CFGP: CLOSED-FORM GRAPH PROMPTING WITHOUT PER-TASK TRAINING

Anonymous authors
Paper under double-blind review

ABSTRACT

Few-shot graph learning typically proceeds by pre-training a graph neural network (GNN) on unlabeled graph data and adapting it to a labeled downstream task. *Graph prompt learning* freezes this encoder and adapts a small set of prompts instead of fine-tuning all parameters. Existing methods, however, still typically train prompt parameters for each task, which is costly and sensitive to initialization, learning rate, and early stopping under extreme label scarcity. We propose **CFGP** (Closed-Form Graph Prompting without Per-Task Training), which builds an instance-specific input prompt in closed form from class signatures derived from the labeled examples and a clustering of all instances, then reads it out with the frozen GNN and a prototype classifier. Adaptation uses no optimizer and no training objective. The same construction covers node- and graph-level tasks, differing only in how the prompt is injected, and cuts adaptation time by more than an order of magnitude at competitive accuracy.

1 INTRODUCTION

Graph neural networks are the standard tool when relational structure carries the signal that tabular models miss (Kipf & Welling, 2017; Velickovic et al., 2018; Hamilton et al., 2017; Gilmer et al., 2017; Ying et al., 2018). Labeled graph data, however, are scarce—annotating nodes or whole graphs is expensive—while unlabeled graphs are plentiful, so the dominant recipe is to pre-train an encoder with self-supervised objectives, generative or contrastive (Li et al., 2023; Grill et al., 2020; Qiu et al., 2020), and then adapt it to a downstream node- or graph-level task. Full fine-tuning is parameter-heavy and overfits under few-shot supervision, and it does not address the mismatch between pre-training objectives and the downstream classification goal (Fang et al., 2023; Liu et al., 2023). We stay in the frozen-encoder setting.

Inspired by prompt tuning in NLP (Li & Liang, 2021), *graph prompt learning* (Fang et al., 2023; Liu et al., 2023) offers a lightweight alternative: the encoder stays frozen and a prompt is inserted so that the frozen model becomes task-aligned. Existing graph prompts differ along two axes: *where the prompt acts*—at the input, inside the encoder's layers, or on the output representations—and *whether the prompt is trained per task* (Table 1). Almost all published methods still optimize the prompt with per-task gradients.

Table 1: Representative graph prompts: where the prompt acts, and whether the prompt is trained per task. The three prompt spaces follow UniPrompt (Huang et al., 2025). EdgePrompt is listed as layer-wise following its own description, whereas UniPrompt figures edge prompts as input-level.

Methods	Prompt space	Per-task prompt
All-in-One (Sun et al., 2023), GPF/GPF+ (Fang et al., 2023), SUPT (Lee et al., 2024)	input	trained
EdgePrompt (Fu et al., 2025a)	layer-wise	trained
GPPT (Sun et al., 2022), GraphPrompt (Liu et al., 2023), ProNOG (Yu et al., 2025a), DAGPrompT (Chen et al., 2025)	representation	trained

Continued on next page

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.

Table 1: Continued

Methods	Prompt space	Per-task prompt
GCoT (Yu et al., 2025b), MSGCOT (Zheng et al., 2026)	input, iterative	trained
CFGP (ours)	input	none

The per-task loop is both slow and unreliable: a 1-shot task supplies one labeled example per class, yet the prompt is fitted for hundreds of epochs (Yu et al., 2025b; Zheng et al., 2026), and the fit depends on initialization, learning rate, and early stopping. UniPrompt (Huang et al., 2025) reports a related finding: a representation-level prompt is no more powerful than linear probing. Fitting GCoT's condition network or MSGCOT's coarsening module on one labeled example per class is therefore closer to training a small classifier than to prompting.

Under one labeled example per class, optimizing a prompt is unstable. **CFGP** (Closed-Form Graph Prompting without Per-Task Training) instead constructs it: the labeled examples supply class signatures, and a clustering of all instances—labeled or not—assigns each instance to those signatures. Combining the two in closed form yields a deterministic, instance-specific input prompt with no gradient step; the frozen GNN then propagates it, with no per-task training.

We make three contributions. (1) **A closed-form prompt.** Class signatures come from the supporting data, and instance assignment from a spherical k-means clustering of the frozen embeddings. The adaptation budget is L frozen forward passes. (2) **One construction for two task types.** The same construction serves node and graph classification; the two differ only in how the prompt is written into the input and, on graphs, in the pooling that maps a graph's node embeddings to one vector. Successive rounds use $K_\ell = \mathrm{fanout}^\ell$ clusters, a label-free substitute for MSGCOT's learned multi-scale hierarchy. (3) **Evidence and controls.** On the official 1-shot suite, CFGP is the strongest method on the node datasets except Pubmed, and leads MSGCOT under balanced Acc on seven of eight datasets (Pubmed is the exception); on COX2 we make no method claim against the full baseline set. Freezing the cluster count, turning the prompt off, and shuffling it across instances each remove the gain (§4.5).

2 Related Work

2.1 Graph Pre-training

Graph pre-training learns transferable representations from unlabeled graph data, in generative, contrastive, and multi-task families (Fu et al., 2025b). Link prediction—predicting whether an edge exists between two nodes—is the simplest objective, and the one our frozen encoders are pre-trained with (Appendix A). GraphMAE masks features and reconstructs them (Hou et al., 2022) and DGI maximizes mutual information between a node and a graph summary (Velickovic et al., 2019); MVGRL, GRACE, GCA, GraphCL and JOAO contrast augmented views (Hassani & Khasahmadi, 2020; Zhu et al., 2020; 2021; You et al., 2020; 2021), BGRL avoids negative pairs (Thakoor et al., 2022), and multi-task pre-training combines several such objectives in one stage (Fu et al., 2025b); data-augmentation surveys catalog how unlabeled graphs are transformed beforehand (Ding et al., 2022). These objectives are task-agnostic and do not address the mismatch with the downstream task. Prompting is the adaptation step that is supposed to close it, without unfreezing the encoder.

2.2 Graph Prompt Learning

Following UniPrompt (Huang et al., 2025), we group graph prompts by the space in which they act. *Input-level* prompts rewrite the input before the encoder (GPF, GPF+, All-in-One (Fang et al., 2023; Sun et al., 2023)). *Layer-wise* prompts act inside the encoder's layers (EdgePrompt (Fu et al., 2025a))—a placement that needs intermediate states a black-box encoder does not expose. *Representation-level* prompts act on the output representations, often as part of the classifier (GraphPrompt, GPPT, ProNOG, DAGPrompT (Liu et al., 2023; Sun et al., 2022; Yu et al., 2025a; Chen et al., 2025)). Table 1 lists these spaces; all of them still train a prompt.

GCoT brings chain-of-thought (CoT) prompting (Wei et al., 2022) to graphs: it fuses the encoder's hidden layers into a "thought", trains a condition network (CondNet) on it to generate node-specific prompts, and writes those prompts back into the feature matrix (Yu et al., 2025b). MSGCOT adds a low-rank coarsening network that produces hierarchical basis vectors, integrated from coarse to fine across steps (Zheng et al., 2026). Both instantiate the prompt from scratch on every 1-shot task. UniPrompt further shows that a *representation-level* prompt on frozen embeddings is equivalent to linear probing, and argues that input-level prompts are the more useful design. A parallel line couples graphs with large language models (Chen et al., 2024; Wang et al., 2024) or studies zero-shot transfer across attribute domains (Shen et al., 2025); we stay in the text-free, frozen-GNN prompt setting. CFGP prompts at the input as well, but trains no prompt.

3 METHOD

CFGP constructs an instance-specific *input* prompt in closed form and trains no prompt parameters; a frozen GNN re-encodes the prompted input and a cosine prototype head reads it out.

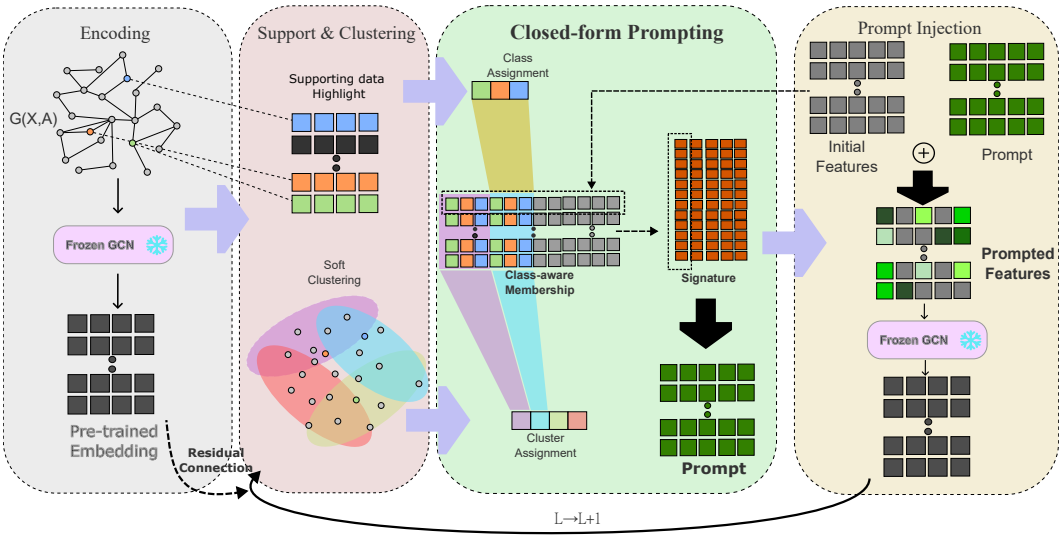


Figure 1: CFGP in one round. **Encoding**: a frozen GCN embeds the unprompted input as the unprompted embedding $\boldsymbol{H}_0$. **Support & Clustering**: the supporting data fix class responsibility $\boldsymbol{R}$ via a cosine prototype head; spherical k-means on those same embeddings yields the label-free membership $\boldsymbol{S}^{\mathrm{km}}$. **Closed-form Prompting**: the outer product of $\boldsymbol{R}$ and $\boldsymbol{S}^{\mathrm{km}}$ is the class-aware membership $\boldsymbol{S}$ over $C \cdot K_\ell$ channels; a membership-weighted signature $\tilde{\boldsymbol{M}}$ gives the instance-specific prompt $\boldsymbol{P} = \boldsymbol{S}\tilde{\boldsymbol{M}}$. **Prompt Injection**: $\boldsymbol{P}$ is written into the input features and re-encoded by the same frozen GCN to $\boldsymbol{H}'$; a residual mix with $\boldsymbol{H}_0$ produces $\boldsymbol{Z} = \mathrm{RowNorm}(w\boldsymbol{H}' + \boldsymbol{H}_0)$, and the loop repeats for L rounds with no backward pass.

3.1 PROBLEM SETTING

A graph is $\mathcal{G} = (\boldsymbol{V}, \boldsymbol{E}, \boldsymbol{X})$ with adjacency $\boldsymbol{A}$ and node features $\boldsymbol{X} \in \mathbb{R}^{N \times F}$, where N counts nodes. Graph classification uses a collection of graphs with no edges between them, concatenated into one disjoint union. $C = |\mathcal{Y}|$ is the number of classes. A message-passing encoder is pre-trained (Appendix A) and then held fixed. Standard graph prompting instead trains a prompt module on a set of labeled examples, with the encoder held fixed, by maximizing

$$\mathbb{E}_{(i,y_i)\in\mathcal{L}}\big[\log p(y_i \mid \boldsymbol{X}, \boldsymbol{A}, i; \mathrm{prompt})\big] \tag{1}$$

with respect to the prompt parameters. CFGP does not solve Eq. (1): it replaces the optimization with closed-form tensor operations.

Each task is 1-shot: one labeled example per class—the *supporting data* $\mathcal{D}_s$—and a test set to predict—the *query data* $\mathcal{D}_q$ (Sun et al., 2023). Node and graph tasks share the construction below

and differ only in how the prompt is written into the input, Eq. (8): the node branch adds it to the node feature, while the graph branch mixes it into that same feature and pools the re-encoded node embeddings back to one vector per graph.

3.2 CLOSED-FORM PROMPT

The core objects are the responsibility $\boldsymbol{R}$, the k-means membership $\boldsymbol{S}^{\mathrm{km}}$, their outer product $\boldsymbol{S}_{i,(c,k)} = R_{ic}\, S^{\mathrm{km}}_{ik}$ over classes and clusters, the membership-weighted signature $\tilde{\boldsymbol{M}}$, and the instance-specific prompt $\boldsymbol{P} = \boldsymbol{S}\tilde{\boldsymbol{M}}$. Figure 1 traces their order of computation—$\boldsymbol{H}_0$, responsibility, membership, signature, prompt, inject, re-encode, residual $\boldsymbol{Z}$, cosine prototype—and the last step returns to the first for L rounds. Two temperatures are fixed across all datasets (Appendix A): the clustering temperature τ of Eq. (4) and the readout temperature τ_u of Eq. (3).

Unprompted embedding and first responsibility. Encode the unprompted input once, with the encoder held fixed,

$$\boldsymbol{H}_0 = \mathrm{GNN}(\boldsymbol{X}, \boldsymbol{A}). \tag{2}$$

We call $\boldsymbol{H}_0$ the *unprompted embedding*, because every round's residual mix is taken relative to it. On graphs the encoder returns one node embedding per node, and we pool them by mean and standard deviation into one vector per graph, which is $\boldsymbol{H}_0$ for that graph. Throughout, $\mathrm{RowNorm}$ is the row-wise ℓ_2 normalization and $\boldsymbol{Z}_0 = \mathrm{RowNorm}(\boldsymbol{H}_0)$. The first responsibility $\boldsymbol{R}^{(0)}$ is the cosine prototype head applied to $\boldsymbol{Z}_0$, not to a prompted embedding. Writing the prototype of class c as the mean of its supporting rows, $\boldsymbol{q}_c$, and $\boldsymbol{z}_i$ as the i-th row of the current embedding,

$$\boldsymbol{q}_c \;=\; \frac{1}{|\mathcal{D}^c_s|}\sum_{i\in\mathcal{D}^c_s} \boldsymbol{z}_i, \qquad \boldsymbol{R}_{ic} \;\propto\; \exp\!\big(\boldsymbol{z}_i \cdot \boldsymbol{q}_c/\tau_u\big), \tag{3}$$

Rows that belong to the labeled examples are replaced by the one-hot label. Test labels are never read at any later step.

One round. The round index is $\ell = 0, \ldots, L-1$. Everything below is computed once per round. L and fanout are external scalars, one pair per dataset (Appendix A); they are not trained, though they are selected per dataset. Round ℓ uses $K_\ell = \mathrm{fanout}^\ell$ clusters, hence $C \cdot K_\ell$ channels, so successive rounds operate at progressively finer granularity; fanout is MSGCOT's coarsening ratio (Zheng et al., 2026) in reverse, since their hierarchy shrinks the graph while ours grows the number of channels. If $K_\ell = 1$, the membership is the responsibility itself, $\boldsymbol{S}^{(\ell)} = \boldsymbol{R}^{(\ell)}$. If $K_\ell > 1$, spherical k-means (k-means with cosine similarity) on the same rows as $\boldsymbol{Z}_0$ yields a label-free soft assignment

$$S^{(\ell),\mathrm{km}}_{ik} \;=\; \mathrm{softmax}_k\Big(\boldsymbol{z}_{0,i} \cdot \boldsymbol{\mu}^{(\ell)}_k \,/\, \tau\Big), \tag{4}$$

and the class-aware membership is their outer product,

$$\boldsymbol{S}^{(\ell)}_{i,(c,k)} \;=\; R^{(\ell)}_{ic}\, S^{(\ell),\mathrm{km}}_{ik}, \qquad \boldsymbol{S}^{(\ell)} \in \mathbb{R}^{N\times(C\cdot K_\ell)}. \tag{5}$$

Here the rows are the instances being classified: nodes for node tasks, graphs for graph tasks. The clustering is recomputed from $\boldsymbol{Z}_0$ and the current responsibility; it is not a learned hierarchy and not a stored chain-of-thought.

On graphs the responsibility and the k-means are per graph. Before the signature, each graph's membership row is copied onto every node embedding of that graph; write $\boldsymbol{S}^{\mathrm{node},(\ell)}$ for that copy, which is the membership the signature uses. Nodes of one graph therefore share its class weights and keep their own input features; after the encoder they are pooled back to one vector per graph, and the next responsibility is per graph again.

Signature, prompt, injection. Let $\boldsymbol{u}_i$ be the feature the encoder will read: the node feature, on node and graph tasks alike. We write the prompt into $\boldsymbol{u}$ rather than into $\boldsymbol{H}_0$, so that it enters the frozen encoder rather than editing its output. The signature is the membership-weighted centroid over all rows—node embeddings on graphs—labeled ones included. In the reported runs $\gamma = 1$, leaving the weights unsharpened,

$$\boldsymbol{m}^{(\ell)}_{(c,k)} \;=\; \frac{\sum_i \big(\boldsymbol{S}^{(\ell)}_{i,(c,k)}\big)^\gamma\, \boldsymbol{u}_i}{\sum_i \big(\boldsymbol{S}^{(\ell)}_{i,(c,k)}\big)^\gamma}, \qquad \boldsymbol{M} \in \mathbb{R}^{CK_\ell\times F}, \tag{6}$$

collected as rows of $\boldsymbol{M}$, one per channel. Centering and writing the membership back gives one prompt row per row of $\boldsymbol{S}$—per node embedding on graphs:

$$\tilde{\boldsymbol{M}} = \boldsymbol{M} - \mathbf{1}\bar{\boldsymbol{m}}^{\top}, \qquad \boldsymbol{P} \;=\; \boldsymbol{S}\,\tilde{\boldsymbol{M}}, \qquad \widehat{\boldsymbol{P}}_{ij} \;=\; \boldsymbol{P}_{ij} \,/\, \max_{j'} |\boldsymbol{P}_{ij'}|. \tag{7}$$

Here $\bar{\boldsymbol{m}}$ is the unweighted mean of the signature rows of $\boldsymbol{M}$, $\mathbf{1}$ is a column of ones, and the division rescales each prompt row so its largest entry is ± 1.

The readout therefore lives in $\boldsymbol{H}_0$ while the prompt lives in $\boldsymbol{u}$: Eq. (6) uses $\boldsymbol{S}$ only as weights over $\boldsymbol{u}_i$, so the prompt has the dimension of the input features, not of the frozen embedding, and the frozen encoder can carry the task signal in Eq. (9). The prompt is not on a fixed scale, so we normalize each prompt row to unit ℓ_2 norm and scale it by $\|\boldsymbol{u}_i\|_2$, entering $\boldsymbol{u}_i$ at the scale of the feature it is written into. On nodes the write is additive; on graphs it mixes the node feature with the prompt,

$$\boldsymbol{u}'_i = \begin{cases} \boldsymbol{u}_i + \lambda\, \|\boldsymbol{u}_i\|_2\, \mathrm{RowNorm}(\widehat{\boldsymbol{P}})_i & \text{nodes,} \\ (1-\lambda)\, \boldsymbol{u}_i + \lambda\, \|\boldsymbol{u}_i\|_2\, \mathrm{RowNorm}(\widehat{\boldsymbol{P}})_i & \text{graphs.} \end{cases} \tag{8}$$

λ is an external scalar; $\lambda = 0$ leaves $\boldsymbol{u}_i$ unchanged, the unprompted readout. The frozen encoder then runs on the updated features,

$$\boldsymbol{H}' = \mathrm{GNN}(\boldsymbol{X}', \boldsymbol{A}), \tag{9}$$

where $\boldsymbol{X}'$ collects the updated rows $\boldsymbol{u}'_i$.

On graphs, the node embeddings of $\boldsymbol{H}'$ are pooled the same way into one vector per graph, which we continue to write as $\boldsymbol{H}'$. A residual mix keeps the result near $\boldsymbol{H}_0$, which has the same shape,

$$\boldsymbol{Z} = \mathrm{RowNorm}\big(w\, \boldsymbol{H}' + \boldsymbol{H}_0\big). \tag{10}$$

w is an external scalar setting how far the round moves from $\boldsymbol{H}_0$: larger w weights $\boldsymbol{H}'$ more heavily, smaller w stays closer to $\boldsymbol{H}_0$. This $\boldsymbol{Z}$ is not the normalization $\boldsymbol{Z}_0$ of the previous paragraph. The next responsibility $\boldsymbol{R}^{(\ell+1)}$ is Eq. (3) on this $\boldsymbol{Z}$, with the labeled rows replaced by the one-hot label again; the next round reads $\boldsymbol{R}^{(\ell+1)}$, not $\boldsymbol{Z}$ itself. After round $L-1$, $\boldsymbol{Z}$ is the embedding that is classified.

Table 6 shows that the closed-form gain needs both $K{>}1$ clustering and an instance-aligned $\boldsymbol{P}$; it does not recover MSGCOT's learned assignment or CondNet. Appendix D records the role alignment and a local linearization remark.

3.3 Adaptation complexity

CFGP has no backward pass or optimizer step: adaptation is the unprompted pass plus L frozen re-encodes and closed-form tensor operations. Trainable prompts are instead fit from scratch for up to E_{ep} epochs per task—the per-task epoch cap of Appendix D. At equal encoder cost per forward pass, the leading ratio is E_{ep}/L, amplified by the missing prompt backward pass. This is why adaptation is fast; it is not a matched-epoch FLOPs certificate. Asymptotics and the Cora worked example are in Appendix D and Table 3; the measured wall-clock is Table 2.

4 Experiment

We evaluate CFGP on the 1-shot suite for both node and graph classification, then reuse the same 1-shot configuration $(\mathrm{fanout}, w, \lambda, L)$ at $k{=}1$–7 on Cora and MUTAG. Setup details, the runtime measurement, and the oracle ceiling are in Appendices A, E and G.

4.1 Setup

Datasets. We use eight benchmarks. Cora, Citeseer, and Pubmed are citation graphs, and Photo is a co-purchase graph; the four are node tasks. MUTAG, COX2, and BZR are molecular graphs, and PROTEINS is a protein-structure collection; the four are graph tasks. Splits and frozen checkpoints are in Appendix A.

Evaluation Setup. Node tasks label the nodes of a single graph, and graph tasks label each graph in a disjoint union. Following MSGCOT (Zheng et al., 2026), we adopt its official 1-shot evaluation on the MSGCOT splits, using tasks 0–99 and reporting mean±std over tasks. Each task provides one labeled example per class and an unlabeled test set. We call these the supporting data $\mathcal{D}_s$ and the query data $\mathcal{D}_q$ (§3.1). Test labels are never used, and the encoder is frozen throughout. As in MSGCOT, the signature is built over all instances of the graph, so evaluation instances contribute their features—never their labels—to $\boldsymbol{M}$; MSGCOT likewise embeds the full $(\boldsymbol{X}, \boldsymbol{A})$ before slicing into labeled and test examples.

CFGP has one configuration $(\mathrm{fanout}, w, \lambda, L)$—the four external scalars of §3—chosen once per dataset and shared by all 100 tasks; the grid and the selected tuples are in Appendix A. These splits provide no validation set, as in MSGCOT, so this choice is made on the evaluation tasks. A single shared choice with no per-dataset selection is the no-tuning lower bound of the same method (balanced 58.43, ahead of MSGCOT on seven of eight at $r{=}8$).

Baselines and Metrics. We compare CFGP against the single-step prompt baselines and GCoT, all of which fit their prompt from scratch on every task (Table 4). These methods and GCoT are our re-run on those same splits (Table 2); MSGCOT is our re-run under its published Table 7 configuration (node $r{=}8$, graph $r{=}1$). As non-prompt reference points we also report GCN, GAT, LP and DGI/Infograph, all with a linear head (Appendix B); they carry no method claim. Our accuracy metric is raw accuracy (raw Acc); on graphs we additionally report balanced accuracy (balanced Acc), the unweighted mean of per-class recalls, and a constant majority predictor as a reference.

4.2 Adaptation speed

CFGP has no per-task prompt training, so adaptation is the unprompted pass plus L frozen re-encodes (§3.3), and closed-form tensor operations. Table 2 times each method on a single RTX 4090, one process per dataset, including interpreter start, data load, and checkpoint load.

Table 2: Wall-clock on one RTX 4090, official 1-shot×100 (tasks 0–99). Rows: methods; columns: datasets (node / graph). One process per dataset; start-up included for every method. Single-step prompt baselines (†): MSGCOT encoder, cosine prototype head, 200/400 epochs with no early stop. MSGCOT / GCoT: our re-run under their official scripts, with early stopping. Not a matched-epoch or matched-backbone FLOPs study (Appendix E). **Bold**: best in the column. The last row is CFGP's time as a percentage of the runner-up's (Appendix E). MSGCOT and GCoT totals are reported in minutes.

	Node				Graph				
Method	Cora	Citeseer	Photo	Pubmed	MUTAG	COX2	BZR	PROTEINS	Total
All-in-One†	55.0 s	56.2 s	60.1 s	53.6 s	127.6 s	104.1 s	99.3 s	99.4 s	10.9 min
GPF†	61.3 s	60.1 s	62.0 s	58.5 s	124.8 s	129.0 s	108.7 s	122.3 s	12.1 min
GPF+†	68.8 s	71.8 s	71.7 s	74.3 s	147.6 s	136.6 s	145.0 s	107.6 s	13.7 min
GraphPrompt†	61.5 s	61.3 s	62.1 s	68.9 s	123.1 s	114.9 s	90.2 s	83.3 s	11.1 min
DAGPrompT†	77.8 s	75.2 s	74.7 s	77.6 s	148.4 s	149.8 s	107.7 s	112.7 s	13.7 min
EdgePrompt†	87.1 s	85.6 s	104.7 s	117.5 s	311.2 s	347.5 s	278.8 s	400.5 s	28.9 min
EdgePrompt+†	96.0 s	112.8 s	130.0 s	148.7 s	465.0 s	504.9 s	456.3 s	432.1 s	39.1 min
SUPT†	94.5 s	99.6 s	94.6 s	93.7 s	192.6 s	192.9 s	205.7 s	199.7 s	19.6 min
MSGCOT	198.0 s	202.0 s	386.7 s	457.7 s	204.8 s	79.9 s	70.5 s	368.2 s	32.8 min
GCoT	78.5 s	69.6 s	225.3 s	48.5 s	619.9 s	753.5 s	623.2 s	421.7 s	47.3 min
CFGP	**4.5** s	**4.6** s	**4.9** s	**4.3** s	**4.3** s	**4.8** s	**4.7** s	**6.3** s	**38.4** s
time as % of 2nd best	8.2%	8.2%	8.2%	8.9%	3.5%	6.0%	6.7%	7.6%	5.9%

On the eight-dataset suite CFGP finishes in 38.4 s; every method that fits a prompt per task takes 10.9–47.3 min, and MSGCOT, the strongest baseline in accuracy, is $51\times$ slower. In every column CFGP reduces adaptation time by 91.1–96.5% relative to the fastest non-CFGP method. It is not an equal-compute or equal-backbone study; CFGP's advantage is exactly that it has no per-task training loop.

Cora makes that gap concrete. Table 3 counts trainable prompt parameters as the entries of every tensor with `requires_grad=True` in the released downstream module—the frozen encoder is not counted—and places that count next to the Cora wall-clock; dimensions and frozen encoder are in Appendix A.

Table 3: Cora as a worked example (N=2708, F=1433, C=7, d=256, T=100). Trainable params: released prompt module; frozen GCN not counted. Walls from our re-run in Table 2. †: single-step prompt baselines (200 epochs, no early stop). Not a matched-epoch FLOPs study.

Method	Trainable params	Per-task adaptation	Cora time
CFGP	0	unprompted pass + L=2 re-encodes, closed form, no Adam	4.5 s
GraphPrompt†	256	200 Adam epochs on cached H	61.5 s
GPF+†	10.3K	200 Adam epochs on cached H	68.8 s
MSGCOT	62.3K	200 epochs of GNN + coarsen + CondNet (fwd+bwd)	198.0 s
GCoT	217.8K	$\leq$50 epochs of 3-layer GNN + CondNet (fwd+bwd)	78.5 s

CFGP is fast because it trains no prompt: no prompt backward pass enters adaptation (§3.3), while trainable prompts are fit from scratch on every task. Shrinking the prompt does not remove that loop: the smallest prompt in Table 3 still costs an order of magnitude more wall-clock than CFGP on Cora, because the cost is set by the number of Adam epochs, not by the parameter count. The reported prompt size depends on whether one counts only the bottleneck parameters or the full released prompt module; under either convention it numbers in the thousands per task, and CFGP has none. That is the E_{ep}/L contrast of §3.3 ($200/2$ and $\leq 50/2$ on Cora); Appendix D expands the asymptotics and Appendix E gives the timing details.

4.3 Accuracy

Table 4 reports raw Acc and Table 5 balanced Acc, both on the same CFGP predictions; Table 9 gives balanced Acc for the full method set and Table 10 the per-class recalls (Appendix C). On raw Acc CFGP is the best trained method on five of the eight datasets, but its macro-average (61.15) is below MSGCOT's (62.65), because the graph-side average is set by the fixed test-class ratio rather than by per-class performance. Balanced Acc removes that confound: CFGP averages 60.13 and leads MSGCOT on seven of eight datasets.

Table 4: 1-shot×100 raw Acc (%). Rows: methods; columns: datasets (node / graph). Single-step prompt baselines (†) and GCoT: our re-run on the official MSGCOT splits (same splits as Table 2). MSGCOT: our re-run under the published Table 7 configuration (node r=8, graph r=1; Appendix A). CFGP: the reported configuration (Table 7). **Bold**: best in the column. Underline: second best.

	Node				Graph			
Method	Cora	Citeseer	Photo	Pubmed	MUTAG	COX2	BZR	PROTEINS
All-in-One†	55.66 ±15.73	39.54 ±9.63	52.93 ±8.88	50.22 ±11.49	62.46 ±14.83	53.81 ±15.64	47.69 ±12.73	54.11 ±9.43
GPF†	57.34 ±14.06	44.11 ±8.62	65.57 ±10.71	56.72 ±11.56	63.32 ±15.01	52.59 ±10.07	48.33 ±10.71	53.99 ±8.83
GPF+†	57.38 ±13.72	44.68 ±8.23	66.48 ±10.39	57.19 ±11.34	62.92 ±15.02	52.08 ±8.90	49.56 ±9.74	54.06 ±6.28
GraphPrompt†	58.42 ±13.28	45.83 ±8.56	67.53 ±10.25	63.37 ±9.76	59.90 ±14.76	51.42 ±4.31	51.26 ±8.93	53.42 ±5.96
DAGPrompT†	58.01 ±13.58	45.72 ±8.70	67.25 ±10.35	62.86 ±10.02	61.46 ±15.00	52.21 ±8.72	49.96 ±9.43	54.83 ±9.04
EdgePrompt†	57.18 ±14.37	44.02 ±8.69	64.91 ±10.58	49.23 ±12.68	62.82 ±14.60	53.85 ±15.32	47.42 ±13.39	55.00 ±10.00
EdgePrompt+†	57.26 ±14.08	44.16 ±8.71	65.20 ±10.33	57.55 ±11.00	61.67 ±14.89	53.80 ±14.49	47.95 ±12.49	55.22 ±9.31
SUPT†	57.24 ±13.69	43.90 ±8.43	65.49 ±10.65	57.31 ±10.63	63.04 ±15.42	51.24 ±8.31	50.59 ±10.29	54.74 ±8.31
GCoT	59.18 ±14.85	46.25 ±9.04	65.92 ±10.77	63.84 ±9.48	60.34 ±14.70	52.09 ±12.34	54.45 ±16.19	55.77 ±10.46
MSGCOT	60.58 ±16.11	49.21 ±10.17	66.73 ±10.49	**64.15 ±10.88**	57.41 ±16.16	72.99 ±7.83	74.68 ±7.05	55.48 ±10.58
majority	39.00	23.00	26.00	43.00	**66.67**	**78.28**	**78.91**	**59.59**
CFGP	**68.19 ±17.18**	**52.54 ±11.40**	**71.08 ±11.48**	63.42 ±10.15	64.46 ±15.27	53.24 ±9.39	58.99 ±13.23	57.25 ±10.09

Node classification. On nodes the two metrics agree, and both are informative. CFGP is the best raw Acc on three of the four node datasets—Cora, Citeseer and Photo—and loses only Pubmed, where MSGCOT and GCoT are ahead by less than a point. The largest margin is on Cora (68.19 against MSGCOT's 60.58).

Graph classification. On graphs, raw Acc does not rank methods. The MSGCOT splits freeze the test counts (MUTAG 33.3/66.7%, COX2 78.3/21.7%, BZR 78.9/21.1%, PROTEINS 59.6/40.4%), so a constant majority predictor already matches or beats every learned column in Table 4, trainable prompts included. Balanced Acc removes the confound. CFGP's macro recall is above MSGCOT's on all four graph datasets, and on seven of eight overall, against the trained prompt at both values of r we report for it: r=1, the value its paper reports, and r=8, which is stronger on some graph datasets.

Table 10 shows why. CFGP's per-class recall stays away from a one-class collapse, whereas the trained prompt on COX2 and BZR is skewed by more than 5:1—at r=1 their balanced Acc of 50.27 and 50.11 is an average of recalls 90.4/10.1 and 92.6/7.6. GCoT, our re-run on all four graph datasets, is a control whose predictions do not mirror the fixed test-class ratio. COX2 itself supports no method claim (Appendix B).

Table 5: Balanced Acc (%) under the same 1-shot×100 splits as Table 4, for CFGP and MSGCOT at the two values of r we report. MSGCOT: our reproduction at r=1 and r=8 (Table 7 coarsening; node default is r=8). CFGP: same predictions as Table 4. Mean: macro-average over the eight datasets, each dataset weighted equally. **Bold**: best of the rows shown. Underline: second best of the rows shown. The full table, with every method of Table 4, is Table 9; on COX2 the best balanced Acc there is GPF's 51.10, which is why that dataset carries no method claim.

	Node				Graph			
Method	Cora	Citeseer	Photo	Pubmed	MUTAG	COX2	BZR	PROTEINS
MSGCOT (r=1)	47.75	40.78	71.60	66.00	53.35	50.27	50.11	54.70
MSGCOT (r=8)	57.64	46.02	73.23	**66.82**	54.72	50.19	49.13	51.47
CFGP	**64.41**	**48.52**	**78.44**	65.98	**60.66**	**50.94**	**55.82**	**56.25**

Table 5 reports balanced Acc for the same predictions. CFGP leads MSGCOT at both values of r on seven of the eight datasets: on nodes by 6.77 points on Cora (64.41 against 57.64 at r=8), 2.50 on Citeseer and 5.21 on Photo, and on all four graph datasets (MUTAG 60.66 against 54.72, COX2 50.94 against 50.19, BZR 55.82 against 49.13, PROTEINS 56.25 against 51.47). Pubmed is the single exception at both values of r (66.00 and 66.82 against CFGP's 65.98). The macro-average is 60.13, against 56.15 at r=8 and 54.32 at r=1.

4.4 Few-shot scaling

The main results are strictly 1-shot. A remaining question is whether the ranking is an artifact of fitting a prompt on a single labeled instance. Figure 2 transfers the 1-shot configuration to k=1–7 on one node dataset (Cora) and one graph dataset (MUTAG), without re-tuning; all three methods share the frozen MSGCOT encoder. The k=1 cells are this matched-encoder run, not the reported numbers of Table 4. Splits for each k are in Appendix A.

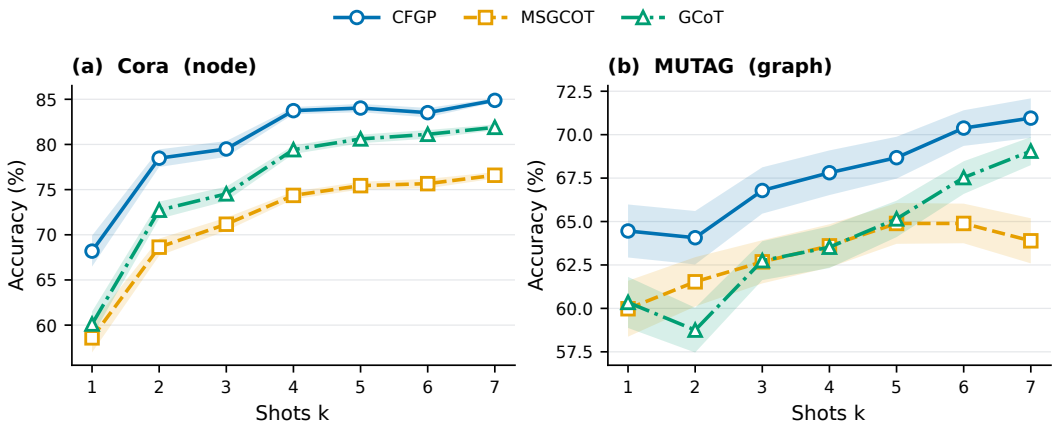


Figure 2: Raw Acc (%) versus shots k on Cora (node) and MUTAG (graph). Mean over tasks 0–99; the shaded band is the SEM. Shared frozen MSGCOT encoder; CFGP keeps the 1-shot configuration of Table 7 and does not retune. The k=1 cells are the matched-encoder reproduction, not Table 4.

On Cora, CFGP is the top curve at every k: 68.19 at 1-shot and 84.88 at 7-shot, remaining 7.9–9.9 points above MSGCOT. GCoT, given the same 2-layer encoder, sits between them. On MUTAG, CFGP remains highest at every k (64.46 to 70.95), while GCoT and MSGCOT stay within 0.4 points

of each other except at $k{=}2$, with GCoT pulling ahead from $k{=}6$. The same 1-shot configuration therefore transfers without retuning.

4.5 Ablations

We ablate three pieces of the construction under the oracle configuration of Appendix G, reporting Δ Acc relative to FULL (Table 6). **A1** freezes $K{\equiv}1$ for all L rounds, so every round keeps a single cluster and the multi-scale assignment is removed. **A2** sets $\lambda{\equiv}0$, which skips prompt injection and equals Frozen-prototype. **A3** keeps the closed-form $\boldsymbol{P}$ but shuffles its rows across instances, so each instance receives another instance's prompt.

On Cora and Citeseer, A1 and A2 each cost about 6–10 points and land near Frozen-prototype, so $K{>}1$ clustering is needed for $\boldsymbol{P}$ to help. A3 drops further (Cora -39.15, Citeseer -15.78). On COX2 the FULL margin over Frozen-prototype is under one point, so A1–A3 stay within -0.57 to -0.92. On BZR that margin is large (-9.23 under A2), but A1 matches A2: without clustering the prompt adds nothing.

Table 6: Δ Acc (pp) vs. FULL under the oracle configuration, $\mathbf{b}{=}\mathbf{0}$, on two node and two graph datasets.

	Node		Graph	
Variant	Cora	Citeseer	COX2	BZR
A1	-9.81	-7.32	-0.57	-9.23
A2	-9.56	-6.39	-0.92	-9.23
A3	-39.15	-15.78	-0.60	-9.71

5 Conclusion

CFGP replaces per-task prompt training with a closed-form construction. On the official 1-shot suite it matches or exceeds trained prompts on accuracy—best trained method on five of eight datasets, ahead of MSGCOT under balanced Acc on seven—while finishing the full run in $38.4\,\mathrm{s}$ against $10.9\,\mathrm{min}$ for the fastest baseline that still fits a prompt. The method uses a frozen encoder, a prototype readout, and a clustering of its embeddings; node and graph tasks share the same construction. Ablations (§4.5) show that the gain needs $K{>}1$ clustering and an instance-aligned prompt (Table 6).

Two limits remain. The four external scalars are chosen per dataset on the evaluation tasks, because the splits provide no validation set; a selection rule that avoids the test tasks is still missing. Under one shared configuration with no per-dataset selection, however, balanced Acc remains ahead of MSGCOT on seven of eight datasets at $r{=}8$ (58.43). Where the frozen embeddings do not separate the classes—Pubmed, and some graph tasks at or below the majority predictor—the closed-form prompt cannot invent the missing signal.

REPRODUCIBILITY STATEMENT

Official 1-shot tasks 0–99, frozen MSGCOT checkpoints, and raw CFP accuracy (not $\max(\text{base}, \text{cfp})$) are specified in Appendix A. Wall-clock numbers in Table 2 are one serial pass on a single RTX 4090; the timing is Appendix E. The reported configuration is the per-dataset value of Table 7, selected over the grid stated in Appendix A; the graph-side choices listed in Table 7 are identical across the four graph datasets. Those per-dataset 4-tuples, applied together with the graph-side choices listed in Table 7, determine every reported cell. Figure 2 transfers that configuration to k=1–7 on Cora and MUTAG with $\mathbf{b}=\mathbf{0}$ and a shared MSGCOT encoder (Appendix A). The oracle-tuned grid is Table 12 with $\mathbf{b}=\mathbf{0}$. Ablation FULL equals the oracle column of Table 12; the three controls of Table 6 use the same configuration. Given that configuration and the frozen checkpoints, every reported cell is deterministic at seed 0 except MUTAG's, the one dataset whose path is order-nondeterministic under CUDA scatter-based node pooling; it is reported as the mean of 5 runs (Appendix E).

AI USE STATEMENT

We used generative AI tools, in the form of large language model assistants, for the following tasks carrying a disclosure requirement: implementing the method and the experimental pipeline; designing and giving feedback on the experimental methodology, including the 1-shot evaluation protocol, the ablation variants and the hyperparameter grid; formulating mathematical claims; cleaning and reformatting the datasets; and interpreting results. The research problem, the core idea of constructing the prompt in closed form rather than fitting it, and the conceptual framework are the authors'. We did not use generative AI tools to propose or refine the hypotheses, to generate synthetic data, or to assist with mathematical proofs.

We additionally used generative AI tools for tasks with recommended disclosure: drafting and editing the manuscript to improve readability, writing and editing software code including the plotting scripts, and identifying, summarising and formatting references.

We have reviewed all AI-assisted work. Code and every reported number were checked against the implementations and the raw experimental logs; notation and equations were checked against our implementation; and all text was read and approved by the authors. We take responsibility for the final content of this work, including all text, claims and artifacts.

REFERENCES

Qin Chen, Liang Wang, Bo Zheng, and Guojie Song. DAGPrompT: Pushing the limits of graph prompting with a distribution-aware graph prompt tuning approach. In *WWW*, pp. 4346–4358, 2025.

Runjin Chen, Tong Zhao, Ajay Kumar Jaiswal, Neil Shah, and Zhangyang Wang. LLaGA: Large language and graph assistant. In *ICML*, volume 235, pp. 7809–7823, 2024.

Kaize Ding, Zhe Xu, Hanghang Tong, and Huan Liu. Data augmentation for deep graph learning: A survey. *ACM SIGKDD Explorations Newsletter*, 24(2):61–77, 2022.

Taoran Fang, Yunchao Zhang, Yang Yang, Chunping Wang, and Lei Chen. Universal prompt tuning for graph neural networks. In *NeurIPS*, volume 36, pp. 52464–52489, 2023.

Xingbo Fu, Yinhan He, and Jundong Li. Edge prompt tuning for graph neural networks. In *ICLR*, 2025a.

Xingbo Fu, Zehong Wang, Zihan Chen, Jiazheng Li, Yaochen Zhu, Zhenyu Lei, Cong Shen, Yanfang Ye, Chuxu Zhang, and Jundong Li. Graph prompting for graph learning models: Recent advances and future directions. In *KDD*, 2025b. doi: 10.1145/3711896.3736566.

Justin Gilmer, Samuel S Schoenholz, Patrick F Riley, Oriol Vinyals, and George E Dahl. Neural message passing for quantum chemistry. In *ICML*, 2017.

Jean-Bastien Grill, Florian Strub, Florent Altché, et al. Bootstrap your own latent: A new approach to self-supervised learning. *NeurIPS*, 2020.

Will Hamilton, Rex Ying, and Jure Leskovec. Inductive representation learning on large graphs. In *NeurIPS*, 2017.

Kaveh Hassani and Amir Hosein Khasahmadi. Contrastive multi-view representation learning on graphs. In *ICML*, 2020.

Zhenyu Hou, Xiao Liu, Yukuo Cen, Yuxiao Dong, Hongxia Yang, Chunjie Wang, and Jie Tang. GraphMAE: Self-supervised masked graph autoencoders. In *KDD*, pp. 594–604, 2022.

Yongqi Huang, Jitao Zhao, Dongxiao He, Xiaobao Wang, Yawen Li, Yuxiao Huang, Di Jin, and Zhiyong Feng. One prompt fits all: Universal graph adaptation for pretrained models. In *NeurIPS*, volume 38, pp. 85497–85530, 2025.

Thomas N Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. In *ICLR*, 2017.

Junhyun Lee, Wooseong Yang, and Jaewoo Kang. Subgraph-level universal prompt tuning. *arXiv preprint arXiv:2402.10380*, 2024.

Xiang Lisa Li and Percy Liang. Prefix-tuning: Optimizing continuous prompts for generation. *ACL*, 2021.

Zhen Li, Yuwei Gong, Pengfei Liu, et al. GraphMAE2: A decoding-enhanced masked graph autoencoder. In *Preprint arXiv:2304.01015*, 2023.

Zemin Liu, Xingtong Yu, Yuan Fang, and Xinming Zhang. GraphPrompt: Unifying pre-training and downstream tasks for graph neural networks. In *WWW*, pp. 417–428, 2023.

Jiezhong Qiu, Qibin Chen, Yuxiao Dong, et al. GCC: Graph contrastive coding for graph neural network pre-training. In *KDD*, 2020.

Yangyi Shen, Jincheng Zhou, Beatrice Bevilacqua, et al. Zero-shot generalization of GNNs over distinct attribute domains. In *ICML*, pp. 54701–54731, 2025.

Mingchen Sun, Kaixiong Zhou, Xin He, Ying Wang, and Xin Wang. GPPT: Graph pre-training and prompt tuning to generalize graph neural networks. In *KDD*, pp. 1717–1727, 2022.

Xiangguo Sun, Hong Cheng, Jia Li, Bo Liu, and Jihong Guan. All in one: Multi-task prompting for graph neural networks. In *KDD*, pp. 2120–2131, 2023.

Shantanu Thakoor, Corentin Tallec, Mehdi Azabou, Mehdi Gheshlaghi Azar, Remi Munos, Petar Veličković, and Michal Valko. Large-scale representation learning on graphs via bootstrapping. In *ICLR*, 2022.

Petar Velickovic, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, and Yoshua Bengio. Graph attention networks. In *ICLR*, 2018.

Petar Velickovic, William Fedus, William L Hamilton, Pietro Liò, Yoshua Bengio, and R Devon Moine. Deep graph infomax. In *ICLR*, 2019.

Duo Wang, Yuan Zuo, Fengzhi Li, and Junjie Wu. LLMs as zero-shot graph learners: Alignment of GNN representations with LLM token embeddings. In *NeurIPS*, volume 37, pp. 5950–5973, 2024.

Jason Wei, Xuezhi Wang, Dale Schuurmans, et al. Chain-of-thought prompting elicits reasoning in large language models. In *NeurIPS*, 2022.

Rex Ying, Ruining He, Kaifeng Chen, Pong Eksombatchai, William L Hamilton, and Jure Leskovec. Graph convolutional neural networks for web-scale recommender systems. In *KDD*, 2018.

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

Yuning You, Tianlong Chen, Yang Wang, and Wei-Ming Shen. Graph contrastive learning automated. In *ICML*, 2021.

Xingtong Yu, Jie Zhang, Yuan Fang, and Renhe Jiang. Non-homophilic graph pre-training and prompt learning. In *KDD*, pp. 1844–1854, 2025a.

Xingtong Yu, Chang Zhou, Zhongwei Kuai, Xinming Zhang, and Yuan Fang. GCoT: Chain-of-thought prompt learning for graphs. In *KDD*, 2025b. doi: 10.1145/3711896.3736974.

Ziyu Zheng, Yaming Yang, Ziyu Guan, Wei Zhao, Xinyan Huang, and Weigang Lu. Beyond single-granularity prompts: A multi-scale chain-of-thought prompt learning for graph. In *WWW*, pp. 547–558, 2026. doi: 10.1145/3774904.3792115.

Yanqiao Zhu, Yichen Xu, Feng Yu, Qiang Liu, Shu Wu, and Liang Wang. Deep graph contrastive representation learning. In *ICML Workshop on Graph Representation Learning and Beyond*, 2020.

Yanqiao Zhu, Yichen Xu, Feng Yu, Qiang Liu, Shu Wu, and Liang Wang. Graph contrastive learning with adaptive augmentation. In *WWW*, 2021.

A SETUP DETAILS

Data and encoder. We use eight 1-shot benchmarks—node (Cora, Citeseer, Photo, Pubmed) and graph (MUTAG, COX2, BZR, PROTEINS)—with the official MSGCOT splits and frozen official GCN checkpoints pretrained by link prediction (a contrastive objective over sampled edges and non-edges): 2-layer on nodes except Pubmed's 3-layer checkpoint, and 3-layer on graphs. The Cora worked example of Table 3 uses $N{=}2708$ nodes, $F{=}1433$ bag-of-words features, $C{=}7$ classes, and a frozen 2-layer GCN of width $d{=}256$.

Evaluation. Official 1-shot$\times$100 (tasks 0–99): one labeled example per class; report mean$\pm$std over tasks. CFGP builds $\boldsymbol{M}$ on all instances; MSGCOT likewise embeds the full $(\boldsymbol{X}, \boldsymbol{A})$ then slices into labeled/test. Test labels are unused at test time. MSGCOT trains its prompt modules from scratch on every task; CFGP trains none.

k-shot transfer (Figure 2). Cora and MUTAG, $k{=}1$–7, official tasks 0–99, raw Acc. All three methods load the frozen MSGCOT checkpoint (2-layer Cora, 3-layer MUTAG). CFGP reuses the configuration of Table 7 with $\mathbf{b}{=}\mathbf{0}$ and does not retune. MSGCOT uses Cora $(c_l, c_r, r) = (1, 0.1, 8)$ and MUTAG $(2, 0.05, 8)$, where c_l is its number of coarsening layers, c_r its coarsening rate, and r its low-rank dimension. GCoT trains its CondNet under the official epoch/patience budget but on the MSGCOT encoder, not GCoT's own 3-layer Cora weights. Cora every k and MUTAG $k{\leq}5$ are official MSGCOT few-shot dumps; MUTAG $6/7$-shot are generated once (seed 0) by the MUTAG rule: k labeled graphs per class, test = the rest. The $k{=}1$ cells are this matched-encoder run, not Table 4.

Baselines. Frozen-prototype ($\lambda{=}0$ inside CFGP; A2 in Table 6); single-step prompt baselines; GCoT; MSGCOT. These baselines and GCoT are our re-run on the official MSGCOT splits, with their raw and balanced Acc in Tables 4, 9 and 10; MSGCOT is our re-run under the published Table 7 configuration (node $r{=}8$, graph $r{=}1$); Appendix E records the re-run and its reproducibility. CFGP computed by us. Graph-side GCoT per-class recall is Table 10.

Reported configuration. The configuration $(\mathrm{fanout}, w, \lambda, L)$ is selected per dataset over a declared grid on the downstream 1-shot accuracy, with $\mathbf{b} = \mathbf{0}$ (Table 7). Selection is per dataset, not per task: the same configuration serves all 100 tasks. What is *not* tuned is the architecture: the graph-side choices listed in Table 7 are one identical block for all four graph datasets, and the node-side choices are one identical block for all four node datasets. The grid is fixed in advance and not expanded per dataset: $\mathrm{fanout} \in \{5, 8, 10\}$, $w \in \{0.5, 1.0, 2.0\}$, $\lambda \in \{0.3, 0.5, 1.0, 1.5, 2.0, 3.0\}$, $L \in \{1, 2, 3\}$ (162 combinations per dataset). We state the asymmetry plainly: the 1-shot splits provide no validation set, so *any* per-dataset selection here—ours or a baseline's—is made on the same evaluation tasks. The baselines likewise report per-dataset values (MSGCOT's Table 7 fixes a coarsening rate per dataset out of a declared set of four, and its Appendix reports sensitivity curves against the same reported metric); our grid is larger, spanning four axes rather than one, and we do not claim otherwise. Two reference points bound this choice: a single shared configuration with no per-dataset selection (balanced 58.43, seven of eight against MSGCOT at $r{=}8$) is the no-tuning lower bound of the same method, and Table 12 unlocks those choices as an upper bound (raw 61.44). Three scalars of the construction are fixed for every dataset and are not part of this grid: the clustering temperature τ of Eq. (4) is 0.1, the readout temperature τ_u of Eq. (3) is 0.5, and the signature exponent γ of Eq. (6) is 1.

Table 7: Reported configuration: a per-dataset $(\mathrm{fanout}, w, \lambda, L)$ selected over a declared grid on the downstream 1-shot accuracy, $\mathbf{b} = \mathbf{0}$. Datasets are grouped into node vs. graph. The last column lists the design choices held fixed, written in the order *inject / pathway / residual*; within each group they are identical for all datasets, and only the configuration varies. Node rows: the prompt is added to the node features and re-encoded by the frozen GCN each round. Graph rows: the prompt is mixed into the node features, and the node embeddings are pooled by mean and standard deviation, and the result is re-encoded by the same GCN.

Side	Dataset	fanout	w	λ	L	inject / pathway / residual
Node	Cora	8	2.00	1.0	2	add, GCN, mix

Continued on next page

Table 7: Continued

Side	Dataset	fanout	w	λ	L	inject / pathway / residual
Node	Citeseer	5	1.00	0.5	2	add, GCN, mix
Node	Photo	8	0.50	2.0	2	add, GCN, mix
Node	Pubmed	8	2.00	0.5	1	add, GCN, mix
Graph	MUTAG	8	0.50	0.3	3	prompt, meanstd, mix
Graph	COX2	10	2.00	0.3	2	prompt, meanstd, mix
Graph	BZR	10	2.00	3.0	3	prompt, meanstd, mix
Graph	PROTEINS	5	0.50	0.5	1	prompt, meanstd, mix

B NON-PROMPT REFERENCE POINTS

Table 4 compares CFGP against methods that fit a prompt on every task. For reference we also ran four baselines that fit none, on the same 1-shot$\times$100 splits: GCN and GAT trained from scratch on the 1-shot support, and LP and DGI/Infograph pre-trained and then fine-tuned end-to-end on that support, all with a linear head. They follow a protocol different from the frozen-encoder prompt methods and carry no claim; Table 8 reports them. On COX2, where the fixed test-class ratio leaves raw Acc uninformative, a logistic regression fit on the frozen encoder reaches 50.07 balanced Acc against CFGP's 50.94; that dataset supports no method claim.

Table 8: Non-prompt reference points on the same 1-shot$\times$100 splits as Table 4. Encoder trained from scratch (GCN, GAT) or pre-trained and then fine-tuned end-to-end on the 1-shot support (LP: link prediction; DGI/Infograph: local–global mutual information), all with a linear head. Mean: macro-average over the eight datasets, each dataset weighted equally.

	Node				Graph				
Method	Cora	Citeseer	Photo	Pubmed	MUTAG	COX2	BZR	PROTEINS	Mean
Raw Acc (%)									
GCN	32.70±13.53	26.97±9.19	57.97±11.31	50.39±9.12	56.65±14.79	56.24±16.20	47.43±14.15	56.75±9.40	48.14
GAT	31.12±14.65	28.90±8.22	51.15±12.24	50.84±9.54	55.61±14.71	55.58±15.00	47.95±14.98	55.74±10.46	47.11
LP	56.35±12.67	42.47±7.57	67.09±9.32	57.54±9.06	62.19±14.49	51.65±6.02	52.14±9.37	54.12±5.57	55.44
DGI/Infograph	51.45±14.58	38.97±8.80	58.79±10.38	56.45±8.62	58.09±15.31	54.72±14.00	49.35±14.95	56.01±10.62	52.98
Balanced Acc (%)									
GCN	37.63	27.50	65.19	51.42	57.58	51.68	52.20	54.42	49.70
GAT	35.38	28.74	57.41	51.92	56.07	51.51	51.80	53.58	48.30
LP	54.07	40.35	74.81	60.16	57.83	50.66	54.56	54.45	55.86
DGI/Infograph	52.20	38.43	66.13	60.04	56.29	51.27	52.60	53.91	53.86

C FULL RESULT TABLES

Table 9 reports balanced Acc for every method of Table 4, not only for the methods discussed in the main text.

Table 10 reports per-class recall for every method of Table 4, including the other baselines and the majority predictor.

D DESIGN REMARKS AND COMPLEXITY

Why not a signature in $\boldsymbol{H}_0$. Building the signature in $\boldsymbol{H}_0$ would put $\boldsymbol{P}$ in the embedding and add it after message passing, so Eq. (9) would not run. The input write is what lets the frozen encoder carry the task signal. On the node branch of Eq. (8), a local linearization about $\boldsymbol{X}$ gives

$$\boldsymbol{H}' \;\approx\; \boldsymbol{H}_0 + J\big(\lambda \,\|\boldsymbol{u}_i\|_2 \,\mathrm{RowNorm}(\widehat{\boldsymbol{P}})_i\big), \tag{11}$$

where J is the local map from input features to representations induced by the frozen GCN. This is an intuition for why input-space injection can move the frozen representation; it is not a guarantee.

Role alignment with MSGCOT. Relative to MSGCOT, spherical k-means replaces the learned assignment, and one product $\boldsymbol{P} = \boldsymbol{S}\tilde{\boldsymbol{M}}$ replaces its coarsening tokens and its per-instance prompt write. The row-wise CondNet is dropped. CFGP trains zero prompt parameters. This is a role alignment, not a claim of recovery.

Table 9: Balanced Acc (%) under the same 1-shot×100 splits as Table 4: the full table. Single-step prompt baselines (†) and GCoT: our re-run. MSGCOT: our reproduction at $r{=}1$ and $r{=}8$ (Table 7 coarsening; node default is $r{=}8$). CFGP: same predictions as Table 4. Mean: macro-average over the eight datasets, each dataset weighted equally. **Bold**: best in the column. Underline: second best.

	Node				Graph				
Method	Cora	Citeseer	Photo	Pubmed	MUTAG	COX2	BZR	PROTEINS	Mean
All-in-One†	45.18	38.01	60.40	52.57	56.09	50.79	53.50	53.82	51.30
GPF†	54.48	41.61	72.48	59.46	57.73	**51.10**	53.09	53.80	55.47
GPF+†	56.28	42.13	73.64	59.92	57.68	51.01	53.35	53.89	55.99
GraphPrompt†	56.76	42.78	74.95	65.91	57.94	50.89	53.24	53.73	57.03
DAGPrompT†	56.08	42.61	74.77	65.44	57.51	51.02	53.23	54.22	56.86
EdgePrompt†	54.35	41.62	71.69	51.10	56.25	50.71	53.60	53.86	54.15
EdgePrompt+†	54.54	41.61	72.25	60.12	56.08	50.91	53.85	54.23	55.45
SUPT†	54.42	41.37	72.64	60.31	57.77	50.68	52.89	54.09	55.52
GCoT	59.32	43.92	72.79	66.40	54.90	50.79	53.13	53.15	56.80
MSGCOT ($r{=}1$)	47.75	40.78	71.60	66.00	53.35	50.27	50.11	54.70	54.32
MSGCOT ($r{=}8$)	57.64	46.02	73.23	**66.82**	54.72	50.19	49.13	51.47	56.15
majority	14.29	16.67	12.50	33.33	50.00	50.00	50.00	50.00	34.60
CFGP	**64.41**	**48.52**	**78.44**	65.98	**60.66**	50.94	**55.82**	**56.25**	**60.13**

Table 10: Graph-side per-class recall: the full table. Each cell: balanced Acc (per-class recalls); balanced = macro recall, and the majority constant scores 50. MSGCOT at $r{=}1$ and $r{=}8$; GCoT and SUPT are our re-runs on the same splits.

	Graph			
Method	MUTAG	COX2	BZR	PROTEINS
CFGP	**60.66** (49.2/72.1)	**50.94** (55.0/46.9)	**55.82** (61.3/50.4)	**56.25** (61.5/51.0)
MSGCOT $r{=}1$	53.35 (41.2/65.5)	50.27 (90.4/10.1)	50.11 (92.6/7.6)	54.70 (58.8/50.6)
MSGCOT $r{=}8$	54.72 (38.9/70.5)	50.19 (87.9/12.5)	49.13 (84.0/14.2)	51.47 (59.7/43.3)
GCoT	54.90 (38.6/71.2)	50.79 (53.1/48.5)	53.13 (55.4/50.8)	53.15 (66.8/39.5)
SUPT	57.77 (42.0/73.6)	50.68 (51.7/49.7)	52.89 (48.9/56.9)	54.09 (57.5/50.7)
majority	50.00 (0.0/100.0)	50.00 (100.0/0.0)	50.00 (100.0/0.0)	50.00 (100.0/0.0)

Asymptotics. Write N for the number of instances (nodes, or graphs on the graph side), E for edges, d for the encoder width, C for the number of classes, L for CFGP rounds, $K_\ell = \text{fanout}^\ell$ for the channel count at round ℓ, and T for the number of 1-shot tasks (T=100 in the official suite). Let Cost_{GNN} denote one frozen-encoder pass—the same unit of work that both methods pay on every pass. Per task, CFGP's dominant work is L frozen re-encodes, plus cosine prototype $O(NCd)$, spherical k-means $O(I \cdot N \cdot K_\ell \cdot d)$ over I iterations, and the signature / prompt product $O(N \cdot CK_\ell \cdot F)$. With $L \in \{1, 2, 3\}$ and small K_ℓ, the suite cost is

$$O\big(T \cdot L \cdot \text{Cost}_{\text{GNN}} + T \cdot \text{poly}(N, C, K, d)\big)$$

in the forward direction only. Trainable prompts are fit for up to E_{ep} epochs per task (MSGCOT: 200 on nodes / 400 on graphs; GCoT: 50 on nodes / 400 on graphs, with early stopping), each with a forward and a backward through the encoder and the prompt:

$$O\big(E_{\text{ep}} \cdot (\text{Cost}_{\text{GNN}} + \text{Cost}_{\text{prompt}})\big)$$

per task, times T. Under a shared Cost_{GNN} order, the leading ratio is E_{ep}/L (tens to hundreds versus 1–3), amplified by the missing prompt backward pass. Table 3 instantiates the contrast on Cora; Table 2 measures wall-clock.

Notation. N instances, E edges, d encoder width, C classes, L CFGP rounds, $K_\ell = \text{fanout}^\ell$, T=100 tasks, E_{ep} the per-task epoch cap above.

E RUNTIME MEASUREMENT

Table 2 is timed on a single NVIDIA GeForce RTX 4090 (driver 550.107.02, CUDA 12.4, 24 GB, 450 W cap), `CUDA_VISIBLE_DEVICES=0`, GPU idle at start. Each (method, dataset) is a separate process. The evaluation is official 1-shot×100 (tasks 0–99). Graph-side labeled/test splits follow MSGCOT. GCoT's original graph loader used a different path and omitted MUTAG and PROTEINS; only that loader path was pointed at the MSGCOT splits, without changing GCoT's model or weights.

MSGCOT re-run and reproducibility. MSGCOT is reproduced from the released code under its published Table 7 configuration (node r=8, graph r=1). A second full re-run under the identical configuration reproduces the node-side values bit-for-bit for Cora, Citeseer, and Pubmed and to within 0.02 points for Photo, whereas the eight graph-side values move by up to 0.68 points. Graph-side margins below that resolution are therefore not resolved by a single run.

Two timing blocks share the table. CFGP / MSGCOT / GCoT use each method's official script (CFGP and MSGCOT load the MSGCOT frozen encoder; GCoT loads its own checkpoints). Rows marked † (All-in-One, GPF, GPF+, SUPT, GraphPrompt, DAGPrompT, EdgePrompt, EdgePrompt+) are a later serial run on the same GPU and splits: same MSGCOT LP-GCN (hid 256), cosine prototype head, node 200 / graph 400 epochs with no early stop; all of them except EdgePrompt and EdgePrompt+ apply their prompt to the cached 256-d encoder output, whereas the EdgePrompt pair injects inside the encoder layers. Training budgets are therefore not aligned across rows, because CFGP has no per-task loop:

- CFGP: no per-task gradient training.
- Single-step prompt baselines (†): 200 epochs per node task and 400 per graph task, run to completion.
- MSGCOT: at most 200 epochs per node task and 400 per graph task, with early stopping.
- GCoT: at most 50 epochs (patience 20) on nodes and 400 (patience 50) on graphs. GCoT's published node budget is already smaller than MSGCOT's.

The leading asymptotic contrast is an E_{ep}-epoch train loop versus L forward rounds; Table 2 measures the resulting wall-clock, not a matched-epoch FLOPs study. Table 3 counts trainable prompt parameters on Cora from the released downstream modules (every `requires_grad` tensor; the frozen GCN is excluded); its GraphPrompt / GPF+ Cora walls are the same baseline cells as in

Table 2. MSGCOT's published Table 4 quotes bottleneck-only counts of 10.37K / 32.76K for MSGCOT / GCoT; the released CondNets that write into the F-dimensional input are larger (62.3K / 217.8K), which is what Table 3 counts. In Table 2, the fastest method other than CFGP is All-in-One on Cora, Citeseer, Photo and the total, GCoT on Pubmed, MSGCOT on COX2 and BZR, and GraphPrompt on MUTAG and PROTEINS. Early-stopping state in the MSGCOT and GCoT node scripts is retained across tasks, so later tasks may stop earlier. Reported MSGCOT / GCoT times are the scripts as they ran, not a full-budget upper bound. GCoT's original graph script prints full-graph features and per-task predictions; those prints were disabled so the graph-side wall-clock is closer to algorithm time than to log I/O. All reported times include interpreter start, data load, and checkpoint load, so CFGP and trainable methods are compared on the same process-level wall-clock.

The fair reading is: same GPU, same splits—38.4 s for CFGP, 10.9–39.1 min for single-step prompt baselines under a fixed budget, 32.8 min for MSGCOT and 47.3 min for GCoT under official scripts. The comparison is not equal compute and not equal backbone depth (GCoT's node encoder is 3-layer; MSGCOT's Cora / Citeseer / Photo encoders are 2-layer). Each method was timed once.

F CLASS-PRIOR DIAGNOSTIC

Adding a class-logit prior $\mathbf{b}$ raises raw Acc by collapsing minority recall. Reported comparisons therefore use $\mathbf{b} = \mathbf{0}$. Table 11 changes only $\mathbf{b}$, under the oracle configuration.

Table 11: Prior diagnostic on *graph* datasets under the oracle configuration: only $\mathbf{b}$ changes. $\mathbf{b}$ buys raw Acc by collapsing minority recall. The *MSGCOT* row is our re-run at r=1 (Table 4), shown for comparison. **Bold**: best balanced Acc in the block. All reported numbers use $\mathbf{b} = \mathbf{0}$.

Dataset	$\mathbf{b}$	raw	balanced	per-class recall	
MUTAG	—	64.39	**60.88**	50.3	71.4
	$[0, 1.5]$	68.20	56.25	20.4	92.1
COX2	—	54.03	**51.79**	55.7	47.8
	$[0.2, 0]$	74.68	50.70	93.1	8.3
	MSGCOT	72.99	50.27	90.4	10.1
BZR	—	60.89	**56.86**	63.8	49.9
	$[0.6, 0]$	71.28	54.73	83.4	26.1
PROTEINS	—	57.23	**54.98**	66.7	43.3
	$[0.1, 0]$	60.70	54.87	85.3	24.5

G ORACLE CEILING

Table 12 unlocks every axis, including the choices held fixed in Table 7, on test Acc over the same 100 tasks. It is an oracle ceiling, not a configuration we recommend: it costs three extra tuned axes to buy $+0.29$ mean raw Acc over the reported configuration (61.44 vs. 61.15), and it does not change the win count (5 of 8). The gap to MSGCOT on raw Acc is therefore not an artifact of leaving those choices fixed. Ablations below use this oracle configuration only as an oracle-ceiling diagnostic.

Table 12: Oracle ceiling: the oracle configuration, with *all* axes unlocked, by test Acc (tasks 0–99), $\mathbf{b}$=$\mathbf{0}$. Rows: methods; columns: datasets (node / graph). Compare the reported configuration of Table 4: 61.15 (5 of 8).

	Node				Graph				
Method	Cora	Citeseer	Photo	Pubmed	MUTAG	COX2	BZR	PROTEINS	Mean
CFGP (oracle configuration)	68.01±17.46	52.61±11.01	70.96±11.46	63.43±10.12	64.39±15.28	54.03±13.53	60.89±14.21	57.23±11.23	61.44
MSGCOT	60.58	49.21	66.73	64.15	57.41	72.99	74.68	55.48	62.65
Δ vs. MSGCOT	+7.43	+3.40	+4.23	−0.72	+6.98	−18.96	−13.79	+1.75	−1.21 (5/8)