LEARNING WHERE TO READ: GRAPH CLASSIFICATION UNDER OBSERVATION BUDGETS

Anonymous authors
Paper under double-blind review

ABSTRACT

Graph-level prediction typically assumes access to the complete input graph. However, when node information must be acquired through external queries or costly measurements, a predictor must decide what to observe within a limited budget. We study budget-limited active graph reading, where an agent sequentially reveals nodes and predicts the graph label from acquired evidence. Learning such a policy is challenging because graph labels provide no direct supervision for exploration actions, and the combinatorial space of reading trajectories makes unguided exploration difficult. We introduce BAGR, a budget-aware graph reader that combines sequential node selection with recurrent evidence accumulation. To guide exploration, we generate privileged teacher trajectories and filter them according to a pretrained classifier's predictive performance on the revealed subgraphs. Imitation learning transfers these demonstrations into an observation-constrained policy, providing a task-informed initialization. On-policy reinforcement learning then refines the policy using classification-based rewards on its own trajectories, enabling task-driven exploration beyond the teacher demonstrations. We evaluate BAGR on CIFAR10 superpixels, COLLAB, and OGBG-PPA under explicit observation and node-budget constraints, and experimental results show gains over the evaluated fixed heuristics.

1 INTRODUCTION

Graphs represent entities and their relationships, making graph-level prediction valuable for diverse applications, such as identifying the research field of a collaboration network (Yanardag & Vishwanathan, 2015), recognizing an image from its superpixel graph (Dwivedi et al., 2023), and determining the taxonomic group of a protein-association network (Hu et al., 2020b). Standard graph classifiers and pooling models assume the complete graph is available prior to inference (Xu et al., 2019; Ying et al., 2018). However, when local graph information must be computed on demand, exhaustive observation becomes prohibitively expensive. For example, in a high-resolution image represented as a graph of spatial regions, obtaining a node's visual features requires running an image encoder on the corresponding crop. Sequentially selecting image regions for processing can mitigate these costs (Mnih et al., 2014; Wang et al., 2020). This necessitates predicting graph properties by actively acquiring a limited budget of evidence (e.g., node attributes and local connectivity), forcing a predictor to decide both *what* to observe next and *how* to integrate the acquired information.

Existing approaches broadly fall into two categories, neither of which fully addresses this localized acquisition setting. First, graph pooling and masking methods decide which parts of a graph are useful. For instance, SAGPool (Lee et al., 2019) scores and retains top nodes, while Soft-Mask (Yang et al., 2021) assigns continuous relevance weights. However, both rely on node features and connections to make these decisions, requiring the entire graph to be processed beforehand. Consequently, they answer *which nodes to keep after reading* rather than *which nodes to read in the first place*. Conversely, learned graph exploration trains agents to sequentially navigate from node to node, accumulating evidence for prediction (Lee et al., 2018; Martinkus et al., 2023). Yet, these agents typically demand access to hidden information (e.g., features or labels of neighboring nodes) before deciding which one to visit. As computing this information is the cost we aim to avoid, an ideal method must learn to route its exploration without peeking at the hidden attributes of nodes.

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.

We formalize this problem as *budget-limited active graph reading*. Starting from a seed node, an agent leverages strictly the information revealed so far to determine its next acquisition. It operates under a strict budget constraint and must predict the graph label solely from the evidence it collects. Learning an effective reading strategy presents two major challenges. First, graph labels provide the correct terminal prediction but not the sequence of reading choices required to reach it. Second, the action space of possible reading sequences grows exponentially, and an isolated observation may only become informative when combined with later evidence. Together, these challenges create a severe credit assignment problem, making it hard to learn a policy through unguided trial and error.

To address these challenges, we introduce *BAGR*, a budget-aware graph reader that jointly learns to acquire information and synthesize it for prediction.[1] To handle the interdependence of sequential observations, BAGR maintains a recurrent summary of the reading history, using it to attend to previously read nodes so that new observations can seamlessly interact with earlier evidence. To overcome the lack of action-level supervision, we initialize the policy via imitation of teacher-generated sequences, followed by on-policy reinforcement learning with staged classification feedback. Because an incorrect prediction may stem from either uninformative observations or an under-trained predictor, relying solely on the reader's early mistakes can derail training. Therefore, our RL procedure initially combines the correct-label probabilities from a frozen pretrained classifier and the reader's active predictor. The frozen classifier acts as a stable reference for evidence quality, while the reader's feedback ensures the model gathers evidence it can independently interpret at test time. Later, rewards depend exclusively on the reader's own predictions. Crucially, this progression prevents the agent from either collapsing under an immature predictor early in training or remaining permanently constrained by a surrogate model at convergence. Ablations confirm that this reinforcement learning procedure achieves comparable results even without the imitation-based initialization.

We evaluate BAGR on CIFAR10 superpixels, COLLAB, and OGBG-PPA, covering image-derived graphs, social collaboration networks, and protein-association networks, respectively. Under matched node budgets, BAGR substantially outperforms baseline methods across most settings. These results demonstrate that task-aware active reading can successfully acquire diagnostic evidence for graph classification while observing only a small fraction of the total nodes.

2 Related Work

Graph pooling and task-relevant subgraph selection. Graph pooling and masking focus on identifying critical subgraphs for property prediction (Grattarola et al., 2024). DiffPool learns clusters of nodes (Ying et al., 2018), Graph U-Nets selects nodes for pooling (Gao & Ji, 2019), and ASAP pools local subgraphs using structure-aware attention (Ranjan et al., 2020); Graph Information Bottleneck seeks informative, compact subgraphs (Yu et al., 2021). SAGPool (Lee et al., 2019) leverages graph convolutions to score and retain top-ranking nodes, whereas Soft-Mask (Yang et al., 2021) learns continuous importance weights to attenuate less informative regions. They use graph information available before selection. Acquiring node features beforehand defeats the purpose of budgeted observation. In contrast, BAGR operates under a strict active acquisition paradigm, selecting candidates based solely on visible context prior to revealing their attributes. For a fair comparison, our adapted SAGPool conceals unread node features and acquires nodes sequentially (Appendix A.2).

Learned graph exploration and sequential classification. Rather than processing full graphs, exploration-based methods learn sequential traversal policies to collect informative evidence for prediction. GAM (Lee et al., 2018) trains a recurrent walker via reinforcement learning, AgentNet (Martinkus et al., 2023) employs cooperative neural agents to navigate the graph, and RWE-SGNN (Huang & Kasai, 2026) learns reinforcement-driven random walks to extract discriminative substructures. However, these methods implicitly rely on pre-existing candidate information—such as neighboring node types in GAM, candidate representations in AgentNet, or global structural priors in RWE-SGNN—before executing a step. Inspecting unobserved candidate attributes violates the active reading constraint, incurring observation costs on nodes that may ultimately be bypassed. BAGR instead starts from a single seed node and uses only currently observed features and visible attachment topology for exploration, ensuring candidate node attributes remain strictly unobserved.

[1] Anonymous code: `https://anonymous.4open.science/r/BAGR-5186`.

Active feature acquisition and budgeted prediction. Active feature acquisition studies sequentially querying missing attributes under explicit observation costs (Aronsson et al., 2025). Budgeted prediction from partially observed attributes has been studied theoretically (Cesa-Bianchi et al., 2011). EDDI (Ma et al., 2019) selects observations using expected information gain, while Shim et al. (2017) and Janisch et al. (2019) learn acquisition policies for classification. These approaches primarily address feature vectors. More closely related, Janisch et al. (2024) extend costly-feature acquisition to nested, variable-size data with dynamically available actions and a separate classifier. BAGR instead reads an arbitrary graph: each acquired node reveals neighboring candidates, and prediction uses the resulting revealed subgraph rather than a fixed vector or tree-structured record. Graph active learning is also budget-driven, but its budget buys node labels for training rather than observations for prediction (Hu et al., 2020a; Zhang et al., 2022).

Graph explanation and interpretable subgraphs. Graph explanation methods identify influential nodes and edges for a trained model's predictions (Yuan et al., 2023). GNNExplainer (Ying et al., 2019) optimizes masks for individual predictions, PGExplainer (Luo et al., 2020) learns reusable edge-mask explanations, and SubgraphX (Yuan et al., 2021) uses Monte Carlo tree search with Shapley values to identify key substructures. However, explainers operate post-hoc with full visibility over the input graph to explain an existing prediction, rather than performing active prediction from unobserved graphs under a budget constraint. Though BAGR's acquired subgraphs offer transparency on what evidence was collected, its goal is active classification rather than model attribution.

3 Problem Setup

When obtaining local graph information is costly, a predictor must decide what to read without first inspecting every node. Thus, we study *budget-limited active graph reading*, where an agent acquires graph information one node at a time and predicts the graph label from collected evidence. Consider a graph G with node features X and a graph label y. We define the agent's access to the graph below.

Budget and seed. Given a graph G, a seed node s_0, and an integer budget B, the agent starts by reading s_0. Each extra node costs one unit, so a completed B-step sequence contains $B + 1$ nodes.

Revelation process and visibility. Let S_t denote the nodes already read after t steps, starting with $S_0 = \{s_0\}$. Reading a node reveals its features and unread neighbors. These unread neighbors of S_t form the *frontier*, denoted by F_t. At each step, the agent selects one frontier node $a_t \in F_t$ and reads it, so that $S_{t+1} = S_t \cup \{a_t\}$. This may expose new candidates for subsequent steps. Knowing that a candidate exists is different from reading its features. The agent can access the features of read nodes without cost. However, the features of unread frontier nodes remain hidden and connections between two unread nodes are not visible. We denote the currently visible information by $\boldsymbol{o}_t$.

Problem definition. Given labeled training graphs, our goal is to learn both a reading policy π and a predictor f. The policy decides which frontier node to read next, while the predictor uses the collected evidence to infer the graph label. Let $\boldsymbol{h}_t$ denote the reading history up to step t, including the observations and earlier choices. We write these two operations as

$$\boldsymbol{a}_t \sim \pi(\cdot \mid \boldsymbol{h}_t), \qquad \hat{\boldsymbol{y}} = f(\boldsymbol{h}_B).$$

Here, $\hat{\boldsymbol{y}}$ is the predicted label after the budget is exhausted. Our objective is to learn the reading policy π and predictor f jointly to maximize prediction accuracy on unseen graphs under the given budget. Each reading decision changes the evidence for classification and future reading.

4 Method

Budget-limited graph reading requires learning which observations to acquire and how to use them for prediction. This presents two major challenges. First, graph labels provide the correct prediction but not the specific reading choices required to reach it. While classification outcomes can guide these choices, an incorrect prediction creates a credit assignment problem. It may stem from acquiring uninformative observations, or from a predictor that has not yet learned to utilize informative ones. Second, the action space of reading sequences grows rapidly, and a single observation may only become informative when combined with future evidence. Therefore, the reader must learn from sparse task feedback while effectively connecting information across sequential steps.

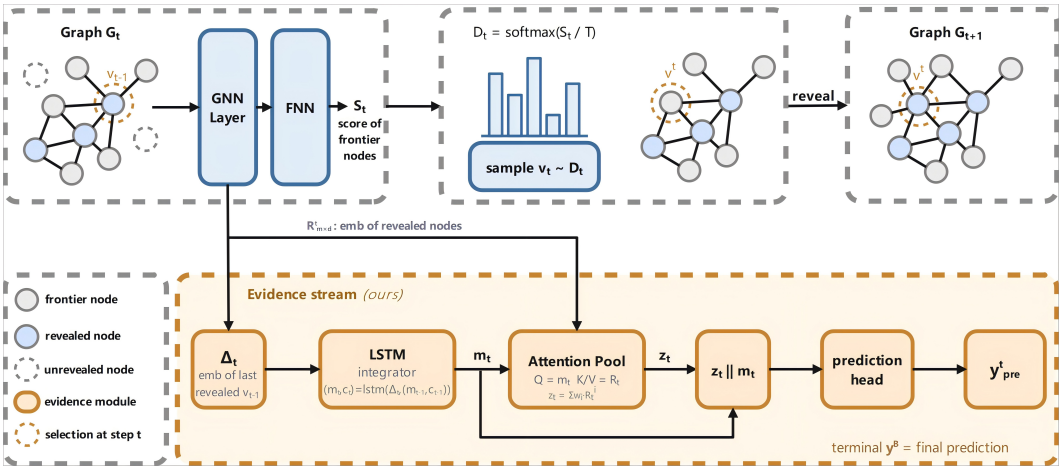

Figure 1: BAGR architecture. A shared graph encoder supports node selection and label prediction. The selection head scores frontier nodes using visible information; the prediction branch combines recurrent evidence accumulation with attention over already-read nodes.

BAGR addresses them through a tailored training procedure and an evidence-accumulation architecture. To overcome the lack of action-level supervision, we employ on-policy reinforcement learning with staged classification feedback. Initially, rewards combine the correct-label probabilities from a frozen pretrained classifier and the reader's active predictor. The frozen classifier acts as a stable reference for evidence quality, ensuring that acquisition decisions are not penalized by a predictor that is still learning. Conversely, the reader's feedback ensures the model gathers evidence it can actually interpret during inference, rather than relying solely on the external classifier's preferences. Blending these signals encourages the selection of mutually interpretable evidence and provides graded feedback even when the final prediction is incorrect. Later in training, rewards depend solely on the reader's predictions, removing the external classifier's constraint. To address the interdependence of sequential observations, a recurrent predictor maintains a summary of the reading history and attends to previously read nodes, allowing new observations to seamlessly interact with earlier evidence. Throughout training, standard graph-label supervision teaches the predictor to interpret this gathered evidence. Finally, while imitation of teacher-generated sequences provides a task-informed initialization for exploration, our RL procedure achieves comparable performance even without it.

4.1 Reader Architecture

BAGR comprises two branches with complementary roles (Figure 1). The selection branch scores unread candidate nodes based on their visible context. The prediction branch maintains a running summary of the reading history and uses it to attend to useful evidence among the nodes already read. Both branches share a graph encoder, allowing graph-label supervision to shape the representations used for selection. While recurrent memory is utilized for prediction, the selection branch computes each decision statelessly, relying solely on the current observation and the remaining budget.

Encoding visible information. The reader must evaluate candidates before observing their node features. Their topological connections to the observed graph give critical cues. Thus, at step t we encode the read nodes S_t and frontier nodes F_t together with the visible connections. Read nodes use observed features, and frontier nodes start from zero vectors, so their representations describe only visible context. A GINE encoder (Hu et al., 2020c) gives a representation $\boldsymbol{e}_v^t$ for each node v.

Message directions distinguish observed evidence from candidate context. Messages travel in both directions between read nodes, allowing acquired observations to inform one another. At the frontier, messages travel only from read nodes to unread candidates, so that a candidate can summarize its observed neighborhood, but its inferred representation is not passed back as if it were newly acquired evidence. Connections between two unread nodes are excluded because they are not visible. The same encoder pass thus supports both candidate scoring and prediction from read nodes.

Selecting the next node. We score a candidate $v \in F_t$ using its representation $\boldsymbol{e}_v^t$, the remaining budget fraction $\rho_t = (B - t)/B$, and visible structural descriptors $\boldsymbol{\phi}_v^t$ of its connections to the observed graph. These descriptors provide direct structural cues without requiring the encoder to

recover them. An MLP produces a score, which is normalized over the frontier:

$$s_v^t = \mathrm{MLP}_{\mathrm{sel}}([\boldsymbol{e}_v^t; \rho_t; \boldsymbol{\phi}_v^t]), \qquad \pi_\theta(v \mid \boldsymbol{o}_t, \rho_t) = \frac{\exp(s_v^t/T)}{\sum_{u \in F_t} \exp(s_u^t/T)}.$$

Here, $[\,;\,]$ denotes concatenation, θ denotes the trainable reader parameters, and T controls the sampling temperature. The policy samples candidates during reinforcement learning to explore alternative reading sequences and selects the highest-scoring candidate at evaluation. Scores are recomputed after each acquisition, so the policy can respond to newly observed information without a recurrent selection state. We omit the budget argument in the training equations below for brevity.

Accumulating evidence and predicting the label. A node that is ambiguous when first encountered may become informative later. Thus, we maintain a recurrent summary of the reading history. Let v_t be the most recently read node, and let $\Delta_t = e_{v_t}^t$ be its representation from the shared encoder. An LSTM (Hochreiter & Schmidhuber, 1997) combines this new evidence with the previous summary:

$$(\boldsymbol{m}_t, \boldsymbol{c}_t) = \mathrm{LSTM}(\Delta_t, (\boldsymbol{m}_{t-1}, \boldsymbol{c}_{t-1})).$$

The hidden state $\boldsymbol{m}_t$ summarizes the reading history, and $\boldsymbol{c}_t$ is the LSTM cell state. This recurrent path allows observations from different steps to interact even when their nodes are far away.

A fixed-size summary may lose useful details, and later observations can change which earlier nodes matter. We therefore use $\boldsymbol{m}_t$ to guide attention over the current representations of all read nodes:

$$\boldsymbol{z}_t = \mathrm{Attention}\big(\boldsymbol{m}_t, \{\boldsymbol{e}_v^t : v \in S_t\}\big), \qquad p_\theta(y \mid \boldsymbol{h}_t) = \mathrm{softmax}\big(\mathrm{MLP}_{\mathrm{pred}}([\boldsymbol{z}_t; \boldsymbol{m}_t])\big).$$

Here, $\boldsymbol{h}_t$ is the reading history defined in Section 3. The attention layer uses learned projections and single-head scaled dot-product attention (Vaswani et al., 2017), with $\boldsymbol{m}_t$ as the query and read-node representations as keys and values (Ilse et al., 2018). The head thus receives both accumulated context and evidence selected in light of that context. Only read nodes enter this pooling. A final encoder and prediction pass incorporates the last acquired node before producing the graph label.

4.2 Training the Reader

Because graph labels do not supply target acquisition actions, the policy must be learned from the classification outcomes of its sampled reading sequences. A primary difficulty is that the predictor evaluating these outcomes is concurrently being trained. Relying solely on its early mistakes may discourage useful reading choices. Conversely, relying permanently on an external classifier might bias the reader toward evidence it cannot independently use. Our training procedure addresses this tension through two stages of reinforcement learning, while graph-label supervision continuously teaches the predictor to interpret the collected evidence. Imitation learning provides an initial policy and predictor, but remains independent of the RL stages.

Imitation-based initialization. We create teacher reading sequences with structural rules, classifier-gradient saliency (Simonyan et al., 2014), simulated reductions in classifier uncertainty (cf. information-based acquisition (Ma et al., 2019)), and combinations of them. These offline teachers can use information hidden from the deployed reader, as in learning with privileged information (Vapnik & Vashist, 2009). A pretrained classifier $\hat{f}$ evaluates each sequence's final revealed subgraph, and we only keep sequences for which it predicts the correct graph label. Let $\mathcal{D}$ be the demonstrations and a_t^τ the choice at step t of sequence τ. We train selection and prediction together:

$$\mathcal{L}_{\mathrm{act}} = \mathrm{mean}_{(\tau,t)\in\mathcal{D}}[-\log \pi_\theta(a_t^\tau \mid \boldsymbol{o}_t^\tau)], \qquad \mathcal{L}_{\mathrm{ev}} = \mathrm{mean}_{(\tau,t)}[-\log p_\theta(y_\tau \mid \boldsymbol{h}_t^\tau)].$$

The action loss teaches the demonstrated choices. The prediction loss uses graph labels to supervise predictions. Demonstrations give a starting point, not a constraint on later exploration. On-policy refinement lets the reader learn from states reached by its own choices, which can differ from the demonstrations (Ross et al., 2011). Reinforcement learning can also be run without the initialization.

Reinforcement learning with staged feedback. The policy now samples its own reading sequences under the observation and budget constraints. Rewards evaluate the evidence acquired when each sequence finishes. Both stages retain classification supervision for the reader's predictor, but differ in how they judge acquisition quality.

Stage 1: Combining reference and reader feedback. Early in training, the reader may fail to recognize useful evidence even when it has acquired it. We therefore include the judgment of the

pretrained classifier $\hat{f}$ as a reference whose parameters remain fixed. We also include the reader's own prediction so that the policy is encouraged to collect evidence it can learn to use. For a completed sequence τ with graph label y_τ, let q_τ be the probability that $\hat{f}$ assigns to y_τ using only the final revealed subgraph, and let $p_\tau = p_\theta(y_\tau \mid \boldsymbol{h}_B^\tau)$ be the reader's probability. We use

$$R_\tau^{(1)} = q_\tau \, p_\tau.$$

Rather than enforcing a strict binary requirement for both models to predict the exact label, this product distinguishes degrees of support for the correct class. It delivers graded feedback while using a fixed evaluator as an anchor, without strictly assuming the pretrained classifier is infallible.

Stage 2: Focusing on the reader's own predictions. The frozen classifier provides guidance, but its preferences need not match those of the recurrent predictor used at inference. Continuing to demand its agreement might therefore bias the reader toward evidence tailored solely for that external model. In the second stage, we remove this dependence and reward the reader's own classification:

$$R_\tau^{(2)} = \begin{cases} r_+, & \arg\max_y p_\theta(y \mid \boldsymbol{h}_B^\tau) = y_\tau, \\ r_-, & \text{otherwise}, \end{cases}$$

where $r_+ > 0$ and $r_- < 0$ specify rewards for correct and incorrect predictions. This stage aligns acquisition with the predictor that will actually use the evidence at test time. The two stages use complementary feedback, which prevents the agent from either collapsing under an immature predictor early in training or remaining permanently constrained by a surrogate model at convergence.

Updating acquisition and prediction together. For a batch $\mathcal{B}$ of sequences, we standardize rewards by subtracting their mean and dividing by their standard deviation. The resulting advantage A_τ measures how a sequence performs relative to the others. The policy objective encourages choices in above-average sequences and discourages those in below-average ones (Williams, 1992):

$$\mathcal{L}_{\mathrm{RL}} = -\frac{1}{|\mathcal{B}|} \sum_{\tau \in \mathcal{B}} A_\tau \sum_{t \in \mathcal{I}_\tau} \log \pi_\theta(a_t^\tau \mid o_t^\tau), \qquad \mathcal{L}^{\mathrm{total}} = \mathcal{L}_{\mathrm{RL}} + \lambda_{\mathrm{ev}} \mathcal{L}_{\mathrm{ev}}.$$

Here, $\mathcal{I}_\tau$ includes only choices made by the policy. We continue applying $\mathcal{L}_{\mathrm{ev}}$ to intermediate and final predictions on the policy's own sequences. As acquisition decisions change, this supervision teaches the predictor to interpret the new evidence rather than remain specialized to teacher examples. These prediction losses provide supervision during reading, but the policy reward still scores the completed sequence, and it does not identify the contribution of each individual choice.

Importantly, gradient paths are decoupled. The selection head receives policy-gradient updates, whereas the recurrent accumulator, attention pooling, and prediction head are updated via classification gradients. The shared encoder integrates both signals. The frozen classifier is used only during training and is not required by the deployed reader. Model details are given in Appendix A.0.

Table 1: Dataset statistics.

Dataset	#Graphs	Avg. #Nodes	Avg. #Edges
CIFAR10	60,000	117	941
COLLAB	5,000	74	9,828
OGBG-PPA	158,100	243	2,266

Table 2: Analysis of reader architecture (%).

Dataset	Variant	5%	10%	15%
CIFAR10	full	18.35	20.07	20.45
	w/o pool	17.16	18.09	19.01
	w/o LSTM	17.56	18.26	19.15
	w/o both	16.91	18.58	19.38
Dataset	Variant	10%	15%	20%
COLLAB	full	16.42	19.96	25.60
	w/o pool	15.72	18.41	21.11
	w/o LSTM	15.72	18.21	20.11
	w/o both	15.88	18.04	20.64

5 EXPERIMENTS

We evaluate BAGR on three graph classification datasets to study its performance under limited observation budgets and the contribution of its reader architecture.

5.1 Experiment Settings

Datasets. We use CIFAR10 superpixels (Dwivedi et al., 2023), COLLAB (Yanardag & Vishwanathan, 2015), and OGBG-PPA (Hu et al., 2020b). CIFAR10 represents images as graphs of superpixels. COLLAB contains scientific collaboration networks. OGBG-PPA gives protein association networks from 37 taxonomic groups. These datasets let us study reading from graphs with different sources of predictive information. We use the supplied train/test splits for CIFAR10 and the OGB splits for PPA. For COLLAB, we construct an 80/10/10 train/validation/test split.

Compared Algorithms. We compare BAGR with four fixed reading strategies, including Random, breadth-first search (BFS), Max-Frontier-Degree, and Random-Walk. They explore the graph through random choices, discovery order, connections to observed regions, and local walks. As they lack a prediction head, the subgraphs they acquire are evaluated by the shared frozen classifier.

We also compare with SAGPool (Lee et al., 2019), Soft-Mask (Yang et al., 2021), and AgentNet (Martinkus et al., 2023). SAGPool and Soft-Mask learn node scores for selecting or weighting graph information, while AgentNet learns where to move and how to predict the graph label. We adapt these methods to select unread frontier nodes using the visible information specified in Section 3. In particular, the pooling scorers are trained on complete graphs and applied to partially observed graphs at evaluation. Following their original designs, the final prediction is made by each method's own prediction head on the acquired subgraph (Appendix A.2). For our method, BAGR-BC denotes the reader after imitation learning, and BAGR-RL denotes the reader after reinforcement fine-tuning.

Evaluation. We report test accuracy after reading under each budget. A budget specifies the number of additional nodes that can be read as a fraction of the graph size. We evaluate CIFAR10 at 5/10/15% budgets and COLLAB at 10/15/20%. As PPA is much larger, we only consider 5%. Starting nodes are chosen reproducibly, and accuracy is averaged over the evaluated graph. The reported run in the main comparisons uses four starting nodes per graph by default.

Experimental Setup. BAGR uses a shared GINE encoder, a node-selection head, and a recurrent evidence predictor, as described in Section 4. The default architecture has two encoder layers with hidden dimension 128 and dropout 0.1 (Srivastava et al., 2014), and an LSTM and attention module of the same hidden dimension. We optimize the imitation and reinforcement objectives with Adam (Kingma & Ba, 2015). During reinforcement training, the reader samples its next node from the policy; during evaluation, it chooses the highest-scoring candidate. The correctness-reward preset uses $r_+ = 1$ and $r_- = -0.1$. The reward source and training schedule can differ across experiments, as discussed in Section 4.2. Further implementation details are given in Appendix A.0.

Table 3: Test accuracy (%) under different node budgets. BC and RL denote BAGR after imitation learning and reinforcement fine-tuning. BAGR achieves the best overall results.

	CIFAR10 superpixels			COLLAB			OGBG-PPA
Method / Budget	5%	10%	15%	10%	15%	20%	5%
Random	16.32	18.69	20.57	15.57	15.57	15.67	20.16
BFS	16.32	18.06	19.59	15.57	15.57	16.47	24.45
Max-Frontier-Degree	16.09	17.31	18.98	15.57	15.72	17.27	26.20
Random-Walk	16.30	19.03	21.36	15.57	15.57	15.62	21.46
SAGPool	10.92	12.52	13.81	16.02	16.37	15.62	10.08
Soft-Mask	12.88	15.34	17.32	**51.90**	51.90	51.90	9.88
AgentNet	16.25	18.30	19.11	**51.90**	51.90	51.90	9.48
BAGR-BC	16.91	17.87	18.39	16.42	19.96	25.60	3.97
BAGR-RL	**27.79**	**33.84**	**36.55**	**51.90**	**53.44**	**55.74**	**26.90**
Full-graph reference		60.50			69.06		65.60

5.2 Results

Comparison against Existing Methods. Table 3 shows that BAGR-RL outperforms all compared methods in most settings. Its advantage is particularly clear on CIFAR10, while on COLLAB it matches or exceeds the strongest baseline. On PPA, it also achieves the highest accuracy among the

compared budgeted methods. Overall, these results support the effectiveness of learning to acquire and interpret graph evidence under limited observation budgets. BAGR-RL also substantially outperforms BAGR-BC across all datasets and budgets. These consistent gains highlight the value of reinforcement fine-tuning. Imitation provides a useful starting point, while further training on the reader's own trajectories improves the overall reading and prediction performance.

Even with a 5% node budget, BAGR-RL recovers a meaningful fraction of the full-graph reference accuracy on CIFAR10 and PPA. Although full observation remains more accurate, these results show that BAGR can collect useful predictive evidence from only a small part of a graph. Its performance further improves as more observations become available on CIFAR10 and COLLAB.

Ablation Study of Imitation Learning. To isolate the impact of teacher demonstrations, we evaluate reinforcement learning both with and without imitation-based initialization (Figure 2, top row). On COLLAB, imitation provides a distinct advantage during the early phases of reinforcement training, though this performance gap steadily diminishes as training progresses. On CIFAR10, the learning curves for both settings exhibit similar trajectories and even intersect. Importantly, both approaches demonstrate continuous improvement via on-policy learning, confirming that effective active reading strategies can emerge without prior demonstrations. The findings indicate that while imitation learning gives a robust, task-informed starting point, it is not a strict prerequisite for BAGR.

Analysis of Two-stage Reinforcement Learning. Figure 2 (bottom row) investigates the transition from the blended classifier–reader feedback to the reader-only reward. On both datasets, accuracy improves steadily during the initial phase and continues to climb after the reward switch, notably without any disruptive performance drop at the transition boundary. This dynamic validates the intended design of our two-stage schedule. The frozen classifier acts as a stabilizing anchor for early exploration, while the subsequent stage smoothly realigns the acquisition policy with the reader's test-time predictor. Thus, agents continue to refine their strategy after the external guidance is lifted, proving that they avoid becoming permanently constrained by the surrogate classifier's preferences.

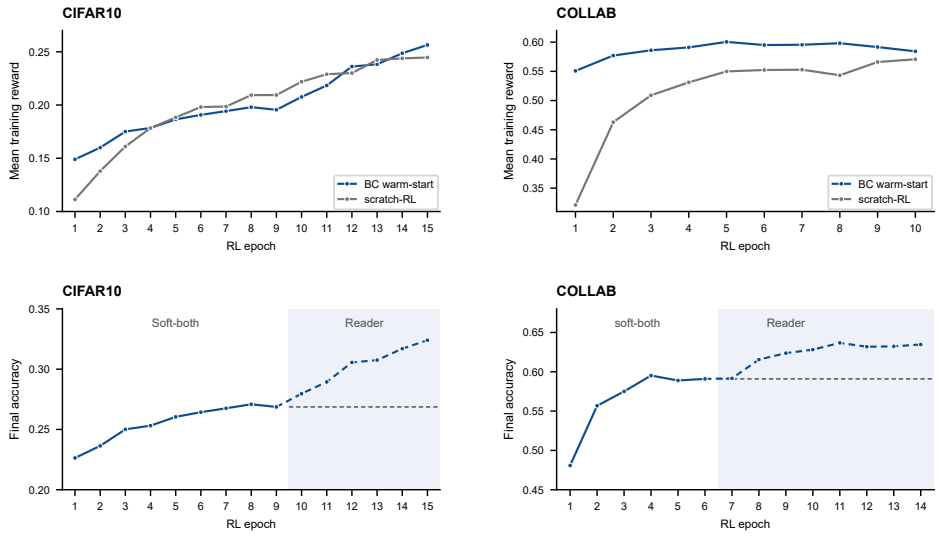

Figure 2: Imitation initialization and staged reinforcement learning. Top: training reward with and without imitation initialization. Bottom: final-step accuracy before and after switching from combined classifier–reader feedback (soft-both) to reader-only rewards (Reader, shaded region).

Effect of the Reader Architecture. Table 2 ablates the recurrent memory (LSTM) and history-guided attention pooling modules following imitation training. The complete reader architecture consistently achieves the highest performance across all reported budgets on both datasets. On COLLAB, the performance penalty for removing the recurrent memory increases at larger budgets, suggesting that retaining temporal context is crucial for processing longer reading trajectories. Conversely, on CIFAR10, the absence of attention pooling is more detrimental than the removal of memory, emphasizing the importance of dynamically isolating relevant evidence from the pool of read nodes. Discarding both components does not consistently result in the largest performance

drop, indicating that their contributions are intertwined rather than strictly additive. These findings validate the synergy of recurrent context accumulation and attention-based evidence retrieval.

Patterns in Node Selection. Figure 3 analyzes the structural properties and predictive impact of the reader's acquisitions. A candidate's *frontier degree* quantifies its connections to already-read nodes. On CIFAR10, the policy initially favors nodes with higher-than-average frontier degrees, but shifts toward candidates with fewer such connections in later steps. On COLLAB, selected nodes consistently exhibit higher connectivity to the read subgraph than the average candidate. When evaluating the impact of these choices on the frozen classifier's predictive entropy, we see that reading these nodes reduces average uncertainty on CIFAR10, yet slightly increases it during most steps on COLLAB. This divergence indicates that the learned policy does not merely optimize for immediate reductions in classifier uncertainty. Furthermore, as predictive entropy measures model confidence rather than correctness, temporary increases in uncertainty do not preclude accurate final predictions.

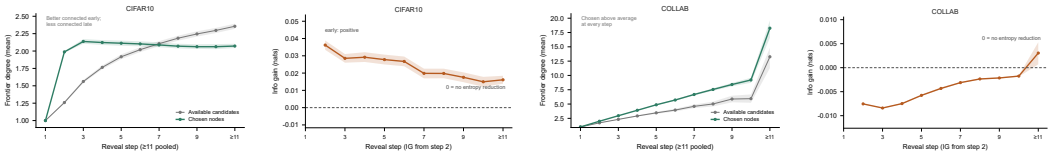

Figure 3: Node-selection patterns on CIFAR10 (left pair) and COLLAB (right pair). For each dataset, the first plot compares the mean frontier degree (number of already-read neighbors) between selected nodes and available candidates. The second plot tracks the mean change in the frozen classifier's uncertainty. Information gain is defined as the reduction in predictive entropy following an acquisition (positive values indicate reduced uncertainty). Steps $t \geq 11$ are aggregated.

Case Studies of Reading Decisions. Figure 4 shows the structural consequences of specific node acquisitions. In the CIFAR10 example, the selected candidate shares only a single edge with the previously read subgraph. Acquiring it reveals previously hidden edges connected to new unread nodes, thereby expanding the visible frontier and unlocking new directions for exploration. In contrast, the selected node in the COLLAB example connects to all already-read nodes, forming multiple local triangles. While these dense structural connections were already visible prior to acquisition, reading the node actively incorporates its features into the evidence pool, yielding a highly cohesive read subgraph for the predictor. These examples highlight two strategic roles of node acquisition, i.e., expanding visible exploratory frontier versus assembling densely connected evidence.

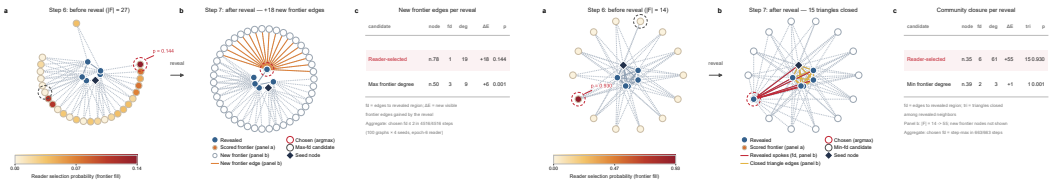

Figure 4: Examples of the graph topology before and after acquiring the node marked by the red dashed circle. Left: acquiring the selected node reveals new orange edges, thereby expanding the visible frontier. Right: the selected node is integrated into a dense cluster of read nodes, with highlighted edges emphasizing the formation of triangles within the acquired evidence subgraph.

6 Conclusion

We investigated budget-limited active graph reading. We introduced BAGR, a novel framework that jointly learns dynamic node selection and evidence interpretation under explicit budget constraints. Extensive experiments on CIFAR10 superpixels, COLLAB, and OGBG-PPA demonstrate that BAGR significantly outperforms baseline methods across various budgeted settings, with our reinforcement learning strategy yielding substantial gains over imitation alone. These results underscore the critical value of jointly learning where to read and how to integrate dynamically acquired evidence. Future work will explore scalable candidate selection for graphs with massive frontiers and evaluate BAGR in real-world domains with explicit information-acquisition costs.

AI USE STATEMENT

The core research idea underlying this paper was conceived by the human authors, not generated by large language models (LLMs). In developing the work, we used generative AI tools to assist with literature discovery and review, provide feedback on the problem formulation and experimental design, and discuss the interpretation of experimental results. LLM-based coding assistants also supported the implementation, debugging, and refactoring of parts of the teacher-trajectory generation, training, and evaluation pipelines. The authors made the final methodological decisions and remained responsible for conducting the experiments.

We additionally used generative AI tools to draft selected sections, suggest improvements to the organization of the paper, and improve the clarity and readability of the manuscript. These drafts and suggestions were subsequently revised by the authors. No reported experiment used LLM-generated synthetic data.

AI-assisted text and code were reviewed by the authors. We take full responsibility for the final content of this work, including its methods, experimental results, claims, references, and code. No AI tool is credited as an author.

REPRODUCIBILITY STATEMENT

Our source code is available in an anonymous repository. It includes the reader implementation, teacher-trajectory generation and filtering, training and evaluation scripts, adapted baselines, and environment setup instructions. Section 3 defines the observation protocol and reading budget, and Section 4 describes the reader architecture and training objectives. Section 5.1 reports the datasets, splits, evaluation budgets, and baseline settings. Appendices A.0, A.1, and A.2 provide implementation details, training pseudocode, and baseline adaptations; Appendix A.3 collects additional evaluation and training settings. Trained checkpoints will be released upon acceptance.

REFERENCES

Linus Aronsson, Arman Rahbar, and Morteza Haghir Chehreghani. A survey on active feature acquisition strategies. *arXiv preprint arXiv:2502.11067*, 2025. URL https://arxiv.org/abs/2502.11067.

Nicolò Cesa-Bianchi, Shai Shalev-Shwartz, and Ohad Shamir. Efficient learning with partially observed attributes. *Journal of Machine Learning Research*, 12:2857–2878, 2011. URL https://jmlr.org/papers/v12/cesa-bianchi11a.html.

Vijay Prakash Dwivedi, Chaitanya K. Joshi, Anh Tuan Luu, Thomas Laurent, Yoshua Bengio, and Xavier Bresson. Benchmarking graph neural networks. *Journal of Machine Learning Research*, 24(43):1–48, 2023. URL https://jmlr.org/papers/v24/22-0567.html.

Hongyang Gao and Shuiwang Ji. Graph U-nets. In *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pp. 2083–2092, 2019. URL https://proceedings.mlr.press/v97/gao19a.html.

Daniele Grattarola, Daniele Zambon, Filippo Maria Bianchi, and Cesare Alippi. Understanding pooling in graph neural networks. *IEEE Transactions on Neural Networks and Learning Systems*, 35(2):2708–2718, 2024. doi: 10.1109/TNNLS.2022.3190922.

Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural Computation*, 9(8): 1735–1780, 1997. doi: 10.1162/neco.1997.9.8.1735.

Shengding Hu, Zheng Xiong, Meng Qu, Xingdi Yuan, Marc-Alexandre Côté, Zhiyuan Liu, and Jian Tang. Graph policy network for transferable active learning on graphs. In *Advances in Neural Information Processing Systems*, volume 33, 2020a. URL https://arxiv.org/abs/2006.13463.

Weihua Hu, Matthias Fey, Marinka Zitnik, Yuxiao Dong, Hongyu Ren, Bowen Liu, Michele Catasta, and Jure Leskovec. Open graph benchmark: Datasets for machine learning on graphs. In *Advances in Neural Information Processing Systems*, volume 33, 2020b. URL https://arxiv.org/abs/2005.00687.

Weihua Hu, Bowen Liu, Joseph Gomes, Marinka Zitnik, Percy Liang, Vijay Pande, and Jure Leskovec. Strategies for pre-training graph neural networks. In *International Conference on Learning Representations*, 2020c. URL https://arxiv.org/abs/1905.12265.

Jianming Huang and Hiroyuki Kasai. Self-interpretable subgraph neural network with deep reinforcement walk exploration. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 40, pp. 21984–21993, 2026. doi: 10.1609/aaai.v40i26.39352. URL https://ojs.aaai.org/index.php/AAAI/article/view/39352.

Maximilian Ilse, Jakub M. Tomczak, and Max Welling. Attention-based deep multiple instance learning. In *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pp. 2127–2136. PMLR, 2018. URL https://proceedings.mlr.press/v80/ilse18a.html.

Jaromír Janisch, Tomáš Pevný, and Viliam Lisý. Classification with costly features using deep reinforcement learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pp. 3959–3966, 2019. doi: 10.1609/aaai.v33i01.33013959.

Jaromír Janisch, Tomáš Pevný, and Viliam Lisý. Classification with costly features in hierarchical deep sets. *Machine Learning*, 113:4487–4522, 2024. doi: 10.1007/s10994-024-06565-4.

Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *International Conference on Learning Representations*, 2015. URL https://arxiv.org/abs/1412.6980.

John Boaz Lee, Ryan Rossi, and Xiangnan Kong. Graph classification using structural attention. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2018. doi: 10.1145/3219819.3219980. URL https://ryanrossi.com/pubs/KDD18-graph-attention-model.pdf.

Junhyun Lee, Inyeop Lee, and Jaewoo Kang. Self-attention graph pooling. In *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pp. 3734–3743. PMLR, 2019. URL https://proceedings.mlr.press/v97/lee19c.html.

Dongsheng Luo, Wei Cheng, Dongkuan Xu, Wenchao Yu, Bo Zong, Haifeng Chen, and Xiang Zhang. Parameterized explainer for graph neural network. In *Advances in Neural Information Processing Systems*, volume 33, 2020. URL https://papers.nips.cc/paper/2020/hash/e37b08dd3015330dcbb5d6663667b8b8-Abstract.html.

Chao Ma, Sebastian Tschiatschek, Konstantina Palla, Jose Miguel Hernandez-Lobato, Sebastian Nowozin, and Cheng Zhang. EDDI: Efficient dynamic discovery of high-value information with partial VAE. In *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pp. 4234–4243. PMLR, 2019. URL https://proceedings.mlr.press/v97/ma19c.html.

Karolis Martinkus, Pál András Papp, Benedikt Schesch, and Roger Wattenhofer. Agent-based graph neural networks. In *International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=8WTAh0tj2jC.

Volodymyr Mnih, Nicolas Heess, Alex Graves, and Koray Kavukcuoglu. Recurrent models of visual attention. In *Advances in Neural Information Processing Systems*, volume 27, 2014. URL https://papers.nips.cc/paper_files/paper/2014/hash/3e456b31302cf8210edd4029292a40ad-Abstract.html.

Ekagra Ranjan, Soumya Sanyal, and Partha Talukdar. ASAP: Adaptive structure aware pooling for learning hierarchical graph representations. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pp. 5470–5477, 2020. doi: 10.1609/aaai.v34i04.5997.

Stephane Ross, Geoffrey Gordon, and Drew Bagnell. A reduction of imitation learning and structured prediction to no-regret online learning. In *Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics*, volume 15 of *Proceedings of Machine Learning Research*, pp. 627–635, 2011. URL https://proceedings.mlr.press/v15/ross11a.html.

Hajin Shim, Sung Ju Hwang, and Eunho Yang. Why pay more when you can pay less: A joint learning framework for active feature acquisition and classification. *arXiv preprint arXiv:1709.05964*, 2017. URL https://arxiv.org/abs/1709.05964.

Karen Simonyan, Andrea Vedaldi, and Andrew Zisserman. Deep inside convolutional networks: Visualising image classification models and saliency maps. In *International Conference on Learning Representations Workshop*, 2014. URL https://arxiv.org/abs/1312.6034.

Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: A simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15:1929–1958, 2014. URL https://www.jmlr.org/papers/v15/srivastava14a.html.

Vladimir Vapnik and Akshay Vashist. A new learning paradigm: Learning using privileged information. *Neural Networks*, 22(5–6):544–557, 2009. doi: 10.1016/j.neunet.2009.06.042.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in Neural Information Processing Systems*, volume 30, pp. 5998–6008, 2017.

Yulin Wang, Kangchen Lv, Rui Huang, Shiji Song, Le Yang, and Gao Huang. Glance and focus: a dynamic approach to reducing spatial redundancy in image classification. In *Advances in Neural Information Processing Systems*, volume 33, 2020. URL https://arxiv.org/abs/2010.05300.

Ronald J. Williams. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine Learning*, 8(3–4):229–256, 1992. doi: 10.1007/BF00992696.

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

Pinar Yanardag and S. V. N. Vishwanathan. Deep graph kernels. In *Proceedings of the 21st ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 1365–1374, 2015. doi: 10.1145/2783258.2783417.

Mingqi Yang, Yanming Shen, Heng Qi, and Baocai Yin. Soft-mask: Adaptive substructure extractions for graph neural networks. In *Proceedings of the Web Conference 2021*, 2021. doi: 10.1145/3442381.3449929. URL https://arxiv.org/abs/2206.05499.

Zhitao Ying, Jiaxuan You, Christopher Morris, Xiang Ren, William L. Hamilton, and Jure Leskovec. Hierarchical graph representation learning with differentiable pooling. In *Advances in Neural Information Processing Systems*, volume 31, 2018. URL https://proceedings.neurips.cc/paper/2018/hash/e77dbaf6759253c7c6d0efc5690369c7-Abstract.html.

Zhitao Ying, Dylan Bourgeois, Jiaxuan You, Marinka Zitnik, and Jure Leskovec. GNNExplainer: Generating explanations for graph neural networks. In *Advances in Neural Information Processing Systems*, volume 32, 2019. URL https://proceedings.neurips.cc/paper/2019/hash/d80b7040b773199015de6d3b4293c8ff-Abstract.html.

Junchi Yu, Tingyang Xu, Yu Rong, Yatao Bian, Junzhou Huang, and Ran He. Graph information bottleneck for subgraph recognition. In *International Conference on Learning Representations*, 2021. URL https://arxiv.org/abs/2010.05563.

Hao Yuan, Haiyang Yu, Jie Wang, Kang Li, and Shuiwang Ji. On explainability of graph neural networks via subgraph explorations. In *Proceedings of the 38th International Conference on Machine Learning*, volume 139 of *Proceedings of Machine Learning Research*, pp. 12241–12252. PMLR, 2021. URL https://proceedings.mlr.press/v139/yuan21c.html.

Hao Yuan, Haiyang Yu, Shurui Gui, and Shuiwang Ji. Explainability in graph neural networks: A taxonomic survey. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 45(5): 5782–5799, 2023. doi: 10.1109/TPAMI.2022.3204236.

Wentao Zhang, Yexin Wang, Zhenbang You, Meng Cao, Ping Huang, Jiulong Shan, Zhi Yang, and Bin Cui. Information gain propagation: A new way to graph active learning with soft labels. In *International Conference on Learning Representations*, 2022. URL https://openreview.net/forum?id=USC0-nvGPK.

This appendix collects implementation details (A.0), the training pipeline (A.1), baseline adaptations (A.2), additional reproducibility details (A.3), and additional experimental results (A.4).

A.0 Implementation Details

The shared encoder is a GINE-style edge-feature convolution, by default 2 layers of width 128 with dropout 0.1 (configurable layer-wise); the selection MLP and the evidence modules (LSTM $128 \to 128$, single-head attention pooling, evidence head $2d \to d \to K$) operate at the same width; the evidence modules add about 0.2M parameters over base, with forward cost comparable to base (Δ_t comes from the same forward pass, so there is no second encoder). Optimization uses Adam (representative configuration: learning rate 10^{-3}, batch size 64, about 10 BC epochs, with RL fine-tuned for a few more epochs); training rollouts sample at temperature and evaluation takes the argmax, with 4 deterministic seeds per graph (§3). Budget tiers, class counts, and feature dimensions per dataset are given by the dataset registry; all hyperparameters and architectures are embedded in the corresponding checkpoints and released with the code, and per-dataset differences are not enumerated here.

A.1 Training Pipeline (Algorithm 1)

The pseudocode below gives the three-stage pipeline of §4.2; budget accounting and the terminal judgment follow the protocol of §3.

A.2 How Baselines Enter the Protocol

Fixed strategies. The four fixed strategies, namely Random, BFS, Max-Frontier-Degree, and Random-Walk, share one protocol interface: at each step they select exactly one unrevealed endpoint from the current frontier, with budget accounting and seed formulas identical to the learned methods, and the frozen upper-bound classifier as their shared terminal judge; they differ only in the **selection rule** that orders reading. Random samples uniformly over frontier edges and reveals their unrevealed endpoint; BFS reveals in first-in-first-out order of frontier-edge discovery; Max-Frontier-Degree reveals the frontier node with the most connections to the revealed subgraph; Random-Walk walks along random edges from the current node and reveals the first unrevealed node it arrives at.

SAGPool. SAGPool (Lee et al., 2019) requires substantive adaptation to enter the protocol: its original form performs one-shot top-k selection on the complete graph, consuming unobserved global structure and guaranteeing neither connectivity to the seed. The adapted version turns its scorer into a stepwise static selector: at each step it recomputes gating scores on the visible subgraph only (revealed $\cup$ frontier, with frontier node features zeroed) and reveals the highest-scoring frontier node (ties broken by smaller id, deterministic); the scorer itself is still trained on complete graphs with standard supervision, and the resulting distribution shift at evaluation is part of that baseline's setting. Consistent with its original design, in which the pooling layers and the classification head are learned jointly, the adapted SAGPool is judged by its own classification head on the revealed subgraph.

Soft-Mask. Soft-Mask (Yang et al., 2021) is the soft-weighting counterpart of SAGPool: a mask generator outputs a continuous gate $m_v \in (0, 1)$ for every node at each layer, message passing is modulated by $x \odot m$, and the masks are supervised only indirectly by the classification loss on complete graphs, with no budget or temporal notion. Its adaptation mirrors SAGPool's two stages: the scorer is trained on complete graphs, and at inference the first-layer mask scores are recomputed on the visible subgraph at every step to reveal the highest-scoring frontier node. Together the two adapted poolers form a paired contrast between soft weighting and hard top-k selection. As with SAGPool, the final prediction is made by Soft-Mask's own classification head on the revealed subgraph.

AgentNet. The original AgentNet (Martinkus et al., 2023) lets a population of identity-carrying agents walk the complete graph with Gumbel-softmax transitions, absorb neighbor states at every step, and read out a classification, with no budget notion since the agent count times the walk length far exceeds a 5–15% node budget. The adaptation translates the policy rather than transplanting a

Algorithm 1 Three-stage training pipeline

Input: training graphs $\{G_i\}$; frozen classifier $\hat{f}$; budget tiers B; teachers {gradient, info-gain, structural, hybrid}
Output: policy and evidence-head parameters θ

Stage A: teacher trajectories.
for each graph G_i, seed s_0, and teacher T **do**
  $S \leftarrow \{s_0\}$
  while $|S| - 1 < B$ **do**
    $F \leftarrow N(S) \setminus S$
    if $F = \emptyset$ **then**
      teleport to an unrevealed node, consuming one unit of budget
    end if
    $a \leftarrow T(\text{observation})$; reveal a; $S \leftarrow S \cup \{a\}$; record (o_t, a_t)
  end while
  if $\hat{f}(G_S) = y$ **then**
    $\mathcal{D} \leftarrow \mathcal{D} \cup \{\text{this trajectory}\}$
  end if
end for
class-balance $\mathcal{D}$
Stage B: behavior cloning.
repeat
  sample a batch from $\mathcal{D}$, reproduce observations by temperature rollout, descend on $\mathcal{L}_{BC}$
until converged
Stage C1: reinforcement fine-tuning with soft-both reward.
repeat
  roll out with θ at temperature; $R \leftarrow \hat{q}\,\hat{p}_\theta(y \mid \boldsymbol{h}_B)$ where $\hat{q}$ is the frozen classifier's probability of y on o_B and $\hat{p}_\theta$ is the reader's; standardize advantages
  descend on $\mathcal{L}^{\text{total}}$
until converged
Stage C2: reinforcement fine-tuning with reader-only reward.
repeat
  roll out with θ at temperature; $R \leftarrow +1$ if the reader predicts y on o_B, otherwise -0.1; standardize advantages
  descend on $\mathcal{L}^{\text{total}}$
until converged

scorer: one agent walking one step corresponds to revealing one frontier node, candidates are restricted to the frontier neighbors of the current position, the initial position is the seed, and when no frontier candidate remains the agent navigates within the revealed region by its exploration score back to a position with candidates, consuming no budget. Consistent with its original design, in which the walking policy and the classification head are learned together, evaluation judges AgentNet by its own readout head.

A.3 Additional Reproducibility Details

Data and protocol. All three benchmarks are publicly available: OGBG-PPA (Hu et al., 2020b), CIFAR10 superpixels (Dwivedi et al., 2023), and COLLAB (Yanardag & Vishwanathan, 2015). Dataset splits and evaluated budgets follow Section 5.1: CIFAR10 uses the supplied splits, PPA uses the OGB splits, and COLLAB uses a constructed 80/10/10 train/validation/test split. We evaluate CIFAR10 at 5/10/15% node budgets, COLLAB at 10/15/20%, and PPA at 5%. The per-graph budget is $B = \max(1, \lfloor r \cdot |V| \rfloor)$ excluding the seed. The seed node is $(idx \times 31 + seed_i \times 17 + 7) \bmod |V|$, a deterministic function of the graph index and the selected evaluation seed. Main comparisons use four starting nodes per graph, except the reported CIFAR10 AgentNet run, which uses two. Learned policies select frontier nodes by argmax at evaluation. The visibility and budget accounting rules are specified in Section 3.

Hyperparameters and training details. Teacher trajectories are generated by gradient saliency, information gain, structural heuristics, and their hybrids, filtered by the frozen classifier's correctness on the final revealed subgraph and class-balanced. The reported training settings are listed below.

	PPA	CIFAR10 superpixels	COLLAB
num of GINE layers	2	2	2
hidden dim	256	128	128
BC optimizer / lr		Adam, 10^{-3}	
BC batch size		64	
BC epochs	10	10	10
RL algorithm / optimizer / lr		REINFORCE, Adam, 10^{-4}	
RL batch size (graphs $\times$ seeds)		32×4	
RL epochs (soft both)	10	10	10
RL epochs (reader)	5	5	4
sample temperature		0.2	
reward (asymmetric)		$+1/-0.1$	
advantage standardization		batch-level	
λ_{ev}		1	

A.4 Additional Experimental Results

This section reports additional analyses of how the reader uses the evidence it collects: the contribution of its predictor, alternative ways to combine states across reading steps, and the distribution of attention over those steps.

A.4.1 Contribution of the Evidence Predictor

To examine the role of prediction after node selection, we compare BAGR-RL's evidence predictor with the frozen classifier applied to the same acquired subgraphs. Table 4 shows that the evidence predictor achieves higher accuracy across all reported datasets and budgets. The acquired nodes are held fixed in this comparison, so the difference concerns how the two predictors use the collected evidence, rather than which nodes they select. This supports learning a predictor for partially observed graphs, although the comparison does not isolate individual architectural or training choices.

Table 4: Test accuracy (%) of two predictors on the same subgraphs acquired by BAGR-RL. The frozen-classifier results are denoted BAGR (RL-Ref); the BAGR-RL results are repeated from Table 3 for comparison.

	CIFAR10 superpixels			COLLAB			OGBG-PPA
Predictor / Budget	5%	10%	15%	10%	15%	20%	5%
Frozen classifier	16.68	19.78	22.98	15.72	18.21	20.26	20.73
BAGR-RL	**27.79**	**33.84**	**36.55**	**51.90**	**53.44**	**55.74**	**26.90**

A.4.2 Combining Evidence across Reading Steps

The standard reader predicts from its final recurrent state, which summarizes the reading history. Here, we examine whether using several states can improve prediction without changing the reading policy. We freeze the RL-trained reader and collect its state sequence for each reading trajectory. We then compare using only the final state, averaging all states with equal weights, and learning attention weights over the states. For each alternative, the prediction head is either kept fixed or retrained using the graph labels. This attention operates across reading steps and is distinct from the node-level attention pooling in the reader architecture.

Table 5 shows that learned temporal attention combined with a retrained prediction head gives the best results on both datasets. Retraining the head with uniform averaging does not improve over the original final-state prediction. Learning attention while keeping the head fixed gives little improvement on COLLAB and a smaller improvement on CIFAR10. Thus, choosing how to combine states and adapting the prediction head work better together than either change alone in these comparisons. These are post-hoc analyses of a fixed reader, not additional policy-training results.

Table 5: Readout ablation, test accuracy (%). All variants use state sequences collected from a fixed RL-trained reader. State aggregation determines which reading steps contribute to prediction; the prediction head is either frozen or retrained.

State aggregation	Prediction head	COLLAB	CIFAR10
Final state only	Frozen	53.59	29.25
Uniform averaging	Retrained	53.14	28.95
Learned attention	Retrained	**56.99**	**31.13**
Uniform averaging	Frozen	52.74	28.14
Learned attention	Frozen	53.64	30.19

A.4.3 WHICH READING STEPS RECEIVE ATTENTION?

Figure 5 shows how the temporal readout distributes attention across the collected states. On CIFAR10, attention concentrates on the final few steps, but spreads over more preceding states as the reading budget increases. On COLLAB, the final state receives most of the weight at every budget. States from the early part of the trajectory receive little weight on either dataset. This does not imply that early observations are unimportant: later recurrent states already summarize evidence collected earlier. Rather, the readout favors states formed after more evidence has been accumulated.

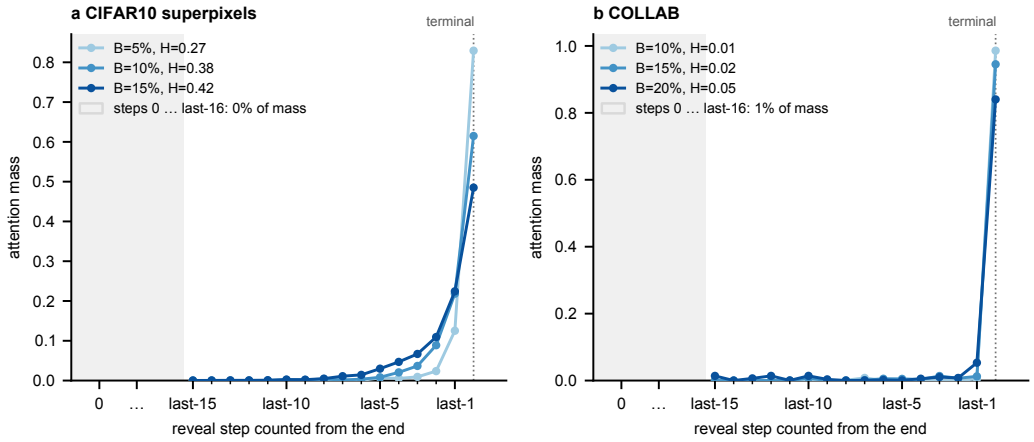

Figure 5: Attention weights over reading states on CIFAR10 (left) and COLLAB (right). Steps are aligned relative to the end of each trajectory; "terminal" marks the final state. Curves correspond to node budgets B, and H reports attention entropy. The shaded area summarizes steps before the final sixteen states, with their total attention mass indicated in the legend.