkin, Vijay Prakash Dwivedi, Anh Tuan Luu, Guy Wolf, and Dominique Beaini. Recipe for a general, powerful, scalable graph transformer. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 35, pp. 14501–14515, 2022.

Leonardo F. R. Ribeiro, Pedro H. P. Savarese, and Daniel R. Figueiredo. struc2vec: Learning node representations from structural identity. In *ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD)*, pp. 385–394, 2017.

Ryoma Sato, Makoto Yamada, and Hisashi Kashima. Random features strengthen graph neural networks. In *SIAM International Conference on Data Mining (SDM)*, pp. 333–341, 2021.

Noam Shazeer, Azalia Mirhoseini, Krzysztof Maziarz, Andy Davis, Quoc Le, Geoffrey Hinton, and Jeff Dean. Outrageously large neural networks: The sparsely-gated mixture-of-experts layer. In *International Conference on Learning Representations (ICLR)*, 2017.

Yangyi Shen, Jincheng Zhou, Beatrice Bevilacqua, Joshua Robinson, Charilaos Kanatsoulis, Jure Leskovec, and Bruno Ribeiro. Zero-shot generalization of GNNs over distinct attribute domains. In *International Conference on Machine Learning (ICML)*, pp. 54701–54731, 2025.

Saurabh Sihag, Gonzalo Mateos, Corey McMillan, and Alejandro Ribeiro. coVariance neural networks. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 35, pp. 17003–17016, 2022.

Jake Snell, Kevin Swersky, and Richard Zemel. Prototypical networks for few-shot learning. In *Advances in Neural Information Processing Systems (NIPS)*, 2017.

Jiabin Tang, Yuhao Yang, Wei Wei, Lei Shi, Lixin Su, Suqi Cheng, Dawei Yin, and Chao Huang. GraphGPT: Graph instruction tuning for large language models. In *ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR)*, pp. 491–500, 2024.

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

Felix Wu, Amauri Souza, Tianyi Zhang, Christopher Fifty, Tao Yu, and Kilian Weinberger. Simplifying graph convolutional networks. In *International Conference on Machine Learning (ICML)*, pp. 6861–6871, 2019.

Qitian Wu, Wentao Zhao, Zenan Li, David Wipf, and Junchi Yan. NodeFormer: A scalable graph structure learning transformer for node classification. In *Advances in Neural Information Processing Systems (NeurIPS)*, pp. 27387–27401, 2022.

Qitian Wu, Wentao Zhao, Chenxiao Yang, Hengrui Zhang, Fan Nie, Haitian Jiang, Yatao Bian, and Junchi Yan. SGFormer: Simplifying and empowering transformers for large-graph representations. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2023.

Lianghao Xia and Chao Huang. AnyGraph: Graph foundation model in the wild. In *Findings of the Association for Computational Linguistics: ACL 2026*, pp. 882–896, 2026.

Lianghao Xia, Ben Kao, and Chao Huang. OpenGraph: Towards open graph foundation models. In *Findings of the Association for Computational Linguistics: EMNLP 2024*, pp. 2365–2379, 2024.

Keyulu Xu, Weihua Hu, Jure Leskovec, and Stefanie Jegelka. How powerful are graph neural networks? In *International Conference on Learning Representations (ICLR)*, 2019.

Jiaxuan You, Rex Ying, and Jure Leskovec. Position-aware graph neural networks. In *International Conference on Machine Learning (ICML)*, pp. 7134–7143, 2019.

Xingtong Yu, Chang Zhou, Yuan Fang, and Xinming Zhang. Text-free multi-domain graph pretraining: Toward graph foundation models. *arXiv preprint arXiv:2405.13934*, 2024.

Haihong Zhao, Aochuan Chen, Xiangguo Sun, Hong Cheng, and Jia Li. All in one and one for all: A simple yet effective method towards cross-domain graph pretraining. In *ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD)*, pp. 4443–4454, 2024.

Jianan Zhao, Le Zhuo, Yikang Shen, Meng Qu, Kai Liu, Michael Bronstein, Zhaocheng Zhu, and Jian Tang. GraphText: Graph reasoning in text space. *arXiv preprint arXiv:2310.01089*, 2023.

Jianan Zhao, Zhaocheng Zhu, Mikhail Galkin, Hesham Mostafa, Michael M. Bronstein, and Jian Tang. Fully-inductive node classification on arbitrary graphs. In *International Conference on Learning Representations (ICLR)*, 2025. arXiv:2405.20445.

Jiong Zhu, Yujun Yan, Lingxiao Zhao, Mark Heimann, Leman Akoglu, and Danai Koutra. Beyond homophily in graph neural networks: Current limitations and effective designs. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 33, 2020.

ORGANIZATION OF THE APPENDIX

Appendix	Content	Supports
A	Invariant inputs, RWSE estimation, encoder equations	Section 3.1
B	Primal and dual ridge readout, GCV, leave-one-out reliability, differentiation	Section 3.2
C	Fusion head and auxiliary losses used in training	Section 3.3, (Q3)
D	Details of support cross-validation and node-level overrides	Section 3.3, (Q3)
E	All hyperparameters and their selection	Section 4.1
F	Source graphs, number of source graphs, branch routing, ablation notes	(Q2) to (Q4)

A BRANCH CONSTRUCTION AND ENCODER DETAILS

This appendix complements Section 3.1.

A.1 RANDOM-WALK STRUCTURAL ENCODINGS

Exact RWSE requires the diagonals of the first ℓ powers of the random-walk matrix $\mathbf{P}_{\mathrm{rw}} = \mathbf{D}_{\mathrm{deg}}^{-1}\mathbf{A}$, where $\mathbf{D}_{\mathrm{deg}}$ is the diagonal degree matrix, which is impractical for large graphs. For these graphs we use the Hutchinson diagonal estimator (Hutchinson, 1989; Bekas et al., 2007). Let $\mathbf{Z} \in \{-1, 1\}^{n \times n_{\mathrm{pr}}}$ have i.i.d. Rademacher entries, set $\mathbf{Z}^{(0)} = \mathbf{Z}$ and $\mathbf{Z}^{(t)} = \mathbf{P}_{\mathrm{rw}}\mathbf{Z}^{(t-1)}$, computed with sparse-dense products only. Then

$$\operatorname{diag}\left(\mathbf{P}_{\mathrm{rw}}^{t}\right) \approx \frac{1}{n_{\mathrm{pr}}} \sum_{j=1}^{n_{\mathrm{pr}}} \mathbf{Z}_{:,j} \odot \mathbf{Z}_{:,j}^{(t)}, \tag{9}$$

which follows from $\mathbb{E}[\mathbf{Z} \odot (\mathbf{P}_{\mathrm{rw}}^{t}\mathbf{Z})] = \operatorname{diag}(\mathbf{P}_{\mathrm{rw}}^{t})\mathbf{1}^{\top}$ and does not form a dense $n \times n$ matrix.

A.2 BRANCH ENCODER ARCHITECTURE

Let $\mathbf{F}$ denote the rotation-invariant node features of Section 3.1. Each covariance branch b has its own parameters. Its input is the affine map $\mathbf{X}_{\mathrm{se},b} = \phi_b(\mathbf{F}) \in \mathbb{R}^{n \times h'}$, and its output $\mathbf{H}_b \in \mathbb{R}^{n \times h'}$ is passed to the ridge readout of Section 3.2, so $D_b = h'$. LN_b denotes LayerNorm and MLP_b consists of two linear layers of width h' separated by a GELU activation.

Sparse branches. For $\mathbf{N}_b \in \{\mathbf{I}, \mathbf{A}\}$ we use a GIN aggregation (Xu et al., 2019):

$$\mathbf{M}_b = \mathrm{MLP}_b\big((1 + \epsilon_b)\,\mathbf{X}_{\mathrm{se},b} + \mathbf{N}_b\mathbf{X}_{\mathrm{se},b}\big), \qquad \mathbf{H}_b = \mathrm{GELU}\big(\mathrm{LN}_b(\mathbf{X}_{\mathrm{se},b} + \mathbf{M}_b)\big), \tag{10}$$

where ϵ_b is learned. The product $\mathbf{N}_b\mathbf{X}_{\mathrm{se},b}$ is a sparse matrix product, so no edge features are materialized.

Global branches. The branch associated with $\mathbf{K}^{(p)}$ uses attention with queries and keys fixed to $\bar{\mathbf{R}}^{(p)}$ and learned values. With learned affine maps $\phi_{V,b}$ and $\phi_{0,b}$,

$$\begin{aligned} \mathbf{V}_b &= \phi_{V,b}(\mathbf{X}_{\mathrm{se},b}), & \mathbf{M}_b &= \mathrm{MLP}_b\big((1 + \epsilon_b)\,\mathbf{X}_{\mathrm{se},b}\big), \\ \mathbf{J}_b &= \phi_{0,b}\big(\bar{\mathbf{R}}^{(p)}(\bar{\mathbf{R}}^{(p)\top}\mathbf{V}_b)\big), & \mathbf{H}_b &= \mathrm{GELU}\big(\mathrm{LN}_b(\mathbf{X}_{\mathrm{se},b} + \mathbf{M}_b + \mathbf{J}_b)\big). \end{aligned} \tag{11}$$

The product is evaluated from right to left as in Equation (3), costing $O(nhh')$ time and $O(n(h + h'))$ memory. These branches apply no adjacency aggregation; graph structure enters through the propagated projections $\bar{\mathbf{R}}^{(p)}$ for $p \geq 1$ and through the structural, norm, and label-propagation entries of $\mathbf{F}$.

B PRIMAL AND DUAL EIGENDECOMPOSITION FOR THE DIFFERENTIABLE RIDGE READOUT

This appendix complements Section 3.2. The readout carries no learned parameters but is differentiable, so the task loss backpropagates through the matrix inverse into the branch encoders. We

decompose the smaller of the primal feature Gram matrix and the dual support Gram matrix and reuse this factorization for regularization selection, prediction, and leave-one-out reliability.

Primal and dual ridge solutions. For branch b, write $\mathbf{H} = \mathbf{H}_{b,S} \in \mathbb{R}^{m\times D}$, $\mathbf{H}_Q = \mathbf{H}_{b,Q}$, and $\mathbf{Y} = \mathbf{Y}_S \in \mathbb{R}^{m\times C}$, where $m = |S|$ and $D = D_b$. For $\lambda > 0$, the equivalent primal and dual solutions are

$$\mathbf{W}^\star_\lambda = (\mathbf{H}^\top\mathbf{H} + \lambda\mathbf{I}_D)^{-1}\mathbf{H}^\top\mathbf{Y} = \mathbf{H}^\top(\mathbf{H}\mathbf{H}^\top + \lambda\mathbf{I}_m)^{-1}\mathbf{Y}, \qquad \hat{\mathbf{Y}}_Q = \mathbf{H}_Q\mathbf{W}^\star_\lambda. \tag{12}$$

We use the primal form when $D \le m$ and the dual form when $m < D$. The eigendecomposition then costs $O(r^3)$ for $r = \min(m, D)$, and forming the selected Gram matrix costs $O(mDr)$. Both forms give the same predictions for a given λ.

Shared eigendecomposition. In the primal case, let $\mathbf{H}^\top\mathbf{H} = \mathbf{U}\,\mathrm{diag}(s)\mathbf{U}^\top$, $\tilde{\mathbf{H}} = \mathbf{H}\mathbf{U}$, and $\mathbf{\Gamma} = \tilde{\mathbf{H}}^\top\mathbf{Y}$. Then

$$\mathbf{W}^\star_\lambda = \mathbf{U}\,\mathrm{diag}\Big(\frac{1}{s_j+\lambda}\Big)\mathbf{\Gamma}, \qquad \hat{\mathbf{Y}}_S = \tilde{\mathbf{H}}\,\mathrm{diag}\Big(\frac{1}{s_j+\lambda}\Big)\mathbf{\Gamma}. \tag{13}$$

In the dual case, let $\mathbf{H}\mathbf{H}^\top = \tilde{\mathbf{U}}\,\mathrm{diag}(s)\tilde{\mathbf{U}}^\top$ and $\mathbf{\Gamma} = \tilde{\mathbf{U}}^\top\mathbf{Y}$. Then

$$\mathbf{W}^\star_\lambda = \mathbf{H}^\top\tilde{\mathbf{U}}\,\mathrm{diag}\Big(\frac{1}{s_j+\lambda}\Big)\mathbf{\Gamma}, \qquad \hat{\mathbf{Y}}_S = \tilde{\mathbf{U}}\,\mathrm{diag}\Big(\frac{s_j}{s_j+\lambda}\Big)\mathbf{\Gamma}. \tag{14}$$

Here s denotes the r eigenvalues of the selected matrix, including any zero eigenvalues; the primal and dual matrices share the same nonzero spectrum. Once the selected matrix is decomposed, changing λ only changes diagonal factors.

Regularization selection. We select λ separately for each branch and support set by GCV (Golub et al., 1979). With the hat matrix $\mathbf{\Psi}_\lambda = \mathbf{H}(\mathbf{H}^\top\mathbf{H} + \lambda\mathbf{I}_D)^{-1}\mathbf{H}^\top$,

$$\mathrm{GCV}(\lambda) = \frac{\mathrm{RSS}(\lambda)/m}{(1 - \mathrm{tr}(\mathbf{\Psi}_\lambda)/m)^2}, \qquad \mathrm{tr}(\mathbf{\Psi}_\lambda) = \sum_{j=1}^{r}\frac{s_j}{s_j+\lambda}, \tag{15}$$

where $\mathrm{RSS}(\lambda) = \|\mathbf{Y} - \hat{\mathbf{Y}}_S\|_F^2$. With the quantities above,

$$\mathrm{RSS}(\lambda) = \begin{cases} \|\mathbf{Y}\|_F^2 - \displaystyle\sum_{j=1}^{D}\frac{s_j + 2\lambda}{(s_j+\lambda)^2}\|\mathbf{\Gamma}_{j,:}\|_2^2, & D \le m, \\ \displaystyle\sum_{j=1}^{m}\frac{\lambda^2}{(s_j+\lambda)^2}\|\mathbf{\Gamma}_{j,:}\|_2^2, & m < D. \end{cases} \tag{16}$$

These expressions evaluate the grid without constructing a hat matrix or a matrix of fitted values per candidate: once $\mathbf{\Gamma}$ and its row norms are formed in $O(mrC)$, each grid value costs $O(r)$. The grid contains 20 log-spaced values from $10^{-4}\bar{s}$ to $10^{4}\bar{s}$, where $\bar{s} = \max\{r^{-1}\sum_{j=1}^{r} s_j, 10^{-8}\}$. Using the smaller matrix keeps the structural zero eigenvalues of the larger matrix out of this scale.

Leave-one-out support reliability. The same factorization gives the hat diagonal

$$\psi_{ii} = (\mathbf{\Psi}_\lambda)_{ii} = \begin{cases} \displaystyle\sum_{j=1}^{D}\frac{\tilde{H}_{ij}^2}{s_j+\lambda}, & D \le m, \\ \displaystyle\sum_{j=1}^{m}\tilde{U}_{ij}^2\,\frac{s_j}{s_j+\lambda}, & m < D. \end{cases} \tag{17}$$

For fixed representations and fixed λ, the exact leave-one-out prediction for support node i is $\hat{\mathbf{y}}_i^{\mathrm{LOO}} = \mathbf{y}_i - (\mathbf{y}_i - \hat{\mathbf{y}}_i)/(1-\psi_{ii})$. The accuracy of these predictions defines $\rho_b \in [0,1]$, which the fusion head reads as a detached reliability feature. This calculation holds the representations and the GCV-selected λ fixed. It does not remove a node's label from the label-dependent inputs and does not reselect λ; the support cross-validation of Section 3.3 recomputes both.

Differentiation and numerical implementation. For fixed $\lambda > 0$, the ridge solution is a smooth function of $\mathbf{H}$, so the query loss backpropagates through the solve. With GCV selection, the discrete

minimizing grid index is not differentiated, but the selected value keeps its dependence on $\bar{s} = \max\{\|\mathbf{H}\|_F^2/r, 10^{-8}\}$ (the mean of the r eigenvalues equals the trace divided by r), so gradients also pass through this scale wherever the clamp is inactive. We compute the eigendecomposition and ridge algebra in float64, clamp negative eigenvalues caused by numerical error to zero, and cast query predictions back to the representation dtype. For numerical protection, the implementation lower-bounds both $1 - \mathrm{tr}(\mathbf{\Psi}_\lambda)/m$ and $1 - \psi_{ii}$ by 10^{-6}; the expressions above are exact before these safeguards.

C Fusion Head and Auxiliary Losses Used in Training

This appendix complements Section 3.3. The fusion head is used only during training; Section 4.4 shows that replacing it by uniform fusion changes the average accuracy by 0.1 points. Let $\hat{\mathbf{Y}} \in \mathbb{R}^{|Q|\times B\times C}$ stack the branch logits. The head reads a gradient-detached copy of $\hat{\mathbf{Y}}$ and, for each query node, three groups of features whose size depends only on B.

1. **Distance features** ($B(B-1)$ dimensions). Following Zhao et al. (2025), the B predictions of a node are treated as a point cloud and summarized by entropy-normalized affinities, with the bandwidth $\beta_{b'}$ of each anchor found by binary search to reach a target entropy:
$$p_{b|b'} = \frac{\exp(-\beta_{b'}\|\hat{\mathbf{y}}_b - \hat{\mathbf{y}}_{b'}\|^2)}{\sum_{b''\neq b'}\exp(-\beta_{b'}\|\hat{\mathbf{y}}_{b''} - \hat{\mathbf{y}}_{b'}\|^2)}. \tag{18}$$
2. **Per-branch confidence** ($3B$ dimensions): the normalized entropy, the top-1 probability, and the top-1 minus top-2 margin of each branch's softmax (Lakshminarayanan et al., 2017).
3. **Per-branch reliability** (B dimensions): the leave-one-out accuracies ρ_b of Section 3.2, broadcast across query nodes.

The concatenation is passed to a one hidden-layer MLP with LayerNorm and dropout that outputs B logits. The attention is $\boldsymbol{\alpha}_n = \mathrm{softmax}(\mathrm{logits}/\tau)$ with temperature τ, followed by dropout and, during training, branch dropout, in which each branch is zeroed with a fixed probability and the weights are renormalized. We use a convex combination rather than top-k selection; the branch dropout and the entropy penalty below keep the head from collapsing onto a single branch.

Prototypical loss. For each covariance branch, class prototypes are the support-set means of $\mathbf{H}_b$, and query nodes are classified by negative squared Euclidean distance to the prototypes (Snell et al., 2017); $\mathcal{L}_{\mathrm{proto}}$ averages this loss over the covariance branches. Because the cross-entropy gradient that reaches branch b is scaled by α_{nb}, a branch that the head down-weights receives little signal; the prototypical loss bypasses the attention and gives each encoder a direct gradient.

Attention entropy penalty. $\mathcal{L}_{\mathrm{ent}} = \frac{1}{|Q|}\sum_n \sum_b \alpha_{nb}\log\alpha_{nb}$ is the mean negative entropy of the attention.

Oracle attention loss.
$$\mathcal{L}_{\mathrm{oracle}} = -\frac{1}{|Q'|}\sum_{n\in Q'}\log\Big(\sum_b \alpha_{nb}\,\mathbf{1}\big[\arg\max_c \hat{y}_{nbc} = y_n\big]\Big), \tag{19}$$
where y_n is the label of query node n and Q' is the set of query nodes for which at least one branch is correct. This loss supervises the routing by observed correctness rather than by confidence, and it uses only labels of the source graphs.

Training loop. In every epoch, each source graph contributes one full-graph forward and backward pass. Gradients are accumulated across source graphs, scaled by the inverse number of source graphs, and a single Adam step is taken. The support set is resampled with a fixed number of labeled nodes per class at a fixed period, and the query set is the training split minus the support nodes (Table 3).

D Details of Support Cross-Validation and Node-Level Overrides

This appendix complements Section 3.3. Five stratified folds are used by default; when a class has fewer support nodes, the number of folds is reduced to the smallest support-class count among

classes with at least two examples. Support nodes of classes with a single example remain in every fitting fold and are excluded from scoring, so S_{cv} contains the remaining support nodes. Graph structure and node attributes remain available in every fold, as they are during ordinary inference, and the trained encoder parameters are fixed. When fewer than κ scored support nodes are available, $\mathcal{N}(n)$ contains all of them, and when no positive similarity is available, $q_{nb} = a_b$. The override rule of Section 3.3 nests the other selection readouts: because every q_{nb} is a convex combination of values in $[0, 1]$, taking $\delta \geq 1$ disables all overrides and recovers global selection, $b^\star_n = b_{\mathrm{g}}$, while $\delta = 0$ and $\mu = 0$ accept every strict local preference and recover per-node selection, $b^\star_n = b_{\mathrm{loc}}(n)$, up to ties between $b_{\mathrm{loc}}(n)$ and b_{g}; applying the learned attention weights of Section 3.3 to the same predictions instead recovers the training-time soft fusion. All readouts compared in Table 2 use the same full-support branch predictions, and global selection, per-node selection, and node-level overrides also share the same out-of-fold predictions, so the node-level variants require no additional model evaluations.

E HYPERPARAMETERS

Table 3 lists every configurable quantity of AnyGRAM and its value in the reported experiments. A single set of values is used for all source pools, target graphs, and ablations, and λ is reselected by GCV for every branch and support set. The learning rate, the attention temperature τ, the branch-dropout rate, the loss weights w_{proto}, w_{ent}, and w_{oracle}, the number of support nodes per class, and the distance-feature target entropy were selected by a random sweep of 30 trials (seed 0) that maximizes the validation accuracy over the the source graph (Cora). All other values, including the widths, the inference constants κ, η, δ, and μ, and the GCV grid, were fixed before the sweep. The resulting configuration is shared by every source pool, target graph, and ablation; nothing is tuned per graph, and no hyperparameter is tuned on the evaluation graphs.

Table 3: **Hyperparameters of AnyGRAM in the reported experiments.** The same values are used in every experiment.

Component	Hyperparameter	Value
Branch bank	Branches B / propagation depth k	10 / 2
Covariance operators	Projection width h	256
	Landmark count L	16
Invariant inputs	RWSE walk length ℓ	20
	Hutchinson probes n_{pr}	150
	Label-propagation hops T	3
Encoders	Encoder width h'	128
	GIN offset ϵ	learned, initialized to 0
Fusion (training only)	Hidden width / dropout / branch dropout	128 / 0.5 / 0.08927
	Attention-weight dropout	0.2
	Attention temperature τ	0.5136
	Distance-feature target entropy	1.162 bits
Readout	GCV grid	20 log-spaced points in $[10^{-4}\bar{s},\ 10^{4}\bar{s}]$
	Ridge algebra	float64, clamps at 10^{-6}
Training	Optimizer	Adam
	Learning rate / weight decay	1.91×10^{-4} / 10^{-5}
	Epochs	200
	Support nodes per class / resampling period	10 / every 20 epochs
	Projection redraw period	every forward pass
	Loss weights w_{proto} / w_{ent} / w_{oracle}	0.08967 / 0.03917 / 1.339
Inference	Projection draws E	10
	Support CV folds	5 (reduced to smallest class count ≥ 2)
	Node-level overrides κ / η / δ / μ	32 / 10 / 0.05 / 2

F ADDITIONAL RESULTS

This appendix supports (Q2) to (Q4).

Choice of source graph. Table 4 complements Sections 4.4 and 4.5. AnyGRAM trained on any of four source graphs with different domains and homophily reaches nearly the same average accuracy

on the 23 graphs outside all four source graphs, and support-validated selection exceeds the learned soft fusion for each source graph.

Table 4: **The average accuracy of AnyGRAM changes by at most 0.21 points across four source graphs.** Zero-shot accuracy (%) ↑ of AnyGRAM with support-validated selection on the 23 graphs outside all four source graphs, mean ± sample standard deviation over five seeds, for each source graph; the last column is the mean over the four source graphs. The footer rows give the average accuracy of four readouts of the same trained models.

Graph	cora	chameleon	roman-empire	wiki-cs	mean of four
actor	34.18 ± 1.06	34.43 ± 0.70	34.36 ± 0.67	33.84 ± 0.73	34.20 ± 0.79
amazon-ratings	43.17 ± 0.19	43.22 ± 0.30	43.16 ± 0.14	43.41 ± 0.15	43.24 ± 0.20
arxiv	62.66 ± 0.02	62.67 ± 0.02	62.67 ± 0.03	62.66 ± 0.03	62.67 ± 0.03
blogcatalog	73.55 ± 5.29	73.55 ± 5.29	73.55 ± 5.29	73.55 ± 5.29	73.55 ± 5.29
brazil	65.38 ± 4.71	63.08 ± 6.44	65.38 ± 7.20	66.15 ± 5.01	65.00 ± 5.84
citeseer	68.48 ± 3.73	68.96 ± 0.90	69.04 ± 1.11	68.26 ± 1.64	68.69 ± 1.85
co-cs	90.02 ± 1.53	90.02 ± 1.53	90.02 ± 1.53	90.02 ± 1.53	90.02 ± 1.53
co-physics	91.57 ± 1.73	91.84 ± 1.78	91.87 ± 1.76	91.81 ± 1.79	91.77 ± 1.77
computers	84.48 ± 1.00	84.48 ± 1.00	84.48 ± 1.00	84.48 ± 1.00	84.48 ± 1.00
cornell	79.46 ± 2.42	79.46 ± 2.42	79.46 ± 2.42	79.46 ± 2.42	79.46 ± 2.42
europe	49.63 ± 3.44	50.50 ± 3.89	50.63 ± 4.64	50.88 ± 4.99	50.41 ± 4.24
full-DBLP	73.84 ± 1.48	72.39 ± 2.69	73.28 ± 2.21	73.38 ± 1.82	73.22 ± 2.05
full-cora	61.81 ± 0.47	61.81 ± 0.47	61.81 ± 0.47	61.81 ± 0.47	61.81 ± 0.47
minesweeper	82.18 ± 0.24	82.18 ± 0.24	82.18 ± 0.24	82.18 ± 0.24	82.18 ± 0.24
photo	90.01 ± 0.77	90.01 ± 0.77	90.01 ± 0.77	90.01 ± 0.77	90.01 ± 0.77
pubmed	79.10 ± 0.00	78.16 ± 2.10	79.10 ± 0.00	79.10 ± 0.00	78.87 ± 0.53
questions	97.10 ± 0.04	97.10 ± 0.04	97.10 ± 0.04	97.10 ± 0.03	97.10 ± 0.04
squirrel	56.41 ± 2.11	56.43 ± 2.09	56.43 ± 2.09	56.41 ± 2.11	56.42 ± 2.10
texas	80.54 ± 2.96	80.54 ± 2.96	78.38 ± 3.82	78.92 ± 2.96	79.60 ± 3.18
tolokers	80.03 ± 0.17	79.94 ± 0.12	79.97 ± 0.18	80.01 ± 0.16	79.99 ± 0.16
usa	52.04 ± 2.38	51.17 ± 1.66	50.52 ± 3.03	53.26 ± 2.07	51.75 ± 2.29
wiki-attr	74.64 ± 2.45	75.84 ± 1.01	74.81 ± 2.72	75.88 ± 1.02	75.29 ± 1.80
wisconsin	87.06 ± 1.07	87.06 ± 1.07	87.06 ± 1.07	87.06 ± 1.07	87.06 ± 1.07
AVERAGE OVER THE 23 GRAPHS, BY READOUT OF THE SAME TRAINED MODELS					
Learned soft fusion	69.32 ± 1.74	70.00 ± 1.69	68.66 ± 2.40	69.51 ± 2.38	69.37 ± 2.05
Global selection	72.04 ± 1.67	71.91 ± 1.71	71.94 ± 1.82	72.14 ± 1.61	72.01 ± 1.70
Per-node selection	71.77 ± 1.62	71.67 ± 1.72	71.67 ± 1.98	71.92 ± 1.62	71.76 ± 1.74
Support-validated selection	72.06 ± 1.71	71.95 ± 1.72	71.97 ± 1.84	72.16 ± 1.62	72.04 ± 1.72

Number of source graphs. Table 5 gives the per-graph values behind Figure 2.

Table 5: **Adding source graphs leaves the accuracy of AnyGRAM within 0.28 points of its one-source value.** Zero-shot accuracy (%) ↑ on 18 fixed target graphs when trained on 1, 3, 5, 7, and 9 source graphs, mean ± sample standard deviation over five seeds.

Number of source graphs	1	3	5	7	9
amazon-ratings	43.17 ± 0.19	43.27 ± 0.24	43.31 ± 0.25	43.22 ± 0.22	43.27 ± 0.34
arxiv	62.66 ± 0.02	62.66 ± 0.03	62.67 ± 0.02	62.65 ± 0.03	62.67 ± 0.03
blogcatalog	73.55 ± 5.29	73.55 ± 5.29	73.55 ± 5.29	73.55 ± 5.29	73.55 ± 5.29
brazil	65.38 ± 4.71	64.62 ± 7.40	56.92 ± 7.40	66.15 ± 6.32	66.92 ± 6.99
chameleon	69.30 ± 1.20	69.30 ± 1.20	69.30 ± 1.20	69.30 ± 1.20	69.30 ± 1.20
citeseer	68.48 ± 3.73	69.58 ± 1.09	69.22 ± 1.04	69.06 ± 1.56	68.90 ± 0.88
co-cs	90.02 ± 1.53	89.93 ± 1.46	90.53 ± 0.74	90.42 ± 0.75	90.62 ± 0.67
co-physics	91.57 ± 1.73	91.78 ± 1.81	92.00 ± 1.71	92.07 ± 1.75	92.10 ± 1.90
cornell	79.46 ± 2.42	79.46 ± 2.42	79.46 ± 2.42	79.46 ± 2.42	79.46 ± 2.42
full-cora	61.81 ± 0.47	61.81 ± 0.47	61.81 ± 0.47	61.81 ± 0.47	61.81 ± 0.47
full-DBLP	73.84 ± 1.48	74.31 ± 2.18	74.23 ± 1.38	73.24 ± 2.96	73.57 ± 2.23
minesweeper	82.18 ± 0.24	82.18 ± 0.24	82.18 ± 0.24	82.18 ± 0.24	82.18 ± 0.23
pubmed	79.10 ± 0.00	79.10 ± 0.00	79.10 ± 0.00	79.10 ± 0.00	79.10 ± 0.00
questions	97.10 ± 0.04	97.10 ± 0.04	97.10 ± 0.04	97.10 ± 0.03	97.10 ± 0.04
squirrel	56.41 ± 2.11	56.43 ± 2.09	56.41 ± 2.11	56.41 ± 2.11	56.41 ± 2.11
wiki-attr	74.64 ± 2.45	75.88 ± 1.02	75.84 ± 0.99	74.37 ± 3.12	74.44 ± 3.07
wiki-cs	77.19 ± 0.37	77.18 ± 0.35	77.20 ± 0.38	77.18 ± 0.37	77.18 ± 0.40
wisconsin	87.06 ± 1.07	87.06 ± 1.07	87.06 ± 1.07	85.49 ± 4.07	87.06 ± 1.07
Average (18 graphs)	74.05 ± 0.62	74.18 ± 0.86	73.77 ± 0.58	74.04 ± 0.60	74.20 ± 0.83

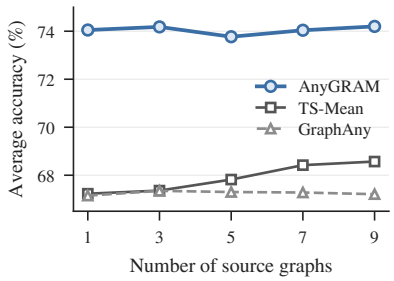


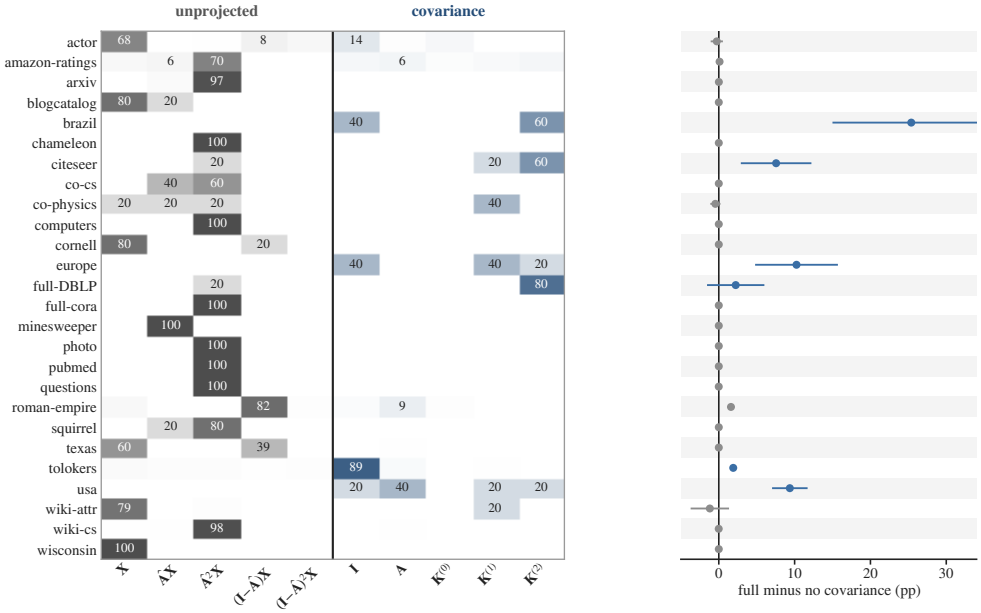
Figure 2: **AnyGRAM reaches its accuracy with one source graph.** Average accuracy (%) on 18 fixed target graphs as the number of source graphs grows; AnyGRAM in blue, baselines in gray. Per-graph values of AnyGRAM are in Table 5.

Table 6: Datasets used for training in Figure 2 and Table 5

Train-set size	Graphs
1	cora
3	cora, texas, tolokers
5	cora, texas, tolokers, photo, roman-empire
7	cora, texas, tolokers, photo, roman-empire, usa, actor
9	cora, texas, tolokers, photo, roman-empire, usa, actor, computers, europe

Branch routing. Figure 3 complements Section 4.3. The selected branch varies across graphs: unprojected branches receive nearly all query nodes on most graphs, while covariance branches receive most of them on Brazil, Europe, USA, full-DBLP, CiteSeer, and Tolokers. Rows split into multiples of 20% are consistent with the five seeds selecting different global branches without node-level overrides. Panel (b) shows that the covariance branches bring their largest paired gains on the same graphs.

Notes on the component ablations. Removing the high-pass branches or the local aggregation term changes the average by at most 0.1 points in Table 2, although their per-graph effects may be larger. Uniform fusion during training reaches 72.0%, which differs from uniformly averaging the branches of the full model at inference (70.0%): in the first case the uniform fusion shapes the encoders and support-validated selection is still applied at inference, while in the second the uniform average replaces the selection. The fixed-λ ablation replaces GCV selection with $\lambda = 10^{-3}\bar{s}$, where $\bar{s}$ is the mean eigenvalue of the support-feature Gram matrix used by the ridge solver, floored at 10^{-8}. The ablation without local aggregation removes the entire additive GIN term $\mathrm{MLP}_{\mathrm{local}}((1+\epsilon)X + A_{\mathrm{op}}X)$, where X denotes the projected invariant inputs and A_{op} is the sparse operator supplied to that encoder. For covariance-operator encoders, which receive no sparse adjacency, this removes $\mathrm{MLP}_{\mathrm{local}}((1+\epsilon)X)$. The residual connection, global covariance term where applicable, and subsequent normalization and activation are retained.



(a) Share of query nodes per branch (%).

(b) Paired gain (pp).

Figure 3: **Covariance branches are selected on a subset of the graphs and bring their largest gains there.** AnyGRAM trained on Cora and evaluated on the 26 target graphs. (a) Share of query nodes assigned to each branch after node-level overrides, pooled over five seeds; unprojected branches in gray, covariance branches in blue; shares below 5% appear as shading only. (b) Paired accuracy difference between the full model and the retrained model without covariance branches, paired within each seed; blue marks graphs where covariance branches receive most query nodes, and whiskers show one sample standard deviation across the five seeds, not confidence intervals.