LEARNING GRAPH REPRESENTATIONS VIA PREDICTIVE LANGUAGE SUPERVISION

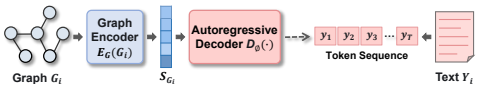
Anonymous authors
Paper under double-blind review

ABSTRACT

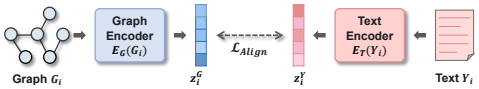
Graph–language pretraining typically relies on autoregressive token prediction or graph–text alignment. Token prediction ties supervision to a particular *linguistic realization*, while alignment may suppress information that is unnecessary for matching but useful for downstream graph tasks. We introduce **GL-JEPA**, which uses a frozen Text Encoder to define a continuous target for each description and a Predictor to reconstruct this target from graph states. Predicting the full description latent reduces dependence on surface form while encouraging graph states to capture description information that may otherwise be suppressed by alignment. For downstream adaptation, query conditioning allows the Predictor to extract task-relevant information from frozen graph representations. Compared with token prediction, GL-JEPA improves by 25.2–43.8 percentage points the rate at which two valid descriptions both outrank a negative. In a controlled diagnostic, latent prediction achieves 95.6% probe accuracy on a target-relevant factor, versus 12.7% for alignment with easy negatives. GL-JEPA also outperforms direct alignment on node-, link-, and graph-level tasks and supports query-conditioned graph question answering across nine tasks with a frozen Graph Encoder.

1 INTRODUCTION

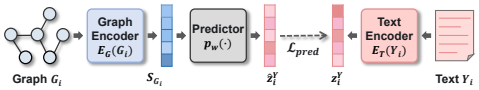
Graph–language models learn representations of entities and relations by integrating relational structure with language (Plenz & Frank, 2024; Jin et al., 2024). The graph input comprises directed structure, node-text attributes, and positional features, while the paired description is withheld from the Graph Encoder (E_G) and used only as language supervision. Unlike task-specific labels, which supervise predefined prediction targets, a paired description can capture multiple complementary aspects of a graph (Zhu et al., 2025; Kong et al., 2025). Graph–language pretraining leverages this richer supervision to learn transferable graph representations for downstream tasks such as node classification, link prediction, graph–text retrieval, and graph question answering (Wen & Fang, 2023; Zhu et al., 2025; Tang et al., 2024; Kong et al., 2025).



(a) Autoregressive Token Prediction



(b) Graph–Text Alignment



(c) Graph–Language Latent Prediction (Ours)

Figure 1: Graph–language supervision interfaces. Token prediction reconstructs an observed wording. Direct alignment matches paired graph and description representations. GL-JEPA uses a separate Predictor to infer the frozen description target from graph context.

Existing methods primarily use language supervision through two interfaces (Figure 1). Let G denote a graph with paired description Y and optional task query Q. **Autoregressive token prediction** (Figure 1a) uses a language decoder D_ϕ, conditioned on $E_G(G)$ and Q, to generate Y as $D_\phi(E_G(G), Q) \to Y$ (Tang et al., 2024; Chen et al., 2024b; Kong et al., 2025). Its loss is defined over the observed token sequence of Y, thereby supervising both graph-relevant content and its particular *linguistic realization*. The same graph semantics can be expressed through different entity orders, relation phrasings, sentence structures, lexical choices, and token orderings. Autoregressive

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.

training therefore incurs loss on surface-form variation that may be irrelevant to downstream graph tasks (Tang et al., 2024; Chen et al., 2024b; Kong et al., 2025).

Direct graph–text alignment (Figure 1b) optimizes similarity between a graph embedding and the embedding of its paired description (Wen & Fang, 2023; Brannon et al., 2024; Zhu et al., 2025; Liu et al., 2026). Let E_T denote the Text Encoder for the paired description Y. With method-specific projections, alignment directly matches $E_G(G) \leftrightarrow E_T(Y)$. This objective supervises the information needed to distinguish matched from mismatched graph–text pairs. Features that do not improve the graph–text alignment objective may be suppressed even when useful for downstream tasks, creating an *objective mismatch*. Graphs may share entities and topics yet differ in relation direction or connectivity. Matching constrains representations to information sufficient for graph–text correspondence. Downstream graph tasks may require additional structural distinctions (Luo et al., 2027; Kong et al., 2025).

Autoregressive token prediction uses the paired description as supervision but requires reconstructing a particular linguistic realization of Y. Direct graph–text alignment avoids token-level reconstruction, but may weakly supervise features that are not necessary for graph–text matching. Neither existing interface directly optimizes for both properties.

Can paired descriptions supervise graph representations without token reconstruction while reducing feature suppression?

To address this question, we introduce **GL-JEPA** (Figure 1c), which draws on joint-embedding predictive architectures (Assran et al., 2023; Chen et al., 2026) to predict a latent description target from graph context. A frozen Text Encoder E_T encodes the paired description Y into a latent description target in a continuous embedding space. The Graph Encoder E_G processes the graph structure and node-text attributes into contextualized node states and a graph summary. The Predictor P_ω predicts the description target from these graph states as $P_\omega(E_G(G)) \rightarrow E_T(Y)$. Training compares the predicted and target representations. The Graph Encoder E_G and Predictor P_ω are updated while E_T remains frozen. The paired description is used only to define the target and is not provided to the graph branch.

Because the description is predicted as a continuous latent representation rather than an observed token sequence, GL-JEPA avoids reconstructing a particular linguistic realization of (Y). Predicting the full description embedding encourages E_G to capture description-relevant graph features and may mitigate feature suppression. During downstream adaptation, P_ω additionally conditions its readout on the task query Q.

Our controlled experiments compare GL-JEPA with autoregressive token prediction and direct graph–text Alignment, and evaluate feature suppression using a synthetic diagnostic. Compared with token prediction, GL-JEPA achieves stronger language-prediction performance while reducing measured training-update cost by 66.0%, including generation-readout fitting. Compared with Alignment, GL-JEPA improves Node accuracy by 3.59 points and Link AUC by 3.82 points, whereas Alignment performs better on bidirectional Retrieval R@1. In the feature-suppression diagnostic, a target-relevant factor is more recoverable from latent-prediction graph states than from Alignment graph states when matching does not require that factor. In complete-system frozen-transfer experiments, GL-JEPA attains the highest mean scores on Node, Link, and Graph tasks among the compared methods, while GraphCLIP performs best on Retrieval. With the 1B Graph Encoder frozen, GL-JEPA also achieves competitive performance across nine graph question-answering tasks.

Our contributions are as follows.

- We identify two challenges in language supervision for text-attributed graphs: token reconstruction is tied to a particular linguistic realization, while direct graph–text alignment may suppress target-relevant features that are unnecessary for pair discrimination.
- We introduce **GL-JEPA** for text-attributed graphs. A frozen Text Encoder defines a continuous target from the paired description, while the Graph Encoder and Predictor predict this target from graph structure and node text. This formulation avoids token-level reconstruction and encourages retention of description-relevant features beyond those required for direct graph–text matching.
- We conduct controlled comparisons against autoregressive token prediction and direct graph–text alignment, together with a synthetic diagnostic for feature suppression. We further evaluate frozen

transfer and query-conditioned graph question answering to assess the quality and usability of the learned graph representations.

2 METHOD

2.1 PROBLEM SETUP

Let $\mathcal{D} = \{(G_i, Y_i)\}_{i=1}^{M}$ denote graphs paired with descriptions. Each graph $G_i = (V_i, E_i, \mathcal{X}_i)$ contains $N_i = |V_i|$ nodes, directed edges E_i, and node texts $\mathcal{X}_i = \{X_{in}\}_{n=1}^{N_i}$. The paired description Y_i provides supervision during pretraining and is not included in the graph input. A frozen Text Encoder E_T maps each node text to $\mathbf{x}_{in} = E_T(X_{in})$ and the paired description to a latent target $\mathbf{z}_i^Y = E_T(Y_i) \in \mathbb{R}^{d_T}$. The Graph Encoder E_G processes the graph structure and node features to produce contextualized node states $\mathbf{H}_i$ and a graph summary $\mathbf{g}_i$, which the Predictor P_ω uses to predict $\mathbf{z}_i^Y$. During paired-description pretraining, E_G and P_ω are updated while E_T remains fixed.

Graph Context Pretraining (Section 2.2) initializes E_G by predicting target representations of masked nodes from a masked graph. Graph–Language Latent Prediction (Section 2.3) then trains E_G and P_ω to predict the paired-description target $\mathbf{z}_i^Y$.

2.2 GRAPH CONTEXT PRETRAINING

Context and target views. For each graph G_i, we construct a masked context view $G_i^{\mathrm{ctx}} = \mathcal{M}(G_i)$ and an unmasked target view $G_i^{\mathrm{tgt}} = G_i$. Let $\mathcal{V}_i^{\mathrm{mask}} \subseteq V_i$ denote the sampled nodes whose input features are masked. The masking operator $\mathcal{M}$ preserves the node set, masks the features of nodes in $\mathcal{V}_i^{\mathrm{mask}}$, independently masks sampled feature coordinates, and drops sampled directed edges.

The context view is processed by the online Graph Encoder $E_G(\cdot;\theta)$. The target view is processed by an encoder initialized from the online Graph Encoder. Its parameters $\bar{\theta}$ are updated by exponential moving average (EMA) as $\bar{\theta} \leftarrow \tau\bar{\theta} + (1-\tau)\theta$, where τ is the EMA momentum.

The two encoders produce $\widetilde{\mathbf{H}}_i = E_G(G_i^{\mathrm{ctx}};\theta)$ and $\mathbf{H}_i^+ = \mathrm{sg}(E_G(G_i^{\mathrm{tgt}};\bar{\theta}))$. Here, $\widetilde{\mathbf{H}}_i = [\widetilde{\mathbf{h}}_{i1};\ldots;\widetilde{\mathbf{h}}_{iN_i}]$ and $\mathbf{H}_i^+ = [\mathbf{h}_{i1}^+;\ldots;\mathbf{h}_{iN_i}^+]$. The vectors $\widetilde{\mathbf{h}}_{in}$ and $\mathbf{h}_{in}^+$ are the representations of the same node n from the masked and unmasked views, respectively. The stop-gradient operator sg blocks gradients through the target encoder. Appendix D reports the node-masking rate, edge-drop rate, EMA schedule, and optimization settings.

Masked-node prediction and objective. The structural Predictor P_ψ is an MLP that predicts the target representation of each masked node as $\widehat{\mathbf{h}}_{in} = P_\psi(\widetilde{\mathbf{h}}_{in})$ for $n \in \mathcal{V}_i^{\mathrm{mask}}$. Unmasked nodes provide context to the Graph Encoder but do not contribute directly to the prediction loss.

For a minibatch $\mathcal{B}$, Graph Context Pretraining minimizes the normalized Smooth-L1 loss between the predicted and target representations of masked nodes as

$$\mathcal{L}_{\mathrm{GCP}} = \frac{1}{|\mathcal{B}|}\sum_{i\in\mathcal{B}}\frac{1}{|\mathcal{V}_i^{\mathrm{mask}}|}\sum_{n\in\mathcal{V}_i^{\mathrm{mask}}}\ell_{\mathrm{SL1}}\left(u_\varepsilon(\widehat{\mathbf{h}}_{in}), u_\varepsilon(\mathbf{h}_{in}^+)\right), \tag{1}$$

where $u_\varepsilon(\mathbf{v}) = \mathbf{v}/\max(\|\mathbf{v}\|_2, \varepsilon)$ with $\varepsilon = 10^{-8}$, and ℓ_{SL1} denotes the coordinate-mean Smooth-L1 loss. Gradients update the online Graph Encoder E_G and P_ψ. After Graph Context Pretraining, P_ψ and the target encoder are discarded, and the learned Graph Encoder initializes Graph–Language Latent Prediction.

2.3 GRAPH–LANGUAGE LATENT PREDICTION

Graph–Language Latent Prediction uses the paired description Y_i to supervise the Graph Encoder. The frozen Text Encoder E_T maps Y_i to the latent target $\mathbf{z}_i^Y$, while E_G produces contextualized node states and a graph summary from G_i. The Predictor P_ω predicts $\mathbf{z}_i^Y$ from these graph representations. Neither Y_i nor $\mathbf{z}_i^Y$ is provided to E_G or P_ω.

Text Encoder (E_T). The frozen Text Encoder E_T maps each node text X_{in} to $\mathbf{x}_{in} = E_T(X_{in})$ and the paired description Y_i to $\mathbf{z}_i^Y = E_T(Y_i) \in \mathbb{R}^{d_T}$. The same encoder and pooling rule are used for both inputs. The node features $\{\mathbf{x}_{in}\}_{n=1}^{N_i}$ are supplied to E_G, whereas $\mathbf{z}_i^Y$ is used only as the prediction target. E_T remains frozen throughout training.

Graph Encoder (E_G). The Graph Encoder takes the directed graph structure and node features $\{\mathbf{x}_{in}\}_{n=1}^{N_i}$ as input. We instantiate E_G with GraphGPS (Rampášek et al., 2022), using GraphSAGE (Hamilton et al., 2017) for local message passing and self-attention for global mixing. GraphSAGE aggregates information over directed edge neighborhoods, while self-attention exchanges information across nodes. The graph is not symmetrized and reverse edges are not added.

The Graph Encoder produces contextualized node states $\mathbf{H}_i = [\mathbf{h}_{i1}, \ldots, \mathbf{h}_{iN_i}]^\top \in \mathbb{R}^{N_i \times d_G}$. We compute the mean node state $\boldsymbol{\mu}_i = N_i^{-1} \sum_{n=1}^{N_i} \mathbf{h}_{in}$ and obtain the graph summary as

$$\mathbf{g}_i = W_g \boldsymbol{\mu}_i + \mathbf{b}_g \in \mathbb{R}^{d_G}. \tag{2}$$

The Predictor receives both $\mathbf{H}_i$ and $\mathbf{g}_i$. Layer counts, state dimensions, positional features, and attention settings are reported in Appendix D.

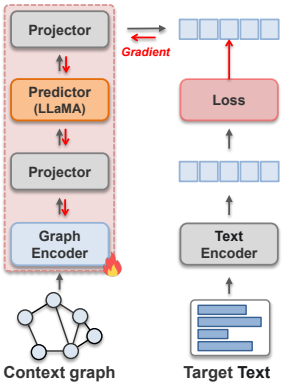


Figure 2: Graph–Language Latent Prediction. The frozen target $\mathbf{z}_i^Y$ is not provided to E_G or P_ω. Only the Graph Encoder and Predictor are trained.

Predictor (P_ω). The Predictor maps the graph summary and contextualized node states to a prediction $\widehat{\mathbf{z}}_i^Y$ of the latent target. The graph summary and node states are projected from dimension d_G to the Predictor dimension $d_P = 2048$. Let $a_G(\mathbf{v}) = W_G\mathbf{v} + \mathbf{b}_G$ and $a_N(\mathbf{v}) = W_N\mathbf{v} + \mathbf{b}_N$, where $W_G, W_N \in \mathbb{R}^{d_P \times d_G}$. Learned embeddings $\mathbf{e}_G, \mathbf{e}_N \in \mathbb{R}^{d_P}$ identify the graph-summary and node tokens. The Predictor input sequence is

$$\mathbf{C}_i = \left[a_G(\mathbf{g}_i) + \mathbf{e}_G; (a_N(\mathbf{h}_{in}) + \mathbf{e}_N)_{n=1}^{N_i} \right] \in \mathbb{R}^{(N_i+1) \times d_P}. \tag{3}$$

We initialize P_ω from the final eight Llama-3.2-1B Transformer blocks (Meta AI, 2024) and disable causal masking. The Transformer processes all valid graph-derived tokens bidirectionally. The Predictor receives node states already contextualized by graph structure and their order-independent mean summary. The Predictor assigns consecutive rotary positions to node tokens.

For a minibatch, let $L = 1 + \max_i N_i$, $\widetilde{\mathbf{C}}_i \in \mathbb{R}^{L \times d_P}$ be the padded Predictor input, and $m_{ij} \in \{0, 1\}$ indicate a valid position. Let $\mathbf{U}_i = \mathcal{T}_\omega(\widetilde{\mathbf{C}}_i; m_i)$ be the Transformer output and $\mathbf{u}_{ij} = \mathbf{U}_i[j]$ the output at position j. The Predictor averages the valid outputs and projects them to the Text Encoder dimension as

$$\widehat{\mathbf{z}}_i^Y = q_\omega \left(\frac{\sum_{j=0}^{L-1} m_{ij} \mathbf{u}_{ij}}{\sum_{j=0}^{L-1} m_{ij}} \right), \qquad q_\omega : \mathbb{R}^{d_P} \to \mathbb{R}^{d_T}. \tag{4}$$

Padding positions are excluded from attention and pooling. The input projections, type embeddings, Transformer blocks, and output projection are trainable parts of P_ω.

Training objective. For each pair (G_i, Y_i), the Predictor produces $\widehat{\mathbf{z}}_i^Y = P_\omega(\mathbf{H}_i, \mathbf{g}_i)$, while the frozen Text Encoder produces $\mathbf{z}_i^Y = E_T(Y_i)$. The predicted and target representations are compared only through the prediction loss. We normalize each representation as $u_\varepsilon(\mathbf{v}) = \mathbf{v} / \max(\|\mathbf{v}\|_2, \varepsilon)$, with $\varepsilon = 10^{-8}$, and minimize

$$\mathcal{L}_{\text{GL-JEPA}} = \frac{1}{|\mathcal{B}| d_T} \sum_{i \in \mathcal{B}} \left\| u_\varepsilon(\widehat{\mathbf{z}}_i^Y) - u_\varepsilon(\mathbf{z}_i^Y) \right\|_2^2. \tag{5}$$

The normalization removes representation magnitude from the loss, and $1/d_T$ averages the squared error over target coordinates. Gradients update E_G and P_ω, while E_T remains frozen. The objective uses neither token likelihood nor in-batch negatives.

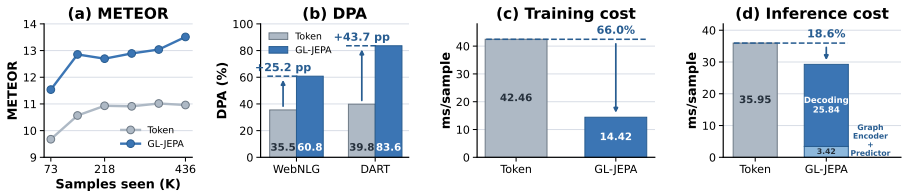


Figure 3: Token versus latent prediction. (a) WebNLG METEOR by pretraining samples (K). (b) Dual-positive accuracy on WebNLG factual negatives and DART mismatched descriptions. (c) Training-update latency, with GL-JEPA including separately measured generation-readout fitting and excluding decoding. (d) WebNLG inference latency, decomposed into graph encoding and prediction, and greedy decoding for GL-JEPA.

3 Experiments

Section 3.1 compares supervision interfaces. Section 3.2 analyzes language prediction and computational cost, while Section 3.3 contrasts Alignment with easy and hard negatives against full-target latent prediction. Sections 3.4 and 3.5 evaluate frozen transfer and graph question answering. Finally, Section 3.6 examines the design choices of GL-JEPA.

3.1 Controlled Comparison of Graph–Language Supervision Interfaces

Setup. Table 1 compares Token, Alignment, and GL-JEPA under a shared protocol without Graph Context Pretraining. All settings are shared across methods, while the language-side modules remain specific to each supervision interface. A Linear Predictor variant of GL-JEPA serves as a capacity control, with nearly the same number of trainable parameters as Alignment. Frozen-transfer evaluation covers node classification and link prediction on seven datasets each, molecular graph classification on four datasets, and full-pool graph–text retrieval on four datasets. Appendix E provides the complete training and evaluation protocols.

Results. Token performs worst across all four transfer metrics, consistent with its objective of reconstructing a particular linguistic realization rather than directly optimizing graph-task representations. Alignment achieves the highest Retrieval R@1, as it directly optimizes graph–text matching, but trails GL-JEPA on Node, Link, and Graph. This pattern is consistent with matching providing weaker supervision for graph features that are not needed for retrieval. The nearly parameter-matched Linear Predictor shows the same advantage over Alignment on graph tasks, indicating that the pattern is not explained by Predictor parameter count alone.

Table 1: Controlled comparison of graph–language supervision interfaces (%). Trainable parameters include the Graph Encoder.

Method	Trainable Params.(M)	Node (Acc.)	Link (AUC)	Retrieval (R@1)	Graph (AUC)
Token	1,031	25.62	67.37	42.52	39.45
Alignment	18.5	55.96	71.90	**75.07**	51.78
GL-JEPA	772.6	**59.55**	**75.72**	67.36	**54.33**
GL-JEPA (Linear Pred.)	18.7	58.40	73.94	65.32	53.02

3.2 Continuous Supervision for Language Prediction

Setup. Figure 3 evaluates language prediction and training cost under matched pretraining data and sample counts. Token and GL-JEPA use the same paired graph–text pretraining data. For METEOR, a separately trained generation adapter connects the frozen GL-JEPA checkpoint to a frozen decoder. This readout is not part of pretraining. Language prediction is evaluated on WebNLG with METEOR at matched numbers of processed pretraining samples. Dual-positive accuracy (DPA) measures whether two valid descriptions both outrank the same negative on WebNLG and DART under each model's native scoring rule. Training and inference cost are measured in time per sample. Appendix E provides the scoring, generation, and timing protocols.

Results. Because token prediction is trained against a single observed wording, it directly rewards reproducing that linguistic realization. GL-JEPA instead predicts a continuous description representation, allowing semantically valid alternatives to be recognized without matching the reference token sequence. GL-JEPA achieves higher WebNLG METEOR at matched pretraining sample counts and improves dual-positive accuracy by 25.23 points on WebNLG and 43.75 points on DART. It also reduces measured training-update cost by 66.0%, including generation-readout fitting, by avoiding token-level reconstruction during pretraining.

Table 2: Frozen transfer across graph tasks (%). Best values are bold. GL-JEPA is shaded blue. Values lower than GL-JEPA are gray.

Model	Node Classification Acc. (%)								Bidirectional Retrieval R@1 (%)				
	Cora	Cite.	Wiki	Insta.	Photo	Comp.	Hist.	Mean	Agenda	GenW.	TekGen	T-REx	Mean
GraphCLIP	62.94	61.07	**70.43**	50.46	52.38	**60.35**	38.92	56.65	73.89	81.33	**91.07**	55.91	**75.55**
ADAligner	57.06	59.03	63.27	51.04	54.92	40.68	38.05	52.01	**75.30**	74.21	81.45	51.29	70.56
G2P2	58.05	59.17	34.03	**59.36**	25.48	37.42	26.27	42.83	66.05	45.11	66.83	**60.79**	59.70
GL-JEPA	**67.36**	**67.43**	68.27	57.65	**56.44**	59.48	**48.26**	**60.70**	71.28	**81.34**	83.16	57.87	73.41
Model	**Link Prediction AUC (%)**								**Graph Classification AUC (%)**				
	Cora	Cite.	Wiki	Insta.	Photo	Comp.	Hist.	Mean	ClinTox	HIV	Tox21	ToxCast	Mean
GraphCLIP	72.36	70.95	84.86	**67.20**	60.28	66.95	78.53	71.59	44.46	54.81	63.28	**61.55**	56.03
ADAligner	55.88	55.55	55.20	50.97	51.09	51.21	53.15	53.29	46.72	54.49	60.86	54.45	54.13
G2P2	74.99	71.77	80.25	63.57	71.50	68.41	80.92	73.06	**51.66**	39.55	43.29	44.86	44.84
GL-JEPA	**77.35**	**75.05**	**85.18**	57.99	**73.90**	**76.97**	**82.63**	**75.58**	50.60	**57.19**	**65.25**	59.55	**58.15**

3.3 Feature Suppression Across Supervision Objectives

Setup. We test whether Alignment preserves information that is present in the target but unnecessary for distinguishing easy graph–text pairs (Robinson et al., 2021; Li et al., 2023). We use two diagnostics. In the synthetic setting, each graph contains two factors, A and B, encoded in node features. Pairs such as (A_1, B_3) and (A_7, B_5) can be distinguished using A alone, whereas (A_1, B_3) and (A_1, B_5) require B. In WikiBio, we construct two-node graphs from nationality and occupation. Easy candidates differ in nationality, while hard candidates share nationality and differ only in occupation. We compare easy-negative Alignment, hard-negative Alignment, and GL-JEPA latent prediction. After pretraining, each Graph Encoder is frozen and a linear probe measures how well B or occupation can be recovered from its graph states. Appendix C gives the full protocols.

Results. With easy negatives, the model can identify the matching description using only A in the synthetic setting or nationality in WikiBio. As a result, B and occupation contribute little to the matching objective and are nearly unrecoverable from the frozen graph states. When the easy cue is held fixed, the model must use B or occupation to distinguish the correct match, and probe accuracy rises sharply with the same encoder architecture. GL-JEPA does not require such factor-specific negatives: by predicting the full target, it retains substantially more information about B and occupation than easy-negative Alignment, although hard-negative Alignment remains higher.

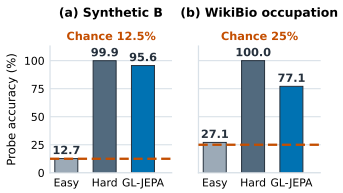


Figure 4: Linear-probe accuracy for B and occupation. GL-JEPA denotes latent prediction. Dashed lines mark chance.

3.4 Frozen Transfer Across Graph Tasks

Setup. Table 2 compares GL-JEPA with GraphCLIP (Zhu et al., 2025), ADAligner (Liu et al., 2026), and G2P2 (Wen & Fang, 2023) under a shared pretraining corpus and frozen-transfer evaluation protocol. All methods are retrained on the same graph–text data while retaining their original architectures and training objectives. Evaluation covers node classification and link prediction on seven datasets (Cora, CiteSeer, WikiCS, Instagram, Photo, Computers, and History), graph–text retrieval on four datasets (Agenda, GenWiki, TekGen, and T-REx), and graph classification on four molecular datasets (ClinTox, HIV, Tox21, and ToxCast). All tasks follow a shared CLIP-style protocol based on embedding compatibility with class descriptions, edge endpoints, or graph descriptions, without updating model parameters on the evaluation datasets. We report mean accuracy for Node, mean AUC for Link and Graph, and bidirectional R@1 for Retrieval. Appendix B provides the full pretraining and evaluation protocols.

Results. GL-JEPA achieves the highest mean scores on Node, Link, and Graph tasks, outperforming the strongest baselines by 4.05 percentage points in Node accuracy and 2.52 percentage points in Link AUC, while attaining a mean Graph AUC of 58.15%. Its advantage is particularly consistent on Link, where it leads on six of seven datasets, although the strongest method varies across individual

Table 3: Graph question answering accuracy (%) across benchmarks. GraphInstruct Mean averages the five GraphWiz tasks. GL-JEPA is shaded light blue. Values lower than GL-JEPA are gray.

Model	Backbone Size	GraphInstruct						Expla.	CLeGR	NLGraph	
		Bip.	Conn.	Cyc.	Ham.	Sub.	Mean		Facts	Conn.	Cyc.
GraphGPT	7B	**92.75**	70.75	71.25	60.50	80.50	75.15	55.05	85.61	49.87	54.97
LLaGA	7B	71.25	68.00	68.75	58.75	45.25	62.40	51.26	88.16	50.13	51.31
GraphWiz	7B	74.00	**87.00**	91.50	52.25	82.25	77.40	**77.98**	84.76	70.89	60.21
GraphSolver	8B	64.50	67.75	68.50	31.75	41.75	54.85	58.66	**95.61**	64.15	52.88
GOFA	7B	68.00	67.75	68.75	32.25	47.50	56.85	58.84	79.21	53.64	52.88
TEA-GLM	7B	33.25	61.75	55.50	28.75	59.75	47.80	59.57	88.07	40.70	36.13
G1-3B	3B	64.50	40.25	**93.50**	49.25	40.25	57.55	44.22	79.66	73.05	52.88
GL-JEPA	1B	73.75	86.75	89.75	**62.75**	**85.75**	**79.75**	61.01	93.58	**86.25**	**99.48**

Node and Graph datasets. In contrast, GraphCLIP achieves higher mean Retrieval R@1 (75.55% versus 73.41% for GL-JEPA), with dataset-level performance varying across methods.

3.5 Graph Question Answering

Setup. Table 3 evaluates graph question answering (GQA) following latent predictive pretraining without autoregressive text reconstruction. We compare GL-JEPA with GraphGPT (Tang et al., 2024), LLaGA (Chen et al., 2024b), GraphWiz (Chen et al., 2024a), GraphSolver (Luo et al., 2027), GOFA (Kong et al., 2025), TEA-GLM (Wang et al., 2024), and G1 (Guo et al., 2025). During GQA adaptation, GL-JEPA keeps the pretrained Graph Encoder E_G frozen while adapting the Predictor P_ω and its output projection. The Predictor conditions on the frozen graph representations and task query to score candidate answers, following the discriminative visual question answering protocol of VL-JEPA (Chen et al., 2026), rather than generating free-form text. Evaluation covers five GraphInstruct tasks—Bipartite, Connectivity, Cycle, Hamiltonian, and Substructure—with 400 test examples each, as well as ExplaGraphs, CLeGR-Facts, and the Connectivity and Cycle subsets of NLGraph. Appendix E provides the adaptation and scoring protocols.

Results. GL-JEPA achieves 79.75% mean accuracy across the five GraphInstruct tasks with E_G frozen, compared with the 77.40% reported in the GraphWiz paper. Among the results, GL-JEPA achieves the highest accuracy on Hamiltonian and Substructure and exceeds the strongest listed baselines on NLGraph Connectivity and Cycle by 13.20 and 39.27 percentage points, respectively. Other systems report higher scores on GraphInstruct Bipartite, Connectivity, and Cycle, as well as ExplaGraphs and CLeGR-Facts. These results demonstrate that GL-JEPA supports competitive query-conditioned graph question answering without updating the pretrained Graph Encoder.

Table 4: Ablation studies of GL-JEPA (%). GQA averages the five GraphInstruct tasks.

Configuration	Node	Link	Retrieval	GQA
(a) Pretraining components				
Full pipeline	60.70	75.58	73.41	79.75
w/o Graph Context Pretraining	58.90	75.22	68.72	77.75
Graph Context Pretraining only	43.14	63.42	2.06	61.70
w/o Pretraining	45.60	71.62	2.52	62.30
(b) Prediction objective				
MSE	60.70	75.58	73.41	79.75
Smooth-L1	60.92	65.10	64.03	76.35
InfoNCE	60.42	63.89	78.36	78.00
(c) Text Encoder variation				
all-MiniLM-L6-v2 (default)	60.70	75.58	73.41	79.75
EmbeddingGemma-300M	62.82	71.40	74.14	74.75
Qwen3-Embedding-0.6B	61.25	75.17	70.81	79.80
BGE-M3 (568M)	62.06	73.78	69.63	75.05

Configuration	Node	Link	Retrieval	GQA
(d) Predictor architecture				
Llama-initialized Transformer	60.70	75.58	73.41	79.75
Random Transformer	60.22	63.88	72.02	78.00
MLP	56.86	71.02	57.91	71.50
Linear	59.07	72.19	63.51	74.10
Attention + MLP	57.07	63.14	62.10	71.05
(e) Predictor input tokens				
Graph + node tokens	60.70	75.58	73.41	79.75
Graph token only	59.75	71.26	55.78	72.30
Node tokens only	60.22	68.04	62.14	80.45
(f) Graph Encoder variation				
GraphGPS (default)	60.70	75.58	73.41	79.75
GraphSAGE (4 layers)	61.12	77.01	66.63	79.25
GCN (4 layers)	60.94	77.32	68.25	78.65

3.6 Ablation of GL-JEPA Design Choices

Setup. We conduct six ablation studies to examine how GL-JEPA's pretraining stages, prediction objective, Text Encoder, Predictor architecture, Predictor input tokens, and Graph Encoder affect

transfer across graph prediction, graph–text retrieval, and graph question answering (Table 4). Within each ablation group, we hold all settings except for changes necessitated by removing a pretraining stage. Node accuracy, Link AUC, and Retrieval R@1 are evaluated with the Graph Encoder frozen, while GQA reports mean accuracy after adapting the Predictor with the Graph Encoder frozen. Appendices B and D detail the evaluation and training protocols.

(a) Pretraining components. The full pipeline outperforms the variant without Graph Context Pretraining across all four metrics, showing the benefit of structural pretraining before description-target prediction.

(b) Prediction objective. InfoNCE improves Retrieval but not graph prediction or question answering. Node accuracy changes little, while Link AUC drops substantially.

(c) Text Encoder variation. Different Text Encoders yield task-dependent results, showing that the continuous description target affects transfer beyond encoder size.

(d) Predictor architecture. Random initialization mainly affects Link performance, whereas replacing the Transformer degrades transfer across more tasks. Predictor architecture contributes beyond initialization.

(e) Predictor input tokens. Graph-summary and graph-node tokens are complementary for frozen transfer. After query-conditioned adaptation, node tokens alone are sufficient for GQA, and additional tokens do not improve performance.

(f) Graph Encoder variation. GraphSAGE and GCN variants (Kipf & Welling, 2017) perform well on local graph prediction but worse on graph–text retrieval, showing that Graph Encoder architecture affects transfer under the same continuous description target.

3.7 Case Study

Interpretation. Figure 5 shows a held-out graph encoding that Anatole de Grunwald produced *English Without Tears* and that the film runs for 89 minutes. A and B express these same facts with different wording, while C changes the producer in B's sentence. Token ranks them $A_{\text{valid}} > C_{\text{false}} > B_{\text{valid}}$, placing the false description above the valid paraphrase B. GL-JEPA ranks them $A_{\text{valid}} > B_{\text{valid}} > C_{\text{false}}$, placing both valid descriptions above the false one. GL-JEPA preserves the factual ordering across paraphrases, while Token does not in this case.

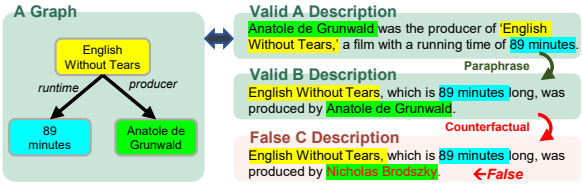


Figure 5: Case study of dual-positive accuracy on a held-out WebNLG example.

4 Related Work

Graph–Language Models. Autoregressive graph–language methods condition language decoders on graph representations and optimize token-level generation losses (Tang et al., 2024; Chen et al., 2024b). Graph context is incorporated through learned tokens, serialized neighborhoods, or retrieved subgraphs (He et al., 2024). Approaches employ instruction tuning for graph question answering and multi-task understanding (Chen et al., 2024a; Luo et al., 2027), extend generation across graph task levels (Kong et al., 2025), align GNN representations with language-model token embeddings (Wang et al., 2024), or use reinforcement learning for graph reasoning (Guo et al., 2025). GL-JEPA predicts a continuous description representation during pretraining and trains a decoder only for generation.

Graph–Text Alignment. Graph–text alignment learns compatibility between matched and mismatched graph–description pairs (Wen & Fang, 2023; Brannon et al., 2024), enabling the transfer of textual supervision to graph representations across datasets and tasks (Zhu et al., 2025; Fang et al., 2025; Liu et al., 2026; Wang et al., 2026). A contrastive objective can be satisfied by features sufficient for pair discrimination even when other target-relevant features remain weakly represented (Robinson et al., 2021; Li et al., 2023). Prior work has explored latent target decoding to mitigate predictive feature suppression in image–caption retrieval (Bleeker et al., 2023). GL-JEPA predicts a

frozen description representation from graph context rather than directly aligning pooled graph and text embeddings.

Joint-Embedding Predictive Architectures. Joint-embedding predictive methods estimate target representations from observed context rather than reconstructing inputs (Grill et al., 2020; Baevski et al., 2022). Image methods predict representations of missing regions without hand-designed augmentation (Assran et al., 2023), while video methods extend latent prediction across time for understanding and planning (Bardes et al., 2024; Assran et al., 2025). Related objectives have been applied to audio, point clouds, and tables (Fei et al., 2023; Saito et al., 2025; Thimonier et al., 2025), as well as language (Huang et al., 2026) and multimodal alignment (Lei et al., 2025). Graph-level work predicts masked-subgraph representations (Skenderi et al., 2025), while vision–language work predicts text embeddings from visual context (Chen et al., 2026). GL-JEPA predicts a frozen description target from graph structure and node text. We compare its graph-task transfer with autoregressive token prediction and direct graph–text alignment.

5 Discussion

Compared with autoregressive token prediction, GL-JEPA replaces token-level reconstruction with prediction of a continuous description target. Token is trained to reproduce the particular wording of the paired description, whereas latent prediction targets its frozen representation. In the controlled comparison and case study, latent prediction more often ranks differently worded valid descriptions above an incorrect one. These results suggest less dependence on the wording of the training description.

Compared with direct graph–text Alignment, latent prediction targets the full description representation rather than only distinguishing matched from mismatched pairs. In the synthetic and WikiBio diagnostics, target-relevant factors that matching need not use are more recoverable from the frozen graph states of the latent-prediction model than from those of Alignment, indicating less feature suppression in these controlled settings. Separately, Alignment's higher Retrieval R@1 does not translate into better Node, Link, or Graph transfer in the controlled comparison.

These findings point to why predicting a frozen description target is useful. The Graph Encoder and Predictor learn to estimate the Text Encoder's representation of the paired description, so pretraining does not reconstruct the observed wording and can supervise factors encoded in that representation even when pair matching does not require them.

Limitations and Future Work. GL-JEPA predicts a single frozen description representation per training example, limiting language supervision to the information encoded in that target. Future work could predict multiple description representations to capture more semantic information. Multi-step latent prediction could combine intermediate Predictor outputs with graph states to refine subsequent predictions before task-specific readout.

6 Conclusion

We introduced GL-JEPA, a graph–language pretraining framework that predicts frozen description representations from graph states without autoregressive token reconstruction or direct graph–text alignment. Compared with Token, GL-JEPA ranks two valid descriptions above a negative without reconstructing their wording. In controlled comparisons, GL-JEPA outperforms Alignment on Node, Link, and Graph tasks, whereas Alignment achieves higher Retrieval R@1. In the controlled feature-suppression diagnostic, a target-relevant factor is more recoverable from latent-prediction graph states than from Alignment graph states when matching does not require that factor. With the Graph Encoder frozen, the learned representations also support text generation through a separate readout and query-conditioned graph question answering through Predictor adaptation.

AI Use Statement

We used generative AI tools, including OpenAI Codex, to assist with experimental-design feedback, implementation and debugging of model and analysis code, construction of figures and tables, and

interpretation of results. These tools also assisted in implementing the procedural generator for the synthetic feature-suppression diagnostic, which samples graph factors A, B, and S under the protocol in Appendix C. They further assisted with literature discovery, citation checks, translation, and drafting and editing parts of the manuscript. The authors decided which suggestions and experiments to adopt, reviewed the AI-assisted outputs, and take responsibility for the final methods, results, claims, and text.

Ethics Statement

This work uses previously released graph datasets, graph–description pairs, and synthetic graphs for representation learning and evaluation. We do not collect new human-subject data. Some pretraining data, including social-network graphs, may contain user-generated text and may inherit biases or other artifacts from the original sources and their description-generation pipelines. GL-JEPA may learn or propagate such biases through its graph and language representations. The experiments focus on representation learning and benchmark evaluation rather than deployment in high-stakes settings. We do not use the learned representations to infer sensitive attributes or to identify individuals.

Reproducibility Statement

The manuscript specifies the data, models, and procedures used for every reported evaluation. Appendix A identifies the five graph–description corpora, their graph counts, and the provenance of the released descriptions. Appendix B fixes the downstream datasets, stored splits, retrieval candidate pools, seeds, and summary metrics. Appendix C gives the complete construction and probe protocol for the synthetic diagnostic. The model configuration, two-stage pretraining schedule, optimization settings, and ablation controls are specified in Appendix D, while Appendix E documents the task-specific readouts, adaptation, and scoring procedures. Training and evaluation code, data manifests, and checkpoint identifiers will accompany the released artifact.

References

Mahmoud Assran, Quentin Duval, Ishan Misra, Piotr Bojanowski, Pascal Vincent, Michael Rabbat, Yann LeCun, and Nicolas Ballas. Self-supervised learning from images with a joint-embedding predictive architecture. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 15619–15629, June 2023. doi: 10.1109/CVPR52729.2023.01499. URL https://openaccess.thecvf.com/content/CVPR2023/html/Assran_Self-Supervised_Learning_From_Images_With_a_Joint-Embedding_Predictive_Architecture_CVPR_2023_paper.html.

Mido Assran, Adrien Bardes, David Fan, Quentin Garrido, Russell Howes, Mojtaba Komeili, Matthew Muckley, Ammar Rizvi, Claire Roberts, Koustuv Sinha, Artem Zholus, Sergio Arnaud, Abha Gejji, Ada Martin, Francois Robert Hogan, Daniel Dugas, Piotr Bojanowski, Vasil Khalidov, Patrick Labatut, Francisco Massa, Marc Szafraniec, Kapil Krishnakumar, Yong Li, Xiaodong Ma, Sarath Chandar, Franziska Meier, Yann LeCun, Michael Rabbat, and Nicolas Ballas. V-JEPA 2: Self-supervised video models enable understanding, prediction and planning, 2025. URL https://arxiv.org/abs/2506.09985.

Alexei Baevski, Wei-Ning Hsu, Qiantong Xu, Arun Babu, Jiatao Gu, and Michael Auli. data2vec: A general framework for self-supervised learning in speech, vision and language. In *Proceedings of the 39th International Conference on Machine Learning*, volume 162 of *Proceedings of Machine Learning Research*, pp. 1298–1312. PMLR, 2022. URL https://proceedings.mlr.press/v162/baevski22a.html.

Adrien Bardes, Quentin Garrido, Jean Ponce, Xinlei Chen, Michael Rabbat, Yann LeCun, Mido Assran, and Nicolas Ballas. Revisiting feature prediction for learning visual representations from video. *Transactions on Machine Learning Research*, 2024. ISSN 2835-8856. URL https://mlanthology.org/tmlr/2024/bardes2024tmlr-revisiting/.

Maurits Bleeker, Andrew Yates, and Maarten de Rijke. Reducing predictive feature suppression in resource-constrained contrastive image-caption retrieval. *Transactions on Machine Learning Research*, April 2023. URL https://openreview.net/forum?id=T1XtOqrVKn.

William Brannon, Wonjune Kang, Suyash Fulay, Hang Jiang, Brandon Roy, Deb Roy, and Jad Kabbara. ConGraT: Self-supervised contrastive pretraining for joint graph and text embeddings. In *Proceedings of TextGraphs-17: Graph-based Methods for Natural Language Processing*, pp. 19–39, Bangkok, Thailand, August 2024. Association for Computational Linguistics. URL https://aclanthology.org/2024.textgraphs-1.2/.

Delong Chen, Mustafa Shukor, Théo Moutakanni, Willy Chung, Lei Yu, Tejaswi Kasarla, Allen Bolourchi, Yann LeCun, and Pascale Fung. VL-JEPA: Joint embedding predictive architecture for vision-language. In *International Conference on Learning Representations*, 2026. URL https://mlanthology.org/iclr/2026/chen2026iclr-vljepa/.

Nuo Chen, Yuhan Li, Jianheng Tang, and Jia Li. GraphWiz: An instruction-following language model for graph computational problems. In *Proceedings of the 30th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pp. 353–364. Association for Computing Machinery, 2024a. doi: 10.1145/3637528.3672010. URL https://doi.org/10.1145/3637528.3672010.

Runjin Chen, Tong Zhao, Ajay Kumar Jaiswal, Neil Shah, and Zhangyang Wang. LLaGA: Large language and graph assistant. In *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pp. 7809–7823. PMLR, 2024b. URL https://proceedings.mlr.press/v235/chen24bh.html.

Yi Fang, Dongzhe Fan, Sirui Ding, Ninghao Liu, and Qiaoyu Tan. UniGLM: Training one unified language model for text-attributed graphs embedding. In *Proceedings of the Eighteenth ACM International Conference on Web Search and Data Mining*, pp. 973–981. Association for Computing Machinery, 2025. doi: 10.1145/3701551.3703586. URL https://doi.org/10.1145/3701551.3703586.

Zhengcong Fei, Mingyuan Fan, and Junshi Huang. A-JEPA: Joint-embedding predictive architecture can listen, 2023. URL https://arxiv.org/abs/2311.15830.

Jean-Bastien Grill, Florian Strub, Florent Altché, Corentin Tallec, Pierre H. Richemond, Elena Buchatskaya, Carl Doersch, Bernardo Avila Pires, Zhaohan Daniel Guo, Mohammad Gheshlaghi Azar, Bilal Piot, Koray Kavukcuoglu, Rémi Munos, and Michal Valko. Bootstrap your own latent: A new approach to self-supervised learning. In *Advances in Neural Information Processing Systems*, volume 33, pp. 21271–21284, 2020. URL https://proceedings.neurips.cc/paper/2020/hash/f3ada80d5c4ee70142b17b8192b2958e-Abstract.html.

Xiaojun Guo, Ang Li, Yifei Wang, Stefanie Jegelka, and Yisen Wang. G1: Teaching LLMs to reason on graphs with reinforcement learning. In *Advances in Neural Information Processing Systems*, volume 38, pp. 58961–59004, 2025. doi: 10.52202/085713-1767. URL https://proceedings.neurips.cc/paper_files/paper/2025/hash/4cd171bc72cdfec310eb6e9e78f7bd90-Abstract-Conference.html.

William L. Hamilton, Rex Ying, and Jure Leskovec. Inductive representation learning on large graphs. In *Advances in Neural Information Processing Systems*, volume 30, pp. 1024–1034, 2017. URL https://proceedings.neurips.cc/paper/2017/hash/5dd9db5e033da9c6fb5ba83c7a7ebea9-Abstract.html.

Xiaoxin He, Yijun Tian, Yifei Sun, Nitesh V. Chawla, Thomas Laurent, Yann LeCun, Xavier Bresson, and Bryan Hooi. G-Retriever: Retrieval-augmented generation for textual graph understanding and question answering. In *Advances in Neural Information Processing Systems*, volume 37, pp. 132876–132907, 2024. doi: 10.52202/079017-4224. URL https://proceedings.neurips.cc/paper_files/paper/2024/hash/efaf1c9726648c8ba363a5c927440529-Abstract.html.

Hai Huang, Yann LeCun, and Randall Balestriero. LLM-JEPA: Large language models meet joint embedding predictive architectures. In *International Conference on Learning Representations*,

2026. URL https://proceedings.iclr.cc/paper_files/paper/2026/hash/ac818ec2da1c976b267de32f59709c69-Abstract-Conference.html.

Bowen Jin, Gang Liu, Chi Han, Meng Jiang, Heng Ji, and Jiawei Han. Large language models on graphs: A comprehensive survey. *IEEE Transactions on Knowledge and Data Engineering*, 36(12): 8622–8642, 2024. doi: 10.1109/TKDE.2024.3469578. URL https://doi.org/10.1109/TKDE.2024.3469578.

Thomas N. Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. In *International Conference on Learning Representations*, 2017. URL https://openreview.net/forum?id=SJU4ayYgl.

Lecheng Kong, Jiarui Feng, Hao Liu, Chengsong Huang, Jiaxin Huang, Yixin Chen, and Muhan Zhang. GOFA: A generative one-for-all model for joint graph language modeling. In *International Conference on Learning Representations*, 2025. URL https://proceedings.iclr.cc/paper_files/paper/2025/hash/652c104b5b0652a03684efeaf805463b-Abstract-Conference.html.

Hongyang Lei, Xiaolong Cheng, Qi Qin, Dan Wang, Huazhen Huang, Qingqing Gu, Yetao Wu, and Luo Ji. M3-JEPA: Multimodal alignment via multi-gate MoE based on the joint-embedding predictive architecture. In *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, pp. 33902–33917. PMLR, 2025. URL https://proceedings.mlr.press/v267/lei25b.html.

Tianhong Li, Lijie Fan, Yuan Yuan, Hao He, Yonglong Tian, Rogerio Feris, Piotr Indyk, and Dina Katabi. Addressing feature suppression in unsupervised visual representations. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, pp. 1411–1420, January 2023. doi: 10.1109/WACV56688.2023.00146. URL https://openaccess.thecvf.com/content/WACV2023/html/Li_Addressing_Feature_Suppression_in_Unsupervised_Visual_Representations_WACV_2023_paper.html.

Yuhang Liu, Minglai Shao, Zengyi Wo, Yunlong Chu, Bing Hao, Shengzhong Liu, Ruijie Wang, and Jianxin Li. Learning noise-resilient and transferable graph-text alignment via dynamic quality assessment. In *Proceedings of the 49th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pp. 1209–1220. Association for Computing Machinery, 2026. doi: 10.1145/3805712.3809579. URL https://doi.org/10.1145/3805712.3809579.

Zihan Luo, Xiran Song, Hong Huang, Jianxun Lian, Chenhao Zhang, Jinqi Jiang, Xing Xie, and Hai Jin. GraphInstruct: Empowering large language models with graph understanding and reasoning capability. *Frontiers of Computer Science*, 21(1):2101302, 2027. doi: 10.1007/s11704-025-51382-0. URL https://journal.hep.com.cn/fcs/EN/10.1007/s11704-025-51382-0.

Meta AI. Llama 3.2: Revolutionizing edge AI and vision with open, customizable models. Meta AI Blog, September 2024. URL https://ai.meta.com/blog/llama-3-2-connect-2024-vision-edge-mobile-devices/.

Moritz Plenz and Anette Frank. Graph language models. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 4477–4494, Bangkok, Thailand, August 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.acl-long.245. URL https://aclanthology.org/2024.acl-long.245/.

Ladislav Rampášek, Mikhail Galkin, Vijay Prakash Dwivedi, Anh Tuan Luu, Guy Wolf, and Dominique Beaini. Recipe for a general, powerful, scalable graph transformer. In *Advances in Neural Information Processing Systems*, volume 35, pp. 14501–14515, 2022. doi: 10.52202/068431-1054. URL https://proceedings.neurips.cc/paper_files/paper/2022/hash/5d4834a159f1547b267a05a4e2b7cf5e-Abstract-Conference.html.

Nils Reimers and Iryna Gurevych. Sentence-BERT: Sentence embeddings using siamese BERT-networks. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pp. 3982–3992. Association for Computational Linguistics, November 2019. doi: 10.18653/v1/D19-1410. URL https://aclanthology.org/D19-1410/.

Joshua Robinson, Li Sun, Ke Yu, Kayhan Batmanghelich, Stefanie Jegelka, and Suvrit Sra. Can contrastive learning avoid shortcut solutions? In *Advances in Neural Information Processing Systems*, volume 34, pp. 4974–4986, 2021. URL https://arxiv.org/abs/2106.11230.

Ayumu Saito, Prachi Kudeshia, and Jiju Poovvancheri. Point-JEPA: A joint embedding predictive architecture for self-supervised learning on point cloud. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, pp. 7348–7357, February 2025. doi: 10.1109/WACV61041.2025.00714. URL https://openaccess.thecvf.com/content/WACV2025/html/Saito_Point-JEPA_A_Joint_Embedding_Predictive_Architecture_for_Self-Supervised_Learning_on_WACV_2025_paper.html.

Geri Skenderi, Hang Li, Jiliang Tang, and Marco Cristani. Graph-level representation learning with joint-embedding predictive architectures. *Transactions on Machine Learning Research*, 2025. URL https://mlanthology.org/tmlr/2025/skenderi2025tmlr-graphlevel/.

Jiabin Tang, Yuhao Yang, Wei Wei, Lei Shi, Lixin Su, Suqi Cheng, Dawei Yin, and Chao Huang. GraphGPT: Graph instruction tuning for large language models. In *Proceedings of the 47th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pp. 491–500. Association for Computing Machinery, 2024. doi: 10.1145/3626772.3657775. URL https://doi.org/10.1145/3626772.3657775.

Hugo Thimonier, José Lucas De Melo Costa, Fabrice Popineau, Arpad Rimmel, and Bich-Liên Doan. T-JEPA: Augmentation-free self-supervised learning for tabular data. In *International Conference on Learning Representations*, 2025. URL https://proceedings.iclr.cc/paper_files/paper/2025/hash/93f01c8d9b355d7bbe3f353b44ccde66-Abstract-Conference.html.

Duo Wang, Yuan Zuo, Fengzhi Li, and Junjie Wu. LLMs as zero-shot graph learners: Alignment of GNN representations with LLM token embeddings. In *Advances in Neural Information Processing Systems*, volume 37, pp. 5950–5973, 2024. doi: 10.52202/079017-0193. URL https://proceedings.neurips.cc/paper_files/paper/2024/hash/0b77d3a82b59e9d9899370b378087faf-Abstract-Conference.html.

Yuhan Wang, Haopeng Zhang, Yibo Ding, Jiaqi Yu, Xinyu Zhao, Yuhang Liu, Ziwei Zhang, Xiao Wang, and Ruijie Wang. S2Aligner: Pair-efficient and transferable pre-training for sparse text-attributed graphs, 2026. URL https://arxiv.org/abs/2605.18579.

Zhihao Wen and Yuan Fang. Augmenting low-resource text classification with graph-grounded pre-training and prompting. In *Proceedings of the 46th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pp. 506–516. Association for Computing Machinery, 2023. doi: 10.1145/3539618.3591641. URL https://doi.org/10.1145/3539618.3591641.

Yun Zhu, Haizhou Shi, Xiaotang Wang, Yongchao Liu, Yaoke Wang, Boci Peng, Chuntao Hong, and Siliang Tang. GraphCLIP: Enhancing transferability in graph foundation models for text-attributed graphs. In *Proceedings of the ACM on Web Conference 2025*, pp. 2183–2197. Association for Computing Machinery, 2025. doi: 10.1145/3696410.3714801. URL https://doi.org/10.1145/3696410.3714801.

A Datasets and Evaluation Coverage

Table 5: Graph counts in the GraphCLIP pretraining corpora used by GL-JEPA.

Corpus	Graphs
PubMed	19,717
OGBN-Arxiv	169,343
ArXiv-2023	46,198
Reddit	33,434
OGBN-Products	196,615

The graph–description pairs are drawn from the GraphCLIP collection (Zhu et al., 2025). We use its released target strings verbatim, including the Qwen-based generation and filtering pipeline. Table 5 reports the number of graphs in each pretraining corpus. Downstream datasets, task families, and metrics are summarized in Appendix B.

The corpora span three graph domains. PubMed, OGBN-Arxiv, and ArXiv-2023 provide scientific citation graphs. Reddit provides social interaction graphs, and OGBN-Products provides product co-purchase graphs. This mixture exposes the Graph Encoder to different node vocabularies, neighborhood structures, and relation semantics before downstream evaluation. Each stored training example consists of a text-attributed graph and its released paired description. Node text is encoded as graph input, whereas the paired description is used only as the target during Graph–Language Latent Prediction.

The counts in Table 5 describe the graph examples available to pretraining, rather than the sizes of the original source networks. Graph sampling retains directed edge information. The controlled Token, Alignment, and GL-JEPA comparison uses the same materialized graph–description examples and varies the language-supervision interface.

B Shared Evaluation Contract

Table 6: Task families and summary metrics used in the main evaluation.

Task family	Datasets or benchmarks	Reported metric
Node classification	Cora, CiteSeer, WikiCS, Instagram, Photo, Computers, History	Mean accuracy
Link prediction	Cora, CiteSeer, WikiCS, Instagram, Photo, Computers, History	Mean AUC
Graph classification	ClinTox, HIV, Tox21, ToxCast	Mean AUC
Graph–text retrieval	Agenda, GenWiki, TekGen, T-REx	Bidirectional R@1
Graph question answering	GraphInstruct (GraphWiz), ExplaGraphs, CLeGR-Facts, NLGraph	Accuracy

Table 1 compares supervision interfaces without Graph Context Pretraining, Table 2 compares complete pretrained systems, and Table 4 evaluates GL-JEPA configurations. Retrieval uses a fixed checkpoint, example order, and candidate pool. The **Full pipeline** row in Table 4 uses the Node, Link, and Retrieval means from Table 2 and the GraphInstruct Mean from Table 3.

For Table 1, each method is pretrained with seeds 90, 91, and 92. Node, Link, and Graph use five evaluation seeds (0–4) for each checkpoint. Retrieval uses fixed candidate pools. The table averages the three checkpoint scores.

For the complete GL-JEPA systems in Tables 2 and 3, Graph Context Pretraining uses seed 88 and Graph–Language Latent Prediction uses seed 90. Table 4 uses the same stage-specific seeds where a stage is run. Query-conditioned adaptation uses seed 90.

Node classification. Node classification uses common cached graph views and fixed class descriptions. Accuracy is averaged without weighting across the seven datasets.

Link prediction. Link prediction uses the same seven node datasets and five stored scoring splits. Each split uses the same positive edges and sampled non-edges for every method. Cosine similarity between endpoint representations supplies the link score. AUC is averaged without weighting across datasets.

Graph–text retrieval. Retrieval uses fixed full candidate pools for Agenda (1,000 examples), GenWiki (1,000), TekGen (2,000), and T-REx (2,000). Each graph ranks every description in its dataset, and each description ranks every graph. For dataset d,

$$R_d = \frac{R_{G\to Y,d} + R_{Y\to G,d}}{2},$$

and the reported mean is $\frac{1}{4}\sum_d R_d$.

Graph classification. Graph classification uses ClinTox, HIV, Tox21, and ToxCast with 80/10/10 splits for evaluation seeds 0–4 and positive/negative class-description pairs. The score is the difference between positive and negative class compatibility. AUC is averaged over valid binary targets containing both classes and then without weighting across the four datasets.

Graph question answering. Graph question answering is evaluated after query-conditioned adaptation with the Graph Encoder frozen. The GraphInstruct mean averages the accuracy of the Bipartite, Connectivity, Cycle, Hamiltonian, and Substructure tasks introduced with GraphWiz, with 400 held-out examples per task. The adapted checkpoint is selected using validation accuracy.

C Controlled Feature-Suppression Diagnostic

Data construction. Each generated example independently samples an easy cue $A \in \{1,\dots,128\}$, a semantic factor $B \in \{1,\dots,8\}$, and a structural factor $S \in \{\text{chain}, \text{star}\}$. Fixed normalized codebooks assign 16-dimensional vectors $\mathbf{c}_A$ and $\mathbf{b}_B$ to the two categorical factors. Each graph has eight nodes. Node 0 receives $5\mathbf{c}_A$, while nodes 1–7 receive $0.5\mathbf{b}_B$. The structural factor selects a bidirectional chain or star adjacency with self-loops. The 138-dimensional continuous target is

$$\mathbf{z} = u([\text{onehot}_{128}(A) \,\|\, \text{onehot}_8(B) \,\|\, \text{onehot}_2(S)])\,.$$

The target contains A, B, and S, while the input scaling makes A the easier cue.

Models and objectives. The synthetic Graph Encoder projects 16-dimensional node features to 48 dimensions, applies two message-passing layers, concatenates the mean and node-0 states, and maps the result to a two-dimensional graph state. A two-layer MLP maps this state through a 64-dimensional hidden layer to the 138-dimensional target space. All three conditions contain 19,580 trainable parameters. Alignment with easy negatives applies symmetric InfoNCE with temperature 0.07 to random batches, so negatives usually differ in both A and B. For example, (A_1, B_3) may be contrasted with (A_7, B_5), where the change in A is sufficient to distinguish the pair. Alignment with hard negatives holds A and S fixed while B varies, as in (A_1, B_3) versus (A_1, B_5). Multi-positive InfoNCE treats examples sharing B as positives rather than false negatives. Full-target latent prediction uses normalized MSE and no negatives. Each condition uses AdamW with learning rate 3×10^{-4}, weight decay 10^{-5}, batch size 256, 1,500 updates, and pretraining seeds 0–2.

Evaluation. After pretraining, the Graph Encoder is frozen. An ℓ_2-regularized logistic-regression probe ($C = 1$) is fitted on 8,192 generated examples and evaluated on 4,096 disjoint examples. Eight-way chance accuracy is 12.5%. For the counterfactual sensitivity check, we replace B with $(B+1) \bmod 8$ while holding A and S fixed, and report $1 - \cos(\mathbf{g}(A,B,S), \mathbf{g}(A,B',S))$.

Table 7: Per-seed factor-B recovery and counterfactual sensitivity in the controlled diagnostic. Probe values are accuracies (%). Sensitivity is the mean cosine distance after changing only B.

Condition	Seed 0	Seed 1	Seed 2	Mean	Sensitivity
Alignment with easy negatives	12.26	13.06	12.74	12.69	0.001
Alignment with hard negatives	100.00	99.83	100.00	99.94	0.835
Full-target latent prediction	87.45	100.00	99.41	95.62	0.535

The results separate two routes to preserving B. Under easy-negative matching, the candidate set lets the model distinguish pairs using the stronger cue A, so B remains poorly decodable. In the hard-negative condition, holding A and S fixed while varying B, together with treating examples that share B as positives, makes B relevant to the matching decision. Full-target prediction instead includes B directly in every target and yields high probe accuracy without factor-specific negative construction. The diagnostic therefore shows how both target content and candidate composition shape which factors remain decodable.

WikiBio two-node diagnostic. We use only WikiBio's official training split and leave its validation and test splits untouched. We select examples with one-token nationality and occupation fields from the four nationalities (American, Indian, British, and French) and four occupations (politician, actor, actress, and writer). The first sentence must state the nationality and mention the occupation; examples containing dates or another selected nationality or occupation are excluded, and exact delexicalized sentences are deduplicated. For each of the 16 nationality–occupation cells, seven examples are used for training and two for testing, giving 112 training and 32 held-out examples. The cell-wise split uses seed 20260925.

We construct a two-node connected graph from each selected WikiBio example. The node features are frozen all-MiniLM-L6-v2 embeddings of the nationality and occupation labels, and the target is the same encoder's frozen embedding of the original first-sentence predicate after removing the biography subject. Easy Alignment samples four candidates with distinct nationalities, so occupation is unnecessary for identifying the matching description. Hard Alignment fixes nationality and uses four distinct occupations. Full-target latent prediction uses four sampled graph–description pairs without negatives. All three conditions use a two-dimensional Graph Encoder representation and the same prediction head, batch size 4, 1,500 updates, AdamW learning rate 3×10^{-4}, and weight decay 10^{-5}; the Alignment temperature is 0.07. We train with seeds 0, 1, and 2. After training, we freeze the Graph Encoder and fit a four-way occupation probe using standardized features and logistic regression ($C = 1$) on the 112 training examples. Accuracy is measured on the 32 held-out examples; chance accuracy is 25%. This auxiliary experiment uses a compact two-node model and does not evaluate the complete GL-JEPA architecture.

D Model, Pretraining, and Optimization Details

Table 8: Default model configuration. The same frozen Text Encoder E_T embeds node text and paired descriptions.

Component	Configuration
Text Encoder (E_T)	all-MiniLM-L6-v2 (Reimers & Gurevych, 2019), mean pooling, $d_T = 384$, frozen
Graph Encoder (E_G)	12 GraphGPS layers, GraphSAGE local operator, $d_G = 384$
Positional encoding	32-step random-walk features projected to 8 dimensions
Predictor (P_ω)	final 8 Llama-3.2-1B blocks (Meta AI, 2024), layers 8–15, $d_P = 2048$
Predictor input	graph summary followed by contextualized node states
Node-token order	ascending original node IDs within each sampled graph

The graph readout is the affine map

$$\mathbf{g}_i = W_g \boldsymbol{\mu}_i + \mathbf{b}_g, \qquad W_g \in \mathbb{R}^{d_G \times d_G}, \quad \mathbf{b}_g \in \mathbb{R}^{d_G}, \tag{6}$$

where $\boldsymbol{\mu}_i = N_i^{-1} \sum_n \mathbf{h}_{in}$. This graph summary is used by Graph–Language Latent Prediction.

Graph Context Pretraining. The context view masks node features with probability 0.35, independently masks feature coordinates with probability 0.10, and drops directed edges with probability 0.25. The target branch receives the unmasked graph. The two-layer structural Predictor maps the contextualized state of each masked

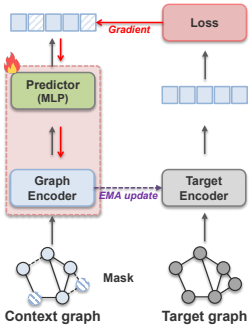


Figure 6: Graph Context Pretraining.

node to its target state. For

$$h_\beta(\delta) = \begin{cases} \delta^2/(2\beta), & |\delta| < \beta, \\ |\delta| - \beta/2, & \text{otherwise}, \end{cases} \qquad \beta = 1,$$

the Smooth-L1 loss is $\ell_{\mathrm{SL1}}(\mathbf{a}, \mathbf{b}) = d_G^{-1} \sum_{k=1}^{d_G} h_\beta(a_k - b_k)$. At update t, the EMA momentum is

$$\tau_t = 0.996 + (1.0 - 0.996)\frac{\min(t, 18180)}{18180}, \qquad \bar{\theta} \leftarrow \tau_t \bar{\theta} + (1 - \tau_t)\theta. \tag{7}$$

The transferred checkpoint is taken after 5,454 updates, for which $\tau_t = 0.9972$.

Table 9: Optimization and capacity of the controlled interfaces without Graph Context Pretraining. Parameter counts are for trainable tensors stored in the evaluated checkpoints.

Interface	Updates	Batch	LR	Weight decay	Params (M)
Token	18,180	512	10^{-5}	10^{-5}	1,031.2
Alignment	18,180	512	10^{-5}	10^{-5}	18.5
GL-JEPA	18,180	512	10^{-5}	10^{-5}	772.6

All three interfaces use AdamW, the same five graph–description corpora, the same 12-layer Graph Encoder architecture, and the same evaluated update count. Table 1 reports three-pretraining-seed means for all interfaces. Language-side modules and parameter counts differ across interfaces. The Linear Predictor in Table 1 provides the nearly parameter-matched control. Graph–Language Latent Prediction uses normalized MSE with $1/(Bd_T)$. Both pretraining phases use $u_\varepsilon(\mathbf{v}) = \mathbf{v}/\max(\|\mathbf{v}\|_2, 10^{-8})$.

Ablation protocol. Each Table 4 variant changes only the component named in its row unless it removes a pretraining stage. Text Encoder variants use the selected frozen encoder for both node text and paired descriptions, with mean pooling and ℓ_2 normalization. For each encoder, Graph Context Pretraining is rerun for 5,454 updates, and its Graph Encoder checkpoint initializes 18,180 updates of Graph–Language Latent Prediction. The remaining data order, optimizer, batch size, and evaluation contract are fixed across these runs. Predictor-architecture, Predictor-input, and Graph Encoder variants retain the default settings in Table 8 except for the named change. The completed GraphSAGE variant uses four message-passing layers and is evaluated at step 18,180.

Directed-edge handling. The cached edge index preserves source-to-target direction. The GraphSAGE operator (Hamilton et al., 2017) aggregates messages along this stored direction. Preprocessing does not symmetrize the graph or add reverse edges.

Compute and availability. Experiments use one NVIDIA RTX PRO 6000 Blackwell GPU (96 GB), PyTorch 2.11 with CUDA 13.0, Python 3.12, bfloat16 precision, and AdamW. Training and evaluation code, data manifests, and checkpoint identifiers will accompany the released artifact.

E Interfaces, Decoding, and Instruction Tuning

This section specifies how the learned graph states are used in the controlled comparison, generation evaluation, valid-description scoring, and graph question answering.

Controlled supervision interfaces. Token trains the graph-to-language Llama decoder with teacher-forced next-token cross-entropy. Alignment compares the pooled graph representation and frozen description representation using symmetric InfoNCE with temperature 0.07. GL-JEPA predicts the frozen description target as

$$\widehat{\mathbf{z}}_Y = P_\omega(S_G), \qquad \mathbf{z}_Y = \mathrm{sg}(E_T(Y))\,. \tag{8}$$

The graph inputs, paired descriptions, Graph Encoder architecture, initialization, batch size, and evaluated update count are shared. The language-side interfaces remain method-specific.

Generation readout. For WebNLG generation, the frozen GL-JEPA checkpoint predicts one 384-dimensional description representation. A trainable adapter $q_\phi : \mathbb{R}^{384} \to \mathbb{R}^{16\times 2048}$ maps it through a two-layer Transformer to 16 virtual prefix tokens for a frozen `meta-llama/Llama-3.2-1B` decoder as

$$p_\phi(Y \mid G) = \prod_{t=1}^{|Y|} p(y_t \mid y_{<t}, q_\phi(\widehat{\mathbf{z}}_Y)) \,. \tag{9}$$

Only q_ϕ is optimized. The adapter is trained with teacher-forced description loss on cached predicted representations and paired descriptions from the five pretraining corpora in Table 5, not on WebNLG examples. Training uses 27,270 updates with batch size 4, gradient accumulation 4, AdamW learning rate 10^{-4}, weight decay 0.01, seed 90, and maximum target length 192. A held-out portion of the pretraining corpora is used for validation. Test generation is greedy (`do_sample=False`) with at most 128 new tokens.

Figure 3(c) reports training-update latency at batch size 8 on an NVIDIA RTX PRO 6000 Blackwell Max-Q. Token includes its Graph Encoder, teacher-forced causal decoder loss, backward pass, and optimizer update with maximum target length 192. GL-JEPA's reported total adds Graph Encoder and Predictor training, backward and optimizer update, and a separately measured teacher-forced generation-readout fitting step through the frozen decoder. Model loading, tokenization, data materialization, validation, logging, and checkpoint writing are excluded. Figure 3(d) reports graph-to-text inference on 64 WebNLG test examples with the same batch size, greedy decoding, and at most 32 generated tokens. Each measurement uses two warm-up batches and five repeats. Component timings use cached outputs from the preceding component. The GL-JEPA training total in panel (c) sums the separately timed pretraining and readout-fitting components, whereas inference totals in panel (d) are measured through the complete method-specific path.

Valid-description scoring. For dual-positive accuracy (DPA), Token scores candidates with length-normalized sequence likelihood, whereas GL-JEPA uses

$$s_{ij} = \cos(\widehat{\mathbf{z}}_{Y_i}, E_T(Y_{ij})) \,. \tag{10}$$

For two valid descriptions Y_{i1}, Y_{i2} and a negative Y_{i-},

$$\mathrm{DPA} = \frac{1}{N}\sum_{i=1}^{N} \mathbb{1}[s_{i1} > s_{i-} \wedge s_{i2} > s_{i-}]. \tag{11}$$

Ties count as failures. WebNLG uses 1,744 one-fact counterfactuals comprising 676 entity substitutions, 535 direction reversals, 304 numeric substitutions, and 229 relation substitutions. Each mutation changes one fact signature while retaining the remaining surface template. No model-based fluency filter is applied. DPA is the fraction of examples where both valid descriptions score above the constructed negative. DART uses 3,621 examples and pairs two valid references with a description from another graph.

Table 10: Decomposition of DPA (%). R1 and R2 denote the rates at which each valid reference outranks the negative. DPA requires both comparisons to succeed.

Dataset	Model	Examples	R1	R2	DPA
WebNLG	Token	1,744	51.78	55.85	35.55
WebNLG	**GL-JEPA**	1,744	69.67	76.78	60.78
DART	Token	3,621	50.51	51.73	39.82
DART	**GL-JEPA**	3,621	88.18	87.90	83.57

Query-conditioned instruction tuning. For each example, the valid sequence is

$$[\texttt{graph-summary}, \texttt{node}_1, \ldots, \texttt{node}_N, \texttt{query}_1, \ldots, \texttt{query}_K].$$

Query tokens use the Llama-3.2-1B tokenizer and input embedding matrix followed by a learned projection to d_P. Rotary positions continue across graph, node, and query tokens. Bidirectional attention is enabled between every valid position and masks only padding. The readout averages

valid query-token outputs. The Graph Encoder and frozen answer Text Encoder E_A are fixed, and only the query-conditioned Predictor and output projection are adapted as

$$\widehat{\mathbf{z}}_A = P_\omega^{\text{task}}(S_G, Q), \qquad s(a \mid G, Q) = \cos(\widehat{\mathbf{z}}_A, E_A(a)) \,. \tag{12}$$

The default configuration uses the same encoder checkpoint for E_A as for E_T. Encoder variants use their corresponding frozen Text Encoder. Candidate embeddings and predictions are ℓ_2-normalized, and candidate cross-entropy uses temperature 0.07.

Following the discriminative VQA evaluation used by VL-JEPA (Chen et al., 2026), we score the deterministic candidate set $\{\texttt{no}, \texttt{yes}\}$ against the query-conditioned prediction and select the higher-scoring answer. Training contains 11,433 examples, validation contains 1,267, and test contains 400 examples per task. Adaptation uses five epochs, batch size 8, candidate batch size 32, AdamW learning rate 10^{-5}, weight decay 0.05, and seed 90. Validation accuracy selects the checkpoint.

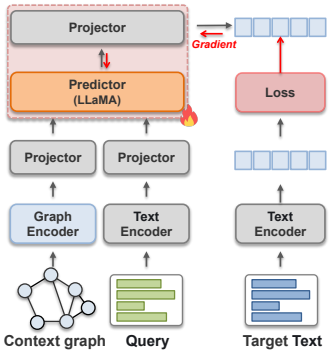


Figure 7: Query-conditioned adaptation. The Graph Encoder and answer Text Encoder E_A are frozen. The query-conditioned Predictor and output projection are adapted.