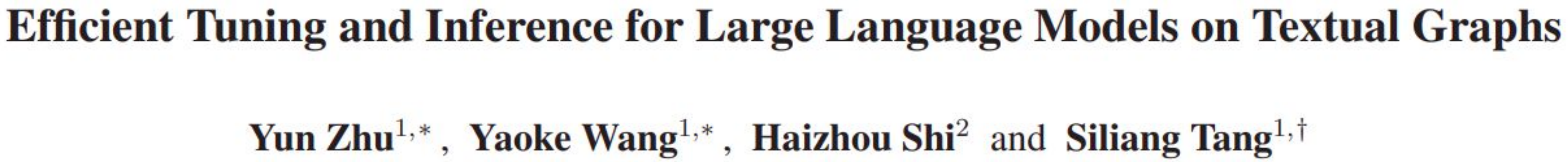
ENGINE



Efficient Tuning and Inference for Large Language Models on Textual Graphs

Yun Zhu[1,*], **Yaoke Wang**[1,*], **Haizhou Shi**[2] and **Siliang Tang**[1,†]

https://arxiv.org/pdf/2401.15569

Key Points

- Using LLM for enhanced textual encoding and integration with GNNs.
- parameter- and memory-efficient fine-tuning method
- Combining the LLMs and GNNs through a **tunable side structure.**
- caching and dynamic early exit
- Freezing weights of the LLM

Earlier methods (A) [Static Embeddings]

static shallow embedding

struggle to capture context-aware information and complex semantic relationships

(e.g., bag-of-words, skip-gram)

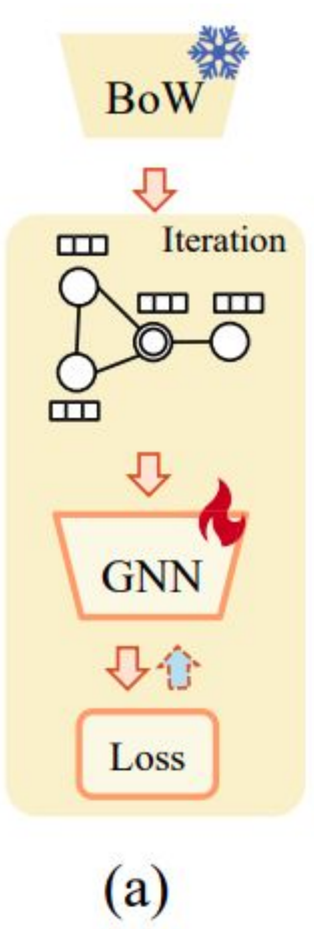


Figure 2: Zhu et al [1].

Earlier methods (B) [Cascading]

combine LMs and GNNs in a cascading

- the initial step involves finetuning pre-trained language models on downstream tasks

enhances the meaningfulness of node embeddings by fine-tuning LMs using neighbor information.

- initially fine-tuning LMs through graph-related tasks (e.g., node classification, link prediction)

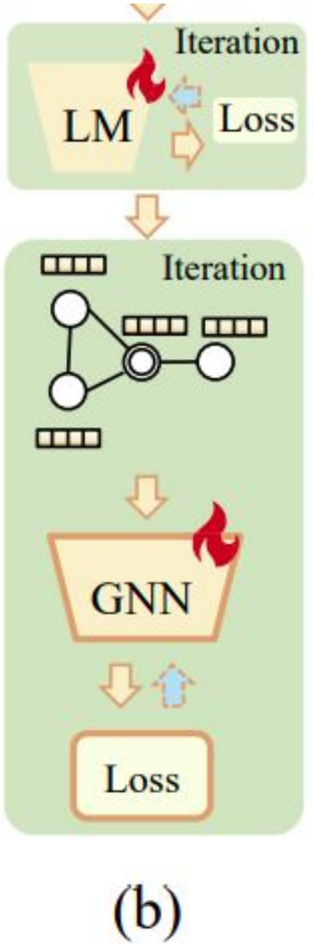


Figure 2: Zhu et al [1].

Earlier methods (C) [Iterative]

combine LMs and GNNs in an iterative or co-training structure

these co-training paradigms face significant scalability issues.

Encoding all neighbors by LMs introduces high-cost fine-tuning and inference overhead due to the substantial number of parameters in language models

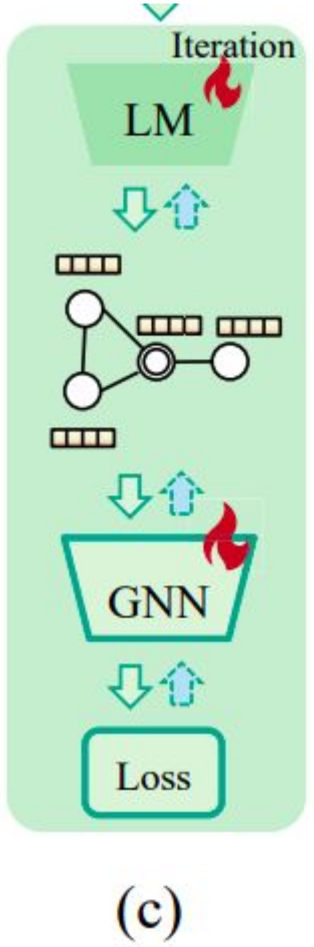
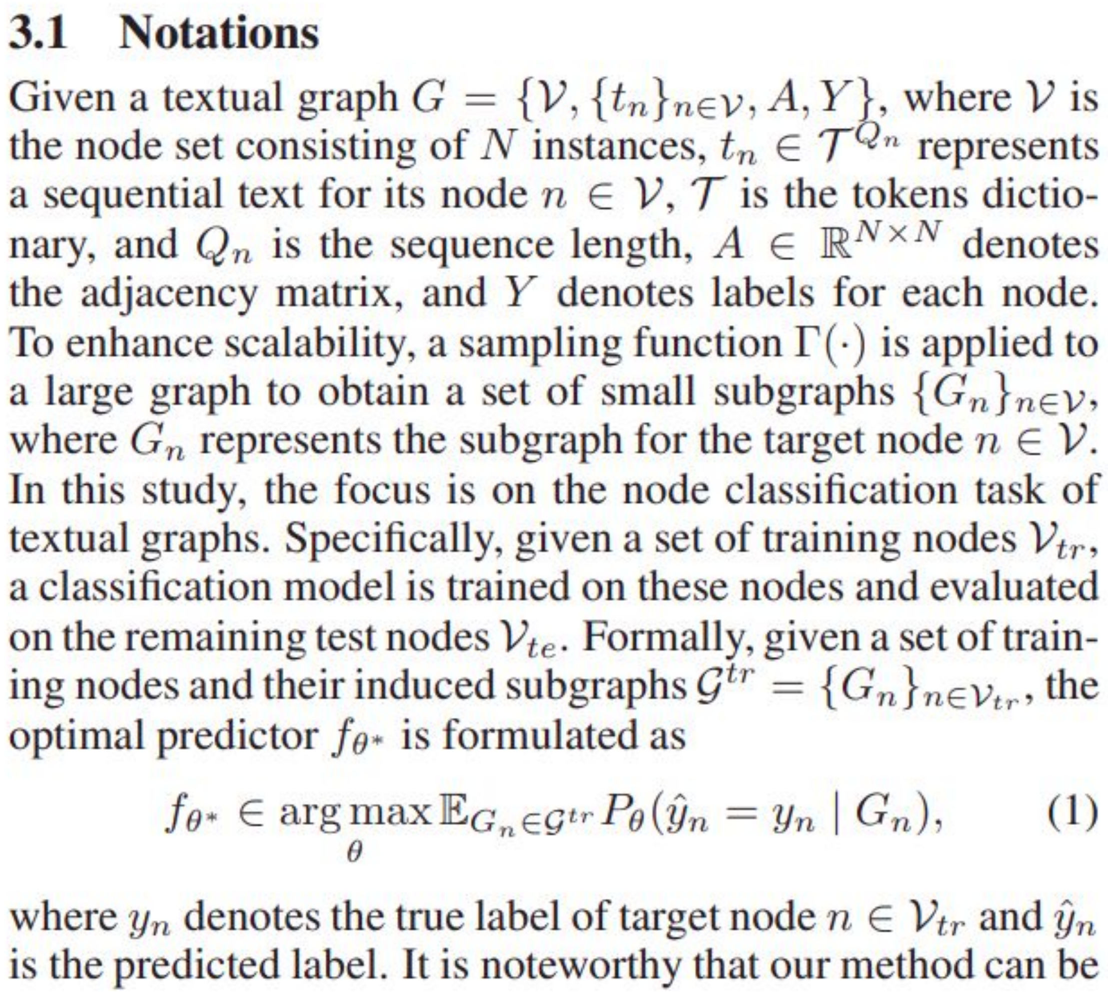


Figure 2: Zhu et al [1].



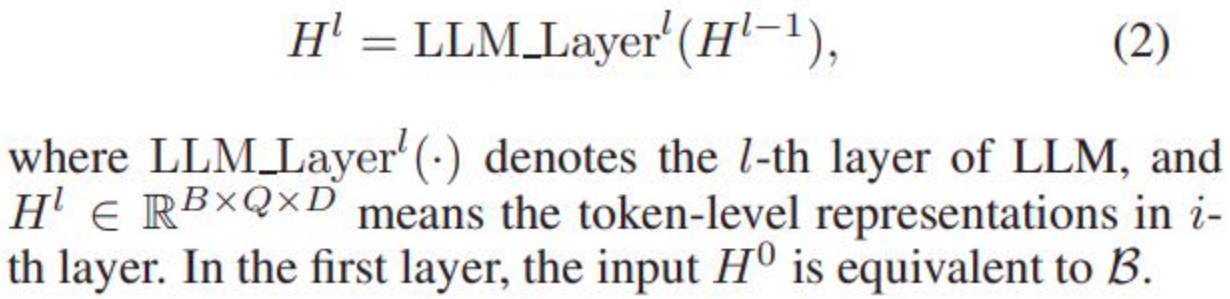
3.1 Notations

Given a textual graph $G = \{\mathcal{V}, \{t_n\}_{n \in \mathcal{V}}, A, Y\}$, where $\mathcal{V}$ is the node set consisting of N instances, $t_n \in \mathcal{T}^{Q_n}$ represents a sequential text for its node $n \in \mathcal{V}$, $\mathcal{T}$ is the tokens dictionary, and Q_n is the sequence length, $A \in \mathbb{R}^{N \times N}$ denotes the adjacency matrix, and Y denotes labels for each node. To enhance scalability, a sampling function $\Gamma(\cdot)$ is applied to a large graph to obtain a set of small subgraphs $\{G_n\}_{n \in \mathcal{V}}$, where G_n represents the subgraph for the target node $n \in \mathcal{V}$. In this study, the focus is on the node classification task of textual graphs. Specifically, given a set of training nodes $\mathcal{V}_{tr}$, a classification model is trained on these nodes and evaluated on the remaining test nodes $\mathcal{V}_{te}$. Formally, given a set of training nodes and their induced subgraphs $\mathcal{G}^{tr} = \{G_n\}_{n \in \mathcal{V}_{tr}}$, the optimal predictor f_{θ^*} is formulated as

$$f_{\theta^*} \in \arg\max_{\theta} \mathbb{E}_{G_n \in \mathcal{G}^{tr}} P_\theta(\hat{y}_n = y_n \mid G_n), \qquad (1)$$

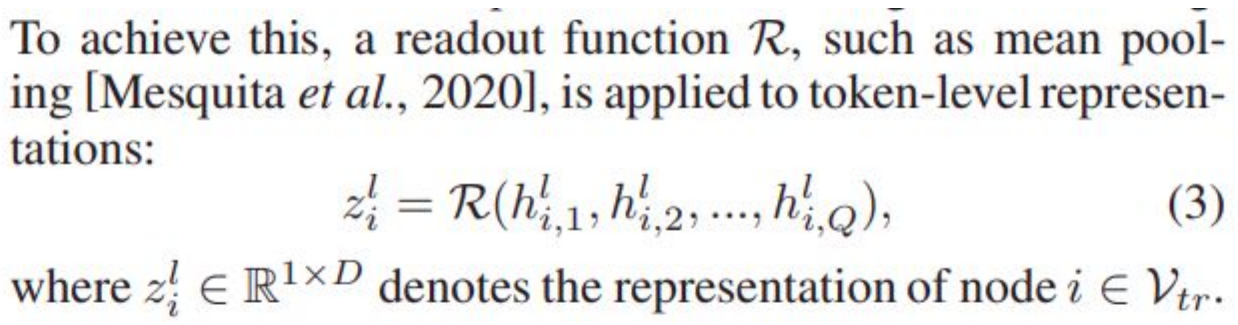
where y_n denotes the true label of target node $n \in \mathcal{V}_{tr}$ and $\hat{y}_n$ is the predicted label. It is noteworthy that our method can be

Zhu et al [1].



$$H^l = \text{LLM\_Layer}^l(H^{l-1}), \tag{2}$$

where $\text{LLM\_Layer}^l(\cdot)$ denotes the l-th layer of LLM, and $H^l \in \mathbb{R}^{B\times Q\times D}$ means the token-level representations in i-th layer. In the first layer, the input H^0 is equivalent to $\mathcal{B}$.

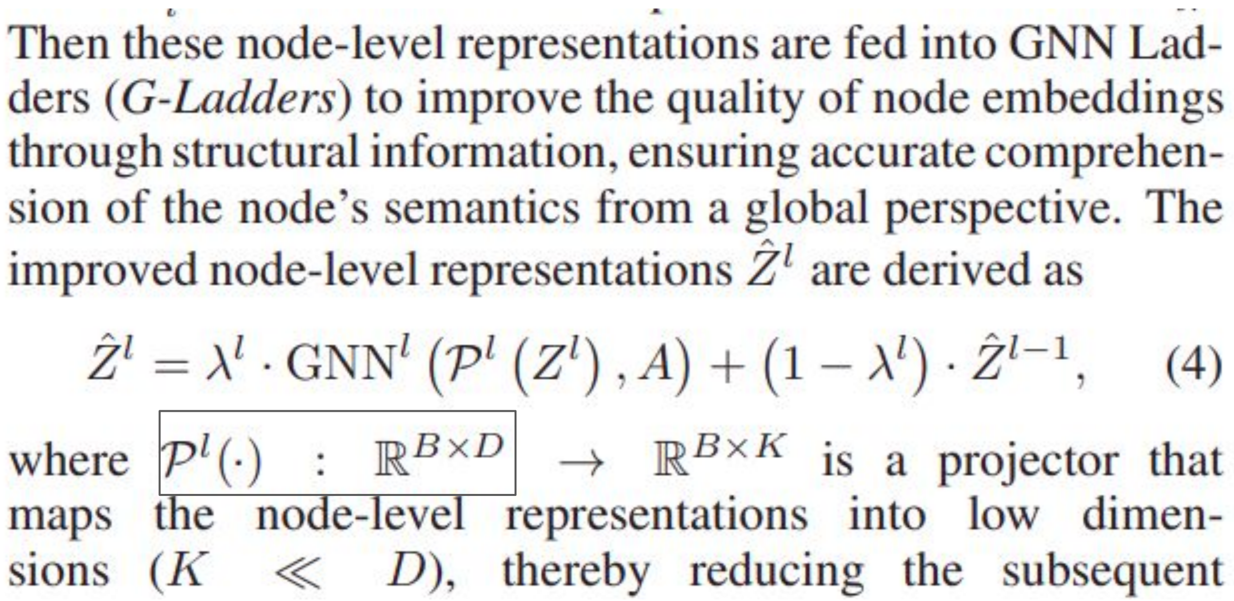


To achieve this, a readout function $\mathcal{R}$, such as mean pooling [Mesquita *et al.*, 2020], is applied to token-level representations:

$$z_i^l = \mathcal{R}(h_{i,1}^l, h_{i,2}^l, ..., h_{i,Q}^l), \tag{3}$$

where $z_i^l \in \mathbb{R}^{1\times D}$ denotes the representation of node $i \in \mathcal{V}_{tr}$.

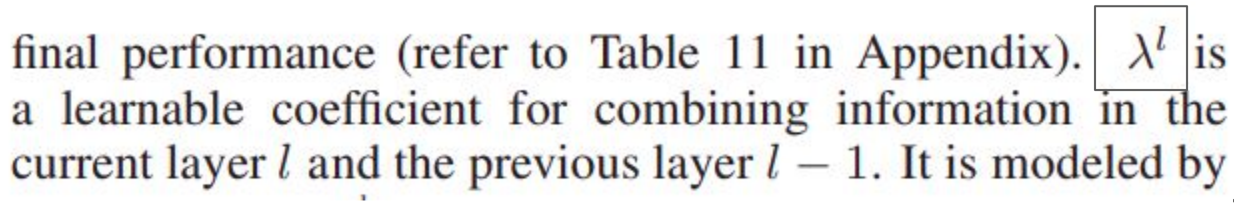
Zhu et al [1].



Then these node-level representations are fed into GNN Ladders (*G-Ladders*) to improve the quality of node embeddings through structural information, ensuring accurate comprehension of the node's semantics from a global perspective. The improved node-level representations $\hat{Z}^l$ are derived as

$$\hat{Z}^l = \lambda^l \cdot \mathrm{GNN}^l\left(\mathcal{P}^l\left(Z^l\right), A\right) + \left(1 - \lambda^l\right) \cdot \hat{Z}^{l-1}, \qquad (4)$$

where $\boxed{\mathcal{P}^l(\cdot) \; : \; \mathbb{R}^{B\times D}} \rightarrow \mathbb{R}^{B\times K}$ is a projector that maps the node-level representations into low dimensions ($K \ll D$), thereby reducing the subsequent



final performance (refer to Table 11 in Appendix). $\boxed{\lambda^l}$ is a learnable coefficient for combining information in the current layer l and the previous layer $l-1$. It is modeled by

Zhu et al [1].

$$\mathcal{L} = \mathbb{E}_{i \in \mathcal{V}_{tr}} \, \mathrm{CE}(\hat{y}_i, y_i), \text{ where } \hat{y}_i = \mathcal{C}(\hat{z}_i^L).$$

Model Architecture

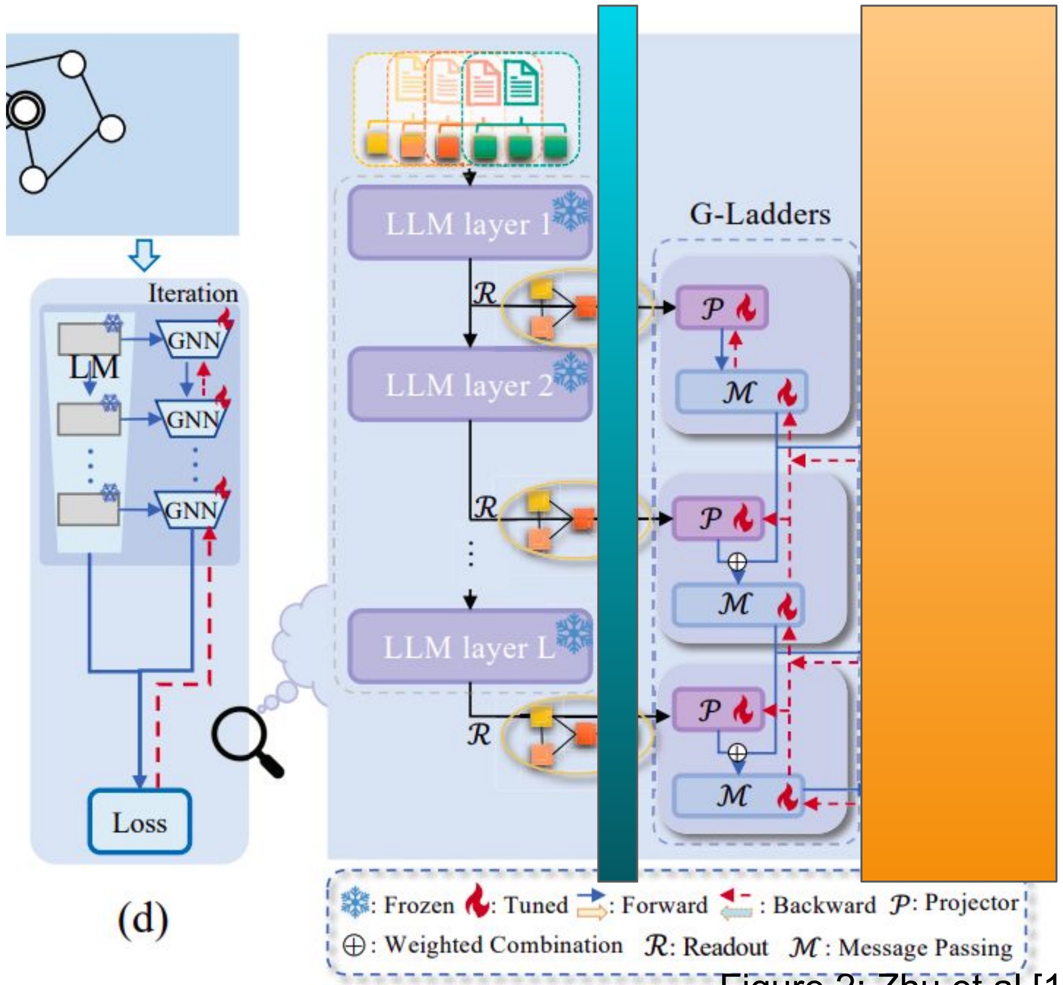
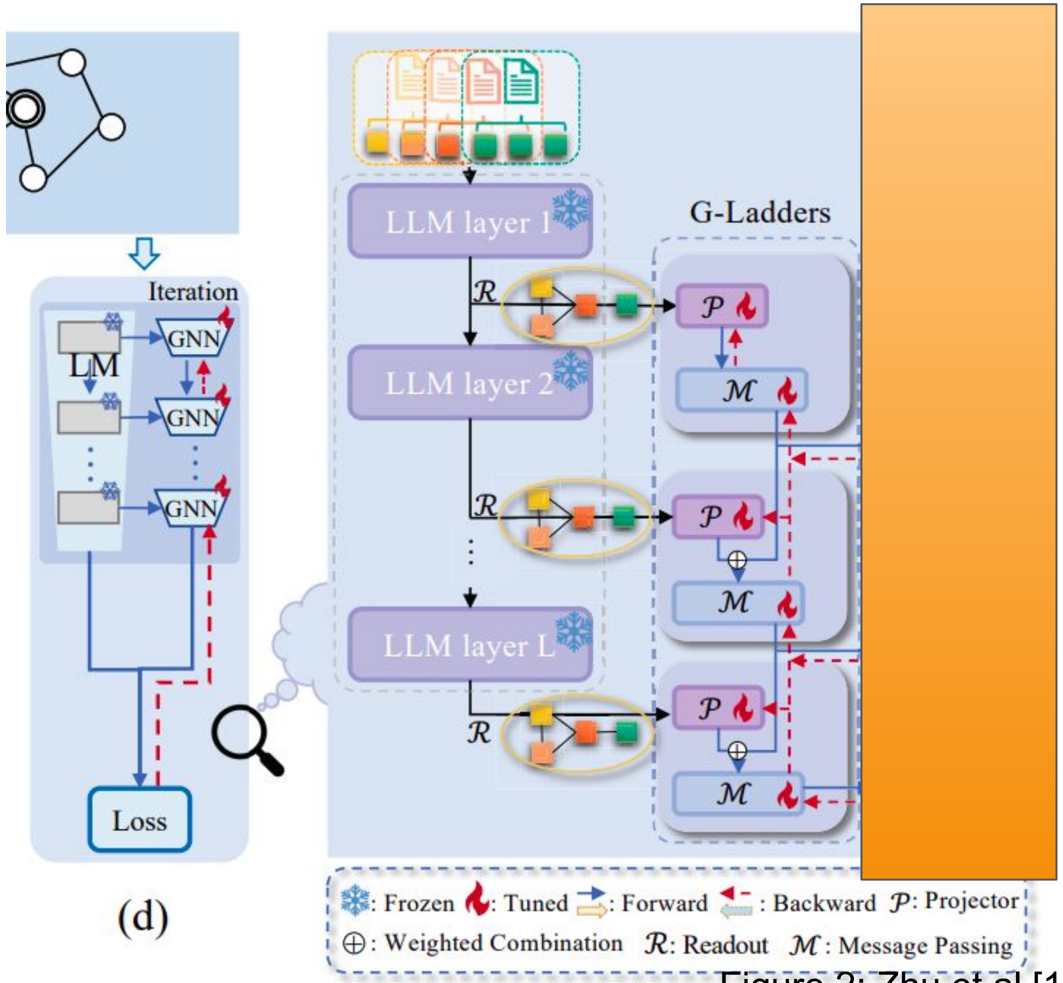

Iteration
LM
GNN
GNN
GNN
Loss
(d)
LLM layer 1
LLM layer 2
LLM layer L
R
R
R
G-Ladders
P
M
P
M
P
M
: Frozen : Tuned : Forward : Backward P: Projector
⊕: Weighted Combination R: Readout M: Message Passing

Figure 2: Zhu et al [1].

Caching

G-Ladders with LLMs via a side structure, eliminating the need for backpropagation through the LLMs.

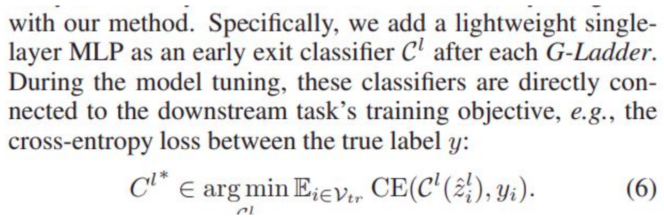
allows us to precompute and cache node embeddings for reuse, further boosting computational efficiency



Iteration
GNN
GNN
GNN
LM
Loss
(d)
LLM layer 1
LLM layer 2
LLM layer L
G-Ladders
$\mathcal{R}$
$\mathcal{P}$
$\mathcal{M}$
❄: Frozen 🔥: Tuned : Forward : Backward $\mathcal{P}$: Projector
⊕: Weighted Combination $\mathcal{R}$: Readout $\mathcal{M}$: Message Passing

Figure 2: Zhu et al [1].

Early Stopping



with our method. Specifically, we add a lightweight single-layer MLP as an early exit classifier $\mathcal{C}^l$ after each *G-Ladder*. During the model tuning, these classifiers are directly connected to the downstream task's training objective, *e.g.*, the cross-entropy loss between the true label y:

$$C^{l^*} \in \arg\min \mathbb{E}_{i\in\mathcal{V}_{tr}} \mathrm{CE}(\mathcal{C}^l(\hat{z}_i^l), y_i). \qquad (6)$$

Patience-based criteria imply that if consecutive p early exit classifiers predict the same results, where p represents the number of layers.

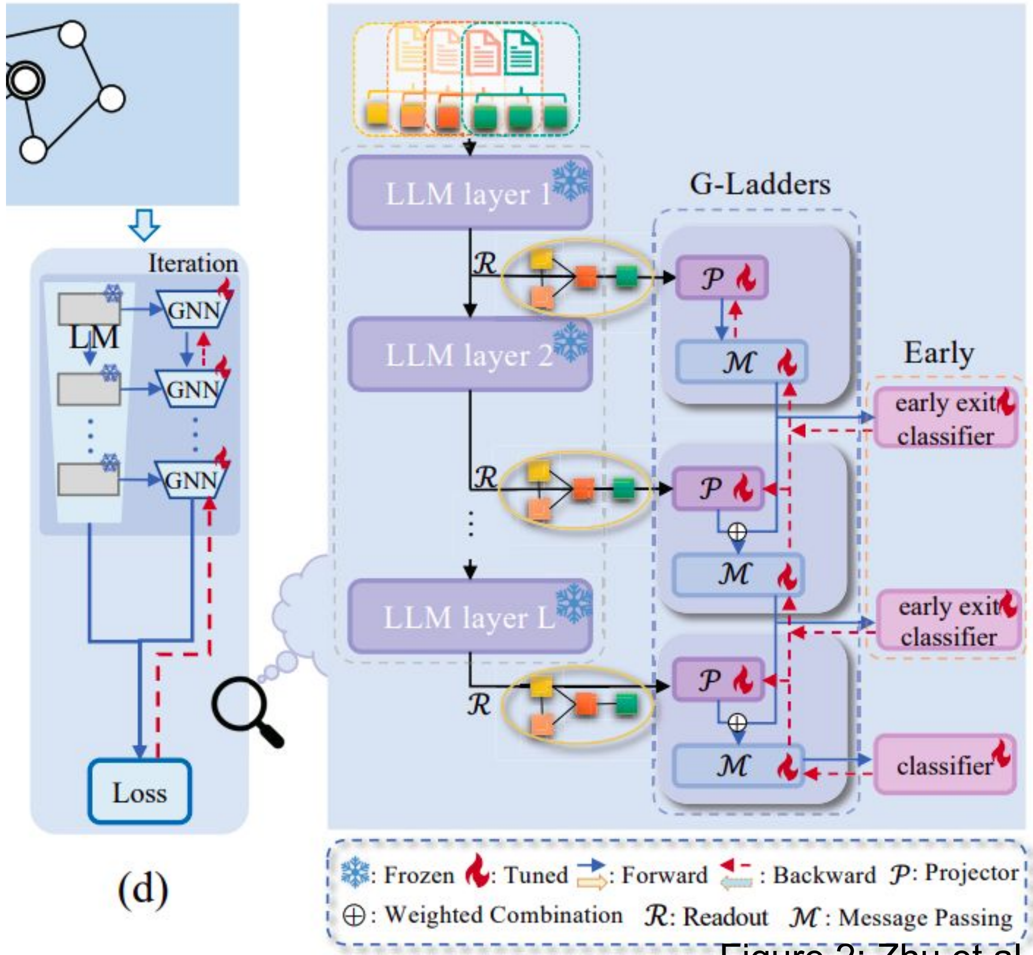


Figure 2: Zhu et al [1].

bibliography

1. Efficient Tuning and Inference for Large Language Models on Textual Graphs (https://arxiv.org/pdf/2401.15569)