GNNAVI: Navigating the Information Flow in Large Language Models by Graph Neural Network

Anonymous ACL submission

Abstract

Large Language Models (LLMs) exhibit strong In-Context Learning (ICL) capabilities when prompts with demonstrations are applied to them. However, fine-tuning still remains crucial to further enhance their adaptability. Prompt-based fine-tuning proves to be an effective fine-tuning method in low-data scenarios, but high demands on computing resources limit its practicality. We address this issue by introducing a prompt-based *parameter-efficient fine-tuning (PEFT)* approach. **GNNAVI** leverages insights into ICL's information flow dynamics, which indicates that label words act in prompts as anchors for information propagation. GNNAVI employs a *Graph Neural Network (GNN)* layer to precisely guide the aggregation and distribution of information flow during the processing of prompts by hardwiring the desired information flow into the GNN. Our experiments on text classification tasks with GPT-2 and Llama2 shows GNNAVI surpasses standard prompt-based fine-tuning methods in few-shot settings by updating just 0.2% to 0.5% of parameters. We compare GNNAVI with prevalent PEFT approaches, such as prefix tuning, LoRA and Adapter in terms of performance and efficiency. Our analysis reveals that GNNAVI enhances information flow and ensures a clear aggregation process.[1]

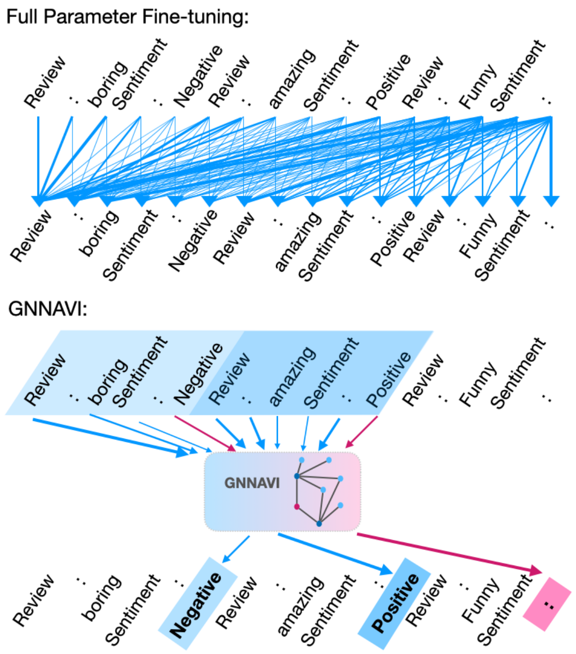


Figure 1: Visualization of Full Parameter Fine-tuning (FPFT) and GNNAVI from the perspective of information flow (top words to bottom words). Without GNNAVI, tokens interact with every preceding word in FPFT, leading to confusion in information flow. Conversely, in GNNAVI, label words aggregate information from preceding words (blue path), and the final token aggregates information from the label words (pink path), resulting in a clearer information aggregation process.

1 Introduction

Large language models (LLMs) show remarkable In-Context-Learning (ICL) capabilities by learning from prompts with demonstrations (Wan et al., 2023; Sun et al., 2023; Patel et al., 2023; Mekala et al., 2023; Ko et al., 2023), with the exponential growth in model sizes. However, fine-tuning LLMs still remains essential for further enhancing their adaptability (Zhang et al., 2023). Prompt-based fine-tuning (Schick and Schütze, 2021a; Ma et al., 2024), adopting objectives that simulate the language modeling process, emerges as a viable technique, particularly in low-data settings (Gao et al., 2021). Yet, the substantial computational demands of Full-Parameter Fine-Tuning (FPFT), which updates billions of parameters, pose a practical challenge. In fact, optimizing a relatively small subset of an LLM's parameters can significantly improve its performance (Ding et al., 2023), paving the way for Parameter-Efficient Fine-Tuning (PEFT) methods. These methods include Adapter (Houlsby et al., 2019), Prompt-Tuning (Lester et al., 2021),

[1] Our code is anonymously available at https://anonymous.4open.science/r/GNNAVI-8CB9.

Prefix Tuning (Li and Liang, 2021), and LoRA (Hu et al., 2022). They offer alternatives to FPFT but are often not tailored to the prompt-based fine-tuning of LLMs.

Recent advances in understanding the ICL mechanism offer a new avenue for PEFT of LLMs. ICL's success in leveraging few-shot demonstrations and prompts (Brown et al., 2020) has motivated the adoption of prompt-based fine-tuning for moderately sized language models in a few-shot learning manner (Ma et al., 2023; Schick and Schütze, 2021b). Recognizing the specific features of fine-tuning LLMs within the framework of ICL, we propose **GNNAVI**, a novel PEFT method designed expressly for prompt-based learning. Our method draws inspiration from recent insights into the underlying process of ICL from an information flow perspective, particularly the role of label words in the prompt (Wang et al., 2023). Label words act as anchors with two functions: aggregating information from context words and directing this information to the last token for accurate predictions. GNNAVI incorporates this understanding through the integration of a Graph Neural Network (GNN) layer (Kipf and Welling, 2017; Hamilton et al., 2017) into LLMs, optimizing the prompt-based fine-tuning process by navigating the information flow within prompts, as visualized in Figure 1. Following the paths of information flow, we insert a GNN layer into the deep layers[2] of the LLM. We treat the input text as a graph, where each token serves as a node, and connect these nodes according to the paths of information flow. The GNN layer aims to guide the information flow by aggregating information from neighbouring nodes.

As a PEFT method, GNNAVI adopts a lightweight fine-tuning strategy, updating only the parameters of the GNN layer. Experimenting with few-shot training examples on GPT2-XL (Radford et al., 2019) and Llama2 (Touvron et al., 2023), GNNAVI achieves remarkable results with just 0.2% of the trainable parameters of the full model, consistently outperforming FPFT and other PEFT methods across various classification tasks. Additionally, we analyze the attention interaction between tokens and find that GNNAVI demonstrates a more stable and clear information aggregation process compared to FPFT.

In summary, our contributions are: **i)** We propose a novel PEFT method, GNNAVI, inspired by the information flow perspective of LLMs. GNNAVI effectively navigates the information aggregation process in LLMs. **ii)** We apply GNNAVI to text classification tasks with few-shot training examples, outperforming baselines while updating only 0.2% to 0.5% of parameters. **iii)** Our work sheds light on the application of GNNs in NLP and provides novel insights for future research. To the best of our knowledge, we are the first to utilize GNNs to enhance the performance of LLMs from the information flow perspective.

[2]We use "deep layers" to refer to the last few layers of the LLM. For instance, in GPT2-XL, there are 48 layers, with the last 12 layers considered as deep layers in our work.

2 Related Work

Prompt-Based Learning GPT-3 (Brown et al., 2020) has sparked interest in prompt-based learning methods, and particularly in the ICL paradigm. This surge in attention has fostered a multifaceted exploration into the factors influencing ICL performance, including input perturbation (Yoo et al., 2022; Min et al., 2022), selection of demonstration (Liu et al., 2022; Nie et al., 2023a), and calibration techniques (Zhao et al., 2021a; Nie et al., 2023b). Concurrently, there has been a deep dive into understanding the underlying mechanism of ICL, employing diverse theoretical frameworks such as gradient descent (Dai et al., 2023), Bayesian inference (Xie et al., 2022) and information flow (Wang et al., 2023). Following the route of ICL, prompt-based fine-tuning has emerged as an effective strategy in scenarios with limited data (Gao et al., 2021; Schick and Schütze, 2021a,b). We leverage insights from these investigations into the ICL mechanism and propose a tailored PEFT method for LLMs.

Parameter-Efficient Fine-Tuning (PEFT) PEFT focuses on enhancing language model performance on downstream tasks by optimizing a small number of parameters, instead of fine-tuning all parameters (Ding et al., 2023). Various PEFT strategies have been explored. Addition-based methods only train modules or parameters added to the model, such as Adapter (Houlsby et al., 2019), Prompt tuning (Lester et al., 2021), and Prefix tuning (Li and Liang, 2021). Specification-based methods selectively fine-tune specific parameters in the original model while keeping the remainder frozen, such as BitFiT (Ben Zaken et al., 2022). Reparameterization-based methods transform existing parameters into a more parameter-efficient form, such as LoRA (Hu et al., 2022). Recent

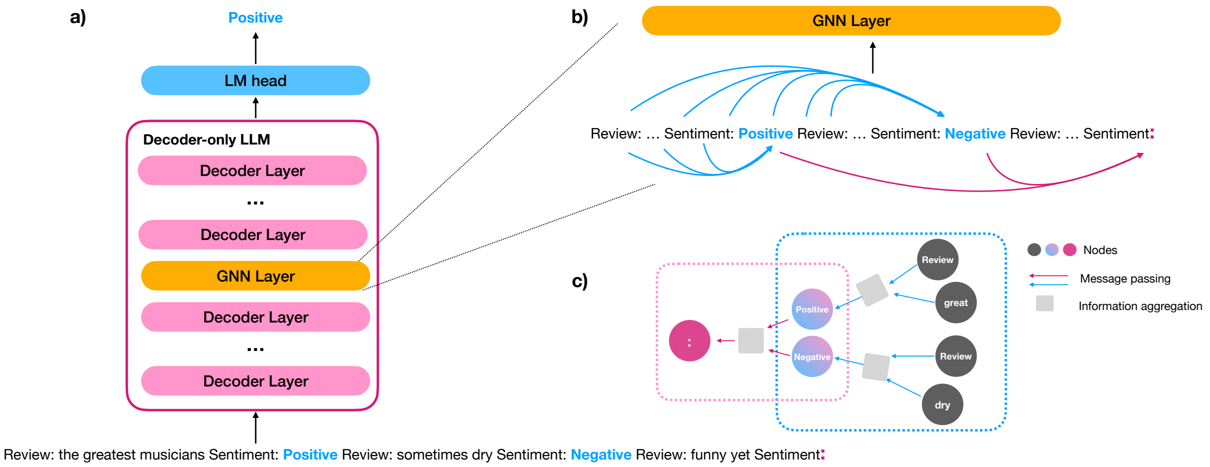


Figure 2: Visualization of GNNAVI with an example of sentiment analysis, where label words and the last token are highlighted in blue and pink, respectively. a) The GNN layer is integrated into a decoder-only LLM. The LLM processes a prompt containing demonstrations and generates the next token as the prediction. b) The input text is transformed into a graph, with tokens as nodes and information flow paths as edges. c) Visualizing the working mechanism of the GNN: Node representations are updated by aggregating information from neighboring nodes. To maintain simplicity, not all nodes are listed.

advancements in PEFT research have increasingly prioritized memory efficiency, aiming to enable the training of LLMs with minimal computational resources, such as MeZO (Malladi et al., 2023) and HiFT (Liu et al., 2024). Our proposed PEFT method is designed specifically for LLMs and draws upon the intricacies of how LLMs process and learn from prompts.

GNN for NLP GNNs are predominantly utilized in NLP tasks involving structural input, such as graph-to-text generation (Gardent et al., 2017) and graph-enhanced question answering (Zhang et al., 2022). Previous approaches employ GNNs to encode complex graph and node representations. For instance, Koncel-Kedziorski et al. (2019) introduced Graph Transformer, which extends graph attention networks (Veličković et al., 2018) for encoding scientific graph inputs, while Li et al. (2021) utilize GNNs to encode knowledge graphs and align them with text embeddings from pretrained language models. Additionally, GNNs serve as auxiliary tools for pretrained language models to encode complex structural information for AMR-to-text generation (Ribeiro et al., 2021). Unlike prior work, we leverage GNNs for information aggregation based on the perspective of information flow.

3 Method

3.1 Architecture of GNNAVI

Intuition Wang et al. (2023) demonstrated that the working mechanism of LLM follows specific paths of information flow. The label words in the input prompt serve two roles for the final predictions: acting as information aggregators by gathering information from their preceding words and propagating the aggregated information to the last token position where the prediction is generated. Building upon their insights, we posit that navigating the flow of information aggregation can enhance both efficiency and effectiveness of LLMs. Leveraging the GNN's proficiency in information aggregation at the graph level, we explore LLMs from a graph theory perspective and utilize GNN as a tool to guide the information flow.

Working Mechanism We illustrate the working mechanism of GNNAVI in Figure 2. For example, in a sentiment analysis task, the prompt comprises one demonstration from each class and the text to be classified. An LLM processes this prompt layer by layer. The GNN layer is inserted after the l-th decoder layer of the LLM[3]. Receiving the token

[3]In our preliminary experiments, GNNAVI performs optimally when the GNN layer is inserted in the last quarter of the layers in LLM. Thus, we add the GNN layer after the 42nd layer of GPT2-XL and after the 28th layer of Llama2-7b in our experiments. A detailed analysis is conducted in §6.1.

representations from the l-th layer, the GNN layer learns node representations by aggregating information from neighboring nodes. Subsequently, the node representations are propagated to the next layer in LLM as hidden states. The nodes are connected following the paths of information flow. As depicted in Figure 2(b), the label words '*Positive*' and '*Negative*' aggregate information from their preceding tokens and pass the information to the last token '*:*' of the prompt. In case the label word is tokenized into subtokens, we use the first subtoken to serve as the label word, following previous work (Zhao et al., 2021b; Wang et al., 2023). We freeze the pretrained parameters of the LLM during training and update only the parameters in the GNN layer.

Graph Neural Network The graph neural network aggregates information from neighboring nodes to model graph and node representations by message passing. To formulate an NLP task on a graph level, we consider the input text as a graph. We define a directed graph $\mathcal{G}$ as a triple $(\mathcal{V}, \mathcal{E}, \mathcal{R})$ with a set of nodes $\mathcal{V} = \{v_1, \ldots, v_n\}$ (one node for each token), a set of relation types $\mathcal{R}$[4], and a set of edges $\mathcal{E}$ of the form (v, r, v') with $v, v' \in \mathcal{V}$, and $r \in \mathcal{R}$. Each node v_i is associated with a feature vector x_i, which is the token representation of the i-th token in the l-th layer. In Figure 2, for instance, the first token '*Review*' is connected with the label token '*Positive*'. This edge is represented by the triple (*Review*, $aggregate$, *Positive*), where $aggregate$ denotes an edge directed towards a label node.

The node representations in GNN layer are updated by aggregating the information from neighboring nodes. The aggregation algorithms vary across different GNN architectures. For example, the learning process of Graph Convolutional Network (GCN) (Kipf and Welling, 2017) is formulatd as:

$$h_v = \sigma\left(W \sum_{v' \in N(v)} \frac{h_{v'}^{(l)}}{|N(v)|}\right) \quad (1)$$

where h_v denotes the updated node representation of v, $h_{v'}^{(l)}$ denotes the token representation of its neighbouring nodes from l-th decoder layer, σ is the activation function, W is the trainable parameter of GNN, $N(v)$ includes all the neighbouring nodes of v.

[4]In our work, we only consider one relation type: the directed edge from node v to node v'.

We also include another GNN architecture, GraphSAGE (Hamilton et al., 2017), in our studies, which involves a more complex learning process:

$$h_v = \sigma\left(W\left(h_v^{(l)} \oplus \text{AGG}(\{h_{v'}^{(l)}, \forall v' \in N(v)\})\right)\right) \quad (2)$$

The concatenation function $\oplus$ concatenates aggregated information with the node current representation, and the aggregation function AGG compiles message passing from neighboring nodes using techniques such as mean, pool and LSTM.[5] We visualize the information aggregation process of GNN in Figure 2(c).

3.2 Task Formulation

In our work, we implement prompt-based finetuning for text classification tasks. Our goal is to predict the correct class given a few examples. We reformulate the task as a language modeling problem. Let M be a language model with vocabulary V, and let $\mathcal{L}$ be a set of label words. The training set $\mathcal{T}$ consists of pairs (s, l), where s is a sequence of tokens from the vocabulary V and l is a label word from the set $\mathcal{L}$. In a sentiment analysis task, for instance, we define a pattern $\mathcal{P}(s, l)$ which associates a text $s =$'Nice performance' and a label word $l =$'Positive' as follows:

Review: Nice performance. Sentiment: Positive

For a k-class classification task, we sample one demonstration per class from the training set $\mathcal{T}$, and concatenate them with the text s to be classified to form the prompt $X(s)$:

$$X(s) = \mathcal{P}(s_1, l_1) \oplus \ldots \oplus \mathcal{P}(s_k, l_k) \oplus \mathcal{P}(s, \varepsilon) \quad (3)$$

$\oplus$ denotes the concatenation of the input demonstrations and ε is the empty string. A more intuitive example is shown in Figure 2. The language model reads the prompt $X(s)$ and predicts the next token l, which is the label assigned to s. M is initialized with pretrained parameters ϕ, and fine-tuned by minimizing the cross-entropy loss:

$$\ell = -\sum_{(s,l) \in \mathcal{T}} \log p_\phi(X(s), l) \quad (4)$$

$p_\phi(.,.)$ returns the probability which M assigns to the correct label l. In our work, we randomly select one demonstration per class to form the prompt and remove them from $\mathcal{T}$. The training examples are then sampled from the remaining samples in $\mathcal{T}$.

[5]We apply mean aggregation to GraphSAGE in this work.

4 Experiments

4.1 Datasets

We implement text classification tasks using five commonly used datasets from different domains, including **SST-2:** Stanford Sentiment Treebank Binary for sentiment analysis (Socher et al., 2013); **EmoC:** EmoContext for 4-label emotion classification (Chatterjee et al., 2019); **TREC:** Text REtrieval Conference Question Classification (TREC) for question type classification containing 6 types (Li and Roth, 2002; Hovy et al., 2001); **Amazon:** binary classification for Amazon reviews (McAuley and Leskovec, 2013); **AGNews:** AG's news topic classification dataset for topic classification with 4 labels (Zhang et al., 2015).

4.2 Experimental Setting

The prompt is designed following the template in Equation 3. We take one demonstration per class to form the prompt[6] and append the sample to be predicted at the end of the prompt. Following a few-shot learning setting, we experiment with different numbers of training samples, namely 5, 10, 20, 50, 100, and 200 samples per class. The training samples are randomly selected from the original training set. Another 1000 samples from the original training set are sampled as the validation set, and 1000 samples from the original test set are used for evaluation.[7] The accuracy on the validation set is employed to identify the best-performing model, which is subsequently evaluated on the test set. We report the average accuracy over five random seeds. The hyperparameters can be found in Appendix A.

4.3 Models

As GNNAVI is built on the base of decoder-only LLMs, we select two large language models, both with over 1 billion parameters, and equip them with GNNAVI. Specifically, we choose GPT2-XL with 1.6 billion parameters (Radford et al., 2019) and Llama2 with 7 billion parameters (Touvron et al., 2023). For the GNN layer, we opt for GCN and GraphSAGE, denoted as **GNNAVI-GCN** and **GNNAVI-SAGE** in the experiments. To integrate GNNAVI with GPT2-XL and Llama2, we modify their source codes from Huggingface (Wolf et al., 2019) and utilize GNN models provided by PyTorch Geometric (Fey and Lenssen, 2019).

[6] The templates of prompts are presented in Appendix B.

[7] The original test set of SST-2 contains less than 1000 samples, so we keep the original test set for model evaluation.

4.4 Baselines

ICL one-shot per class: In-context learning (ICL) follows the scenario where the LLM is initialized with pre-trained parameters and instructed by demonstrations to perform text classification tasks. None of the model parameters are updated. We sample one demonstration per class to form the prompt. The demonstrations used to form the prompt are consistent with those used for other methods under the same random seed.

ICL few-shot per class: To compare with the low-data fine-tuning setting, we implement ICL with 5 additional shots per class as the demonstrations. This setting is comparable to a training set with a size of 5 samples per class. Due to the limited input length of GPT2-XL, AGNews and Amazon are set to 4 additional shots per class.

Low-Rank Adaptation (LoRA): LoRA is a PEFT method that reduces the number of trainable parameters by injecting trainable rank decomposition matrices into each layer of the LLM (Hu et al., 2022). We implement LoRA using the Python library PEFT (Mangrulkar et al., 2022).

Prefix-tuning (Prefix): Prefix-tuning utilizes a soft-prompt strategy, incorporating virtual tokens into the LLM and updating only the parameters of the virtual tokens (Li and Liang, 2021). We implement prefix-tuning using the PEFT library (Mangrulkar et al., 2022). The number of virtual tokens[8] is set to maintain a comparable size of trainable parameters as for GNNAVI.

Adapter: We insert a standard adapter module after the feed-forward sub-layer of each layer in the LLM (Houlsby et al., 2019). The adapter module is added using AdapterHub (Pfeiffer et al., 2020; Poth et al., 2023).

Full Parameter Fine-tuning (FPFT): Full parameter fine-tuning is implemented as a strong baseline, where all the model parameters are updated during the training process.

5 Results

We report the results with 5 and 200 training examples in Table 1, which reflect the performance under the scenarios where only limited training examples are available and sufficient training examples are provided respectively. Full results are presented in Appendix C.

[8] The number of virtual tokens can be found in Appendix A.

Method	#Param	SST-2	EmoC	TREC	Amazon	AGNews	Average	#Param	SST-2	EmoC	TREC	Amazon	AGNews	Average
	GPT2-XL							Llama2						
$k = 0$														
ICL	-	55.44	6.48	54.68	53.32	72.12	48.41	-	67.55	9.60	70.36	94.98	84.14	65.33
$k = 5$														
ICL	-	63.17	6.30	57.68	53.67	50.43	46.25	-	86.93	20.18	45.72	92.30	80.16	65.06
LoRA	2.5M	91.98	50.60	75.20	88.80	**85.20**	78.36	4.2M	**95.42**	64.20	**88.40**	91.80	86.60	85.28
Prefix	6.1M	59.13	73.46	32.92	60.00	75.40	60.18	39.3M	50.96	58.56	21.36	49.36	25.78	41.20
Adapter	15.4M	79.82	76.00	**79.60**	**91.45**	81.25	81.62	198M	50.92	**84.05**	18.80	49.45	24.80	45.60
FPFT	1.6B	62.13	61.30	65.28	73.00	80.82	68.51	6.7B	94.63	61.92	81.72	**95.86**	**87.58**	84.34
GNNAVI-GCN	2.6M	**84.31**	75.48	76.72	90.90	83.16	**82.11**	16.8M	94.56	78.30	83.2	94.00	86.25	86.63
GNNAVI-SAGE	5.1M	81.95	**78.70**	77.92	88.66	82.88	82.02	33.6M	92.91	80.12	80.80	95.66	86.06	**87.11**
$k = 200$														
LoRA	2.5M	**90.83**	80.80	90.80	82.00	86.20	86.13	4.2M	91.29	**86.80**	93.60	95.80	90.40	91.32
Prefix	6.1M	50.92	80.18	69.80	59.80	79.08	67.96	39.3M	48.35	81.72	45.68	52.28	27.54	51.11
Adapter	15.4M	88.65	80.70	**96.60**	92.30	**89.80**	89.61	198M	50.92	85.05	88.20	49.45	81.50	67.57
FPFT	1.6B	68.97	73.70	80.16	74.82	85.34	76.60	6.7B	**95.64**	79.90	**96.76**	96.12	**91.44**	91.97
GNNAVI-GCN	2.6M	90.67	78.82	91.88	92.94	89.20	88.70	16.8M	95.36	82.85	95.50	**96.45**	91.05	**92.24**
GNNAVI-SAGE	5.1M	90.46	**82.68**	92.32	**93.44**	89.28	**89.64**	33.6M	95.30	81.94	94.76	95.96	90.68	91.73

Table 1: Results of different training methods (accuracy). k denotes the number of training examples per class, #Param denotes the number of trainable parameters. The best scores are highlighted with **bold**.

5.1 Overall Performance

Observing the results of GPT2-XL, GNNAVI remarkably rivals ICL, FPFT, and other parameter-efficient baselines. Under the low-data setting of 5 training examples, both GNNAVI-GCN and GNNAVI-SAGE outperform FPFT by over 13%, achieving higher accuracy than other PEFT methods by 0.4% to 21%. Increasing the number of training examples to 200, the average performance of GNNAVI improves to 89.64% and outperforms other baselines.

Similar to GPT2-XL, GNNAVI achieves the best performance with Llama2 among all the baselines. With only 5 training examples, GNNAVI-SAGE achieves 2.77% higher average accuracy than FPFT. Comparing with other PEFT methods, GNNAVI shows higher average accuracy from 1.8% to 35%. And with 200 training examples, GNNAVI-GCN achieves 92.24% average accuracy, outperforming FPFT, Prefix-tuning, Adapter, and LoRA.

5.2 Efficiency Analysis

	SST-2	EmoC	TREC	Amazon	Agnews
GPT2-XL	4.7×	6.3×	4.1×	3.9×	3.4×
Llama2	4.3×	2.4×	1.6×	1.4×	1.2×

Table 2: The ratio by which the training process is accelerated for one training epoch for GNNAVI-GCN compared to FPFT.

GNNAVI significantly reduces the number of trainable parameters compared to the baselines for both GPT2-XL and Llama2. GNNAVI-GCN for GPT2-XL achieves the highest average accuracy with 5 training examples containing only 2.5 million trainable parameters, which is 615 times smaller than FPFT, six times smaller than Adapter, twice smaller than Prefix, and similar to LoRA. As for Llama2, GNNAVI saves over 6.6 billion trainable parameters compared to FPFT and achieves better results. GNNAVI-GCN also updates fewer parameters than Prefix and Adapter. Although LoRA contains fewer trainable parameters than GNNAVI-GCN in Llama2, the performance of LoRA cannot compete with GNNAVI-GCN and GNNAVI-SAGE. Table 2 shows that by saving a significant amount of training parameters, GNNAVI-GCN speeds up the training process by a factor of up to 6 compared to FPFT.

5.3 Influence of Training Examples

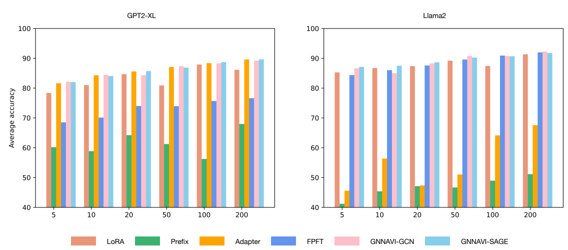
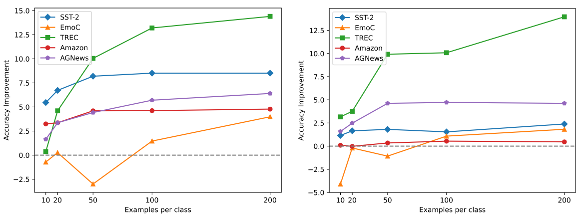


Figure 3: Results of average accuracy with different number of training examples. The x-axis denotes the number of training examples per class.

Adding more training examples improves the accuracy for most baselines and GNNAVI. As depicted in Figure 3, GNNAVI consistently outperforms other methods as the number of training examples increases. While other methods also show improvement with more training examples, the extent of improvement is not as consistent as for GNNAVI, particularly for Prefix and Adapter.

Figure 4 shows the performance of GNNAVI for the different tasks as a function of the number of training examples. We observe that the effect of adding training examples is similar for both GPT2-XL and Llama2. Notably, adding more training examples yields significant improvements, especially in low-data settings (e.g. with 10, 20, and 50 training examples) where GNNAVI shows a substantial improvement, except for EmoC. However, the significance diminishes when more than 50 training examples are provided, the improvement is not as pronounced here as in low-data settings.



(a) GPT2-XL (b) Llama2

Figure 4: The improvement gained by adding training examples for GNNAVI-SAGE, compared to using 5 training examples per class.

6 Ablation Study

In §6.1 of this section, we delve into the influence of the position where the GNN layer is inserted in the LLM. In §6.2, we investigate the effects of removing one of the information flow paths on performance. All of these studies are conducted using GNNAVI-SAGE with 5 training samples per class under the experimental settings outlined in §4.2.

6.1 Position of GNN Layer

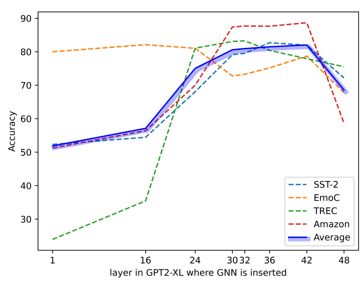


Figure 5: Performance Comparison with GNN inserted at various positions in GPT2-XL.

The position where the GNN layer is inserted significantly impacts the model's performance. Figure 5 illustrates the performance of GNNAVI when the GNN layer is inserted at different locations in GPT2-XL. With the exception of EmoC, all tasks exhibit lower performance when the GNN layer is added in the first 10 layers of GPT2-XL. Performance improves as the GNN is added in deeper layers, reaching peak accuracy around the 44th layer. Subsequently, accuracy declines until the last layer. This trend may stem from the gradual initiation of the information flow process in the early layers of LLM, where the GNN's influence is limited due to insufficient token interaction. Conversely, in the final layers, the information flow process is nearly complete, rendering it too late for the GNN to guide effectively. Despite variations in performance changes across tasks, the average performance suggests that the optimal placement for the GNN layer is between the 38th and 42nd layers for GPT2-XL.

6.2 Removal of Information Flow

We conduct an ablation study to investigate how removing specific information flow paths affects the results while retaining others. In our approach, we connect the label words to their preceding words to aggregate information and to the last token to distribute the information from the label words. These connections are referred to as the aggregation and distribution paths in the ablation study. As illustrated in Figure 6, we remove one path and retain another.

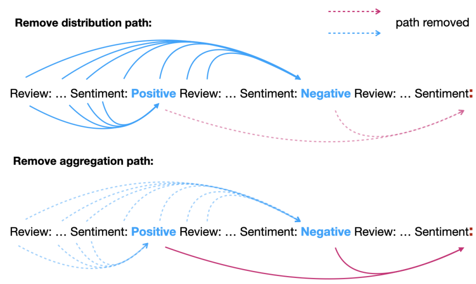


Figure 6: Visualisation of the ablation study on the removal of information flow.

	SST-2	EmoC	TREC	Amazon	Agnews	Average
GNNAVI-SAGE	81.95	78.70	77.92	88.66	82.88	82.02
-aggregation	-0.07	-1.10	-0.68	+0.56	-0.08	-0.27
-distribution	+3.07	-12.88	-2.44	+1.64	-1.44	-2.41

Table 3: Ablation Study: Removal of information flow. The name indicates the removed path.

As shown in Table 3, both the aggregation and distribution paths contribute significantly to the performance. Removing either of them results in a decrease in the average accuracy across the five tasks. Except for the two binary classification tasks SST-2 and Amazon, removing the distribution path causes a greater drop in performance. Based on these results, we conclude that the distribution path plays a more significant role in the information flow process, especially for tasks with more than two labels.

7 Further Discussion: Information Flow

While the attention mechanism in LLM offers an information flow perspective for interpreting the model's working mechanism (Wang et al., 2023), it treats the input text as a fully connected graph. In contrast, GNNAVI explicitly connects the context tokens to the label tokens for information aggregation and the label tokens to the final token for information distribution. Thereby, the correct information flow is hardwired into the GNN. There is no need to learn it by adjusting the attention weights. To further investigate the differences in information flow between FPFT and GNNAVI, we utilize the saliency technique (Simonyan et al., 2013) for interpretation. Following the approach of Wang et al. (2023), we compute the saliency score for each element of the attention matrix using a Taylor expansion (Michel et al., 2019):

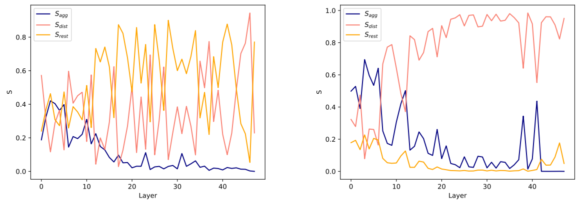
$$I_l = \sum_h \left| A_{h,l}^{\top} \frac{\partial L(x)}{\partial A_{h,l}} \right|, \tag{5}$$

where $A_{h,l}$ represents the attention matrix of the h-th attention head in the l-th layer. x is the input, and $L(x)$ is the loss function. The saliency matrix I_l for the l-th layer is obtained by averaging the values across all attention heads. Each element $I_l(i,j)$ of the matrix denotes the significance of the information flow from the j-th word to the i-th word in the prompt.

We employ three quantitative metrics to assess the information flow: S_{agg} measures the information flow of the aggregation path from previous context words to label words, S_{dist} measures the information distribution from label words to the last token, and S_{rest} accounts for other information flow between remaining words excluding S_{agg} and S_{dist}. The average significance of information flow can be formulated as:

$$S = \frac{\sum_{(i,j)\in C} I_l(i,j)}{|C|}, \tag{6}$$

where C is the total number of token interactions involved.[9]



(a) FPFT (b) GNNAVI

Figure 7: Comparison of information flow between FPFT and GNNAVI for SST-2. Both models are trained with 5 training examples per class.

As depicted in Figure 7, the information flow of GNNAVI appears more stable compared to FPFT. In FPFT, without guided navigation, tokens interact with every preceding word, leading to a trend of confusion between the information flow S_{dist} and S_{rest}. This indicates a struggle to identify the 'right' information for final prediction. Conversely, GNNAVI adheres to the information flow guided by the GNN, resulting in stable curves that depict a consistent information aggregation process, aligning with the findings of Wang et al. (2023). Compared to FPFT, the stable curves affirm that GNNAVI serves as a navigator, ensuring the information flows in predefined directions.

8 Conclusion

In this work, we propose a novel PEFT method, GNNAVI, leveraging GNN to navigate information flow within LLMs. Specifically tailored for prompt-based fine-tuning, GNNAVI significantly reduces the number of trainable parameters by simply adding a GNN layer into LLMs to guide the information flow within the prompt. GNNAVI outperforms FPFT and other PEFT methods across various classification tasks, even with few training examples. Our work offers insights into handling LLMs from a graph perspective and presents a novel application of GNNs in NLP. Future work could explore different token connectivities for GNNs or utilize GNNs to control the information flow in LLMs.

[9] The full formulas of S_{agg}, S_{dist}, and S_{rest} can be found in Appendix D.

Limitation

Although GNNAVI introduces a novel insight for NLP research, there are several limitations in our work. Firstly, GNNAVI is susceptible to the quality of the demonstrations. We find that its performance heavily relies on the selection of demonstrations when only a few training examples are available. However, this issue is alleviated with an increase in the number of training examples. Secondly, while GNNAVI builds upon the information flow of LLMs, it offers a more transparent working mechanism. However, as a black-box model, the working mechanism of the GNN layer is not investigated in this work. Thirdly, we only evaluated the performance of GNNAVI on text classification tasks, other NLP tasks are not explored in this study. We leave these limitations for future work.

Ethics Statement

This study adhered to the ACM Code of Ethics. The datasets employed in our research are publicly accessible, and we utilized them solely for the purpose of evaluating our models. Any potential inaccuracies in the datasets are beyond our responsibility.

References

Elad Ben Zaken, Yoav Goldberg, and Shauli Ravfogel. 2022. BitFit: Simple parameter-efficient fine-tuning for transformer-based masked language-models. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 1–9, Dublin, Ireland. Association for Computational Linguistics.

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. 2020. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901.

Ankush Chatterjee, Kedhar Nath Narahari, Meghana Joshi, and Puneet Agrawal. 2019. SemEval-2019 task 3: EmoContext contextual emotion detection in text. In *Proceedings of the 13th International Workshop on Semantic Evaluation*, pages 39–48, Minneapolis, Minnesota, USA. Association for Computational Linguistics.

Damai Dai, Yutao Sun, Li Dong, Yaru Hao, Shuming Ma, Zhifang Sui, and Furu Wei. 2023. Why can GPT learn in-context? language models secretly perform gradient descent as meta-optimizers. In *Findings of the Association for Computational Linguistics: ACL 2023*, pages 4005–4019, Toronto, Canada. Association for Computational Linguistics.

Ning Ding, Yujia Qin, Guang Yang, Fuchao Wei, Zonghan Yang, Yusheng Su, Shengding Hu, Yulin Chen, Chi-Min Chan, Weize Chen, et al. 2023. Parameter-efficient fine-tuning of large-scale pre-trained language models. *Nature Machine Intelligence*, 5(3):220–235.

Matthias Fey and Jan E. Lenssen. 2019. Fast graph representation learning with PyTorch Geometric. In *ICLR Workshop on Representation Learning on Graphs and Manifolds*.

Tianyu Gao, Adam Fisch, and Danqi Chen. 2021. Making pre-trained language models better few-shot learners. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 3816–3830, Online. Association for Computational Linguistics.

Claire Gardent, Anastasia Shimorina, Shashi Narayan, and Laura Perez-Beltrachini. 2017. The WebNLG challenge: Generating text from RDF data. In *Proceedings of the 10th International Conference on Natural Language Generation*, pages 124–133, Santiago de Compostela, Spain. Association for Computational Linguistics.

Will Hamilton, Zhitao Ying, and Jure Leskovec. 2017. Inductive representation learning on large graphs. *Advances in neural information processing systems*, 30.

Neil Houlsby, Andrei Giurgiu, Stanislaw Jastrzebski, Bruna Morrone, Quentin De Laroussilhe, Andrea Gesmundo, Mona Attariyan, and Sylvain Gelly. 2019. Parameter-efficient transfer learning for NLP. In *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pages 2790–2799. PMLR.

Eduard Hovy, Laurie Gerber, Ulf Hermjakob, Chin-Yew Lin, and Deepak Ravichandran. 2001. Toward semantics-based answer pinpointing. In *Proceedings of the First International Conference on Human Language Technology Research*.

Edward J Hu, yelong shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. 2022. LoRA: Low-rank adaptation of large language models. In *International Conference on Learning Representations*.

Thomas N. Kipf and Max Welling. 2017. Semi-supervised classification with graph convolutional networks. In *International Conference on Learning Representations*.

Dohwan Ko, Ji Lee, Woo-Young Kang, Byungseok Roh, and Hyunwoo Kim. 2023. Large language models are temporal and causal reasoners for video question answering. In *Proceedings of the 2023 Conference* *on Empirical Methods in Natural Language Processing*, pages 4300–4316, Singapore. Association for Computational Linguistics.

Rik Koncel-Kedziorski, Dhanush Bekal, Yi Luan, Mirella Lapata, and Hannaneh Hajishirzi. 2019. Text Generation from Knowledge Graphs with Graph Transformers. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 2284–2293, Minneapolis, Minnesota. Association for Computational Linguistics.

Brian Lester, Rami Al-Rfou, and Noah Constant. 2021. The power of scale for parameter-efficient prompt tuning. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 3045–3059, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.

Junyi Li, Tianyi Tang, Wayne Xin Zhao, Zhicheng Wei, Nicholas Jing Yuan, and Ji-Rong Wen. 2021. Few-shot knowledge graph-to-text generation with pre-trained language models. In *Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021*, pages 1558–1568, Online. Association for Computational Linguistics.

Xiang Lisa Li and Percy Liang. 2021. Prefix-tuning: Optimizing continuous prompts for generation. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 4582–4597, Online. Association for Computational Linguistics.

Xin Li and Dan Roth. 2002. Learning question classifiers. In *COLING 2002: The 19th International Conference on Computational Linguistics*.

Jiachang Liu, Dinghan Shen, Yizhe Zhang, Bill Dolan, Lawrence Carin, and Weizhu Chen. 2022. What makes good in-context examples for GPT-3? In *Proceedings of Deep Learning Inside Out (DeeLIO 2022): The 3rd Workshop on Knowledge Extraction and Integration for Deep Learning Architectures*, pages 100–114, Dublin, Ireland and Online. Association for Computational Linguistics.

Yongkang Liu, Yiqun Zhang, Qian Li, Shi Feng, Daling Wang, Yifei Zhang, and Hinrich Schütze. 2024. Hift: A hierarchical full parameter fine-tuning strategy. *arXiv preprint arXiv:2401.15207*.

Bolei Ma, Ercong Nie, Helmut Schmid, and Hinrich Schütze. 2023. Is prompt-based finetuning always better than vanilla finetuning? insights from cross-lingual language understanding. In *Proceedings of the 18th Conference on Natural Language Processing (KONVENS 2023)*, Ingolstadt, Germany. KONVENS 2023 Organizers.

Bolei Ma, Ercong Nie, Shuzhou Yuan, Helmut Schmid, Färber Michael, Frauke Kreuter, and Hinrich Schütze. 2024. Topro: Token-level prompt decomposition for cross-lingual sequence labeling tasks. In *Proceedings of the 18th Conference of the European Chapter of the Association for Computational Linguistics*, Malta, Malta. Association for Computational Linguistics.

Sadhika Malladi, Tianyu Gao, Eshaan Nichani, Alex Damian, Jason D. Lee, Danqi Chen, and Sanjeev Arora. 2023. Fine-tuning language models with just forward passes. In *Thirty-seventh Conference on Neural Information Processing Systems*.

Sourab Mangrulkar, Sylvain Gugger, Lysandre Debut, Younes Belkada, Sayak Paul, and Benjamin Bossan. 2022. Peft: State-of-the-art parameter-efficient fine-tuning methods. `https://github.com/huggingface/peft`.

Julian McAuley and Jure Leskovec. 2013. Hidden factors and hidden topics: understanding rating dimensions with review text. In *Proceedings of the 7th ACM conference on Recommender systems*, pages 165–172.

Dheeraj Mekala, Jason Wolfe, and Subhro Roy. 2023. ZEROTOP: Zero-shot task-oriented semantic parsing using large language models. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 5792–5799, Singapore. Association for Computational Linguistics.

Paul Michel, Omer Levy, and Graham Neubig. 2019. Are sixteen heads really better than one? In *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc.

Sewon Min, Xinxi Lyu, Ari Holtzman, Mikel Artetxe, Mike Lewis, Hannaneh Hajishirzi, and Luke Zettlemoyer. 2022. Rethinking the role of demonstrations: What makes in-context learning work? In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pages 11048–11064, Abu Dhabi, United Arab Emirates. Association for Computational Linguistics.

Ercong Nie, Sheng Liang, Helmut Schmid, and Hinrich Schütze. 2023a. Cross-lingual retrieval augmented prompt for low-resource languages. In *Findings of the Association for Computational Linguistics: ACL 2023*, pages 8320–8340, Toronto, Canada. Association for Computational Linguistics.

Ercong Nie, Helmut Schmid, and Hinrich Schuetze. 2023b. Unleashing the multilingual encoder potential: Boosting zero-shot performance via probability calibration. In *Findings of the Association for Computational Linguistics: EMNLP 2023*, pages 15774–15782, Singapore. Association for Computational Linguistics.

Arkil Patel, Satwik Bhattamishra, Siva Reddy, and Dzmitry Bahdanau. 2023. MAGNIFICo: Evaluating the in-context learning ability of large language models to generalize to novel interpretations. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 2167–2189, Singapore. Association for Computational Linguistics.

Jonas Pfeiffer, Andreas Rücklé, Clifton Poth, Aishwarya Kamath, Ivan Vulić, Sebastian Ruder, Kyunghyun Cho, and Iryna Gurevych. 2020. Adapterhub: A framework for adapting transformers. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP 2020): Systems Demonstrations*, pages 46–54, Online. Association for Computational Linguistics.

Clifton Poth, Hannah Sterz, Indraneil Paul, Sukannya Purkayastha, Leon Engländer, Timo Imhof, Ivan Vulić, Sebastian Ruder, Iryna Gurevych, and Jonas Pfeiffer. 2023. Adapters: A unified library for parameter-efficient and modular transfer learning. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 149–160, Singapore. Association for Computational Linguistics.

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, Ilya Sutskever, et al. 2019. Language models are unsupervised multitask learners. *OpenAI blog*, 1(8):9.

Leonardo F. R. Ribeiro, Yue Zhang, and Iryna Gurevych. 2021. Structural adapters in pretrained language models for AMR-to-Text generation. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 4269–4282, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.

Timo Schick and Hinrich Schütze. 2021a. Exploiting cloze-questions for few-shot text classification and natural language inference. In *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: Main Volume*, pages 255–269, Online. Association for Computational Linguistics.

Timo Schick and Hinrich Schütze. 2021b. It's not just size that matters: Small language models are also few-shot learners. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 2339–2352, Online. Association for Computational Linguistics.

Karen Simonyan, Andrea Vedaldi, and Andrew Zisserman. 2013. Deep inside convolutional networks: Visualising image classification models and saliency maps. *CoRR*, abs/1312.6034.

Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, Christopher D. Manning, Andrew Ng, and Christopher Potts. 2013. Recursive deep models for semantic compositionality over a sentiment treebank. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 1631–1642, Seattle, Washington, USA. Association for Computational Linguistics.

Xiaofei Sun, Xiaoya Li, Jiwei Li, Fei Wu, Shangwei Guo, Tianwei Zhang, and Guoyin Wang. 2023. Text classification via large language models. In *Findings of the Association for Computational Linguistics: EMNLP 2023*, pages 8990–9005, Singapore. Association for Computational Linguistics.

Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, et al. 2023. Llama 2: Open foundation and fine-tuned chat models. *arXiv preprint arXiv:2307.09288*.

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

Zhen Wan, Fei Cheng, Zhuoyuan Mao, Qianying Liu, Haiyue Song, Jiwei Li, and Sadao Kurohashi. 2023. GPT-RE: In-context learning for relation extraction using large language models. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 3534–3547, Singapore. Association for Computational Linguistics.

Lean Wang, Lei Li, Damai Dai, Deli Chen, Hao Zhou, Fandong Meng, Jie Zhou, and Xu Sun. 2023. Label words are anchors: An information flow perspective for understanding in-context learning. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 9840–9855, Singapore. Association for Computational Linguistics.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, et al. 2019. Huggingface's transformers: State-of-the-art natural language processing. *arXiv preprint arXiv:1910.03771*.

Sang Michael Xie, Aditi Raghunathan, Percy Liang, and Tengyu Ma. 2022. An explanation of in-context learning as implicit bayesian inference. In *International Conference on Learning Representations*.

Kang Min Yoo, Junyeob Kim, Hyuhng Joon Kim, Hyunsoo Cho, Hwiyeol Jo, Sang-Woo Lee, Sang-goo Lee, and Taeuk Kim. 2022. Ground-truth labels matter: A deeper look into input-label demonstrations. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pages 2422–2437, Abu Dhabi, United Arab Emirates. Association for Computational Linguistics.

Renrui Zhang, Jiaming Han, Aojun Zhou, Xiangfei Hu, Shilin Yan, Pan Lu, Hongsheng Li, Peng Gao, and Yu Qiao. 2023. Llama-adapter: Efficient fine-tuning of language models with zero-init attention. *arXiv preprint arXiv:2303.16199*.

X Zhang, A Bosselut, M Yasunaga, H Ren, P Liang, C Manning, and J Leskovec. 2022. Greaselm: Graph reasoning enhanced language models for question answering. In *International Conference on Representation Learning (ICLR)*.

Xiang Zhang, Junbo Zhao, and Yann LeCun. 2015. Character-level convolutional networks for text classification. In *Advances in Neural Information Processing Systems*, volume 28. Curran Associates, Inc.

Zihao Zhao, Eric Wallace, Shi Feng, Dan Klein, and Sameer Singh. 2021a. Calibrate before use: Improving few-shot performance of language models. In *International Conference on Machine Learning*, pages 12697–12706. PMLR.

Zihao Zhao, Eric Wallace, Shi Feng, Dan Klein, and Sameer Singh. 2021b. Calibrate before use: Improving few-shot performance of language models. In *Proceedings of the 38th International Conference on Machine Learning*, volume 139 of *Proceedings of Machine Learning Research*, pages 12697–12706. PMLR.

A Hyperparameters

We present the hyperparameters for GNNAVI and other baselines in Table 4. The models were trained using NVIDIA A100-SXM4-40GB GPUs. Due to limited resources, the batch size was set to 1, and full parameter fine-tuning of Llama2 was implemented using 8 bits. We observed that for Llama2, GNNAVI and other PEFT methods were sensitive to the selection of prompts with very few training samples, and thus could not achieve optimal performance. To address this, we replaced these results by using another random seed to change the demonstrations in the prompt.

B Demonstration Templates and Label Words

The templates for the prompt are presented in Table 5. $[S]$ denotes the demonstration selected to form the prompt, $[L]$ represents the label word of the demonstration, and $[S_i]$ denotes the sample to be predicted.

C Full Results

Due to space constraints, the complete results are provided in Table 6. Each value in the table represents the average accuracy over five experiments conducted with different random seeds.

D Formula of Saliency Score

We utilize l to denote the label words, such as 'Positive' and 'Negative', while f represents the final token, such as ':'. Additionally, t denotes other tokens excluding label and final tokens.

S_{agg} calculates the mean significance of information flow from the previous context words to label words:

$$S_{agg} = \frac{\sum_{(i,j)\in C_{tl}} I_l(i,j)}{|C_{tl}|}, \quad C_{tl} = \{(l_k, j) : k \in [1, C], j < l_k\}. \tag{7}$$

S_{dist} calculates the mean significance of information flow from the label words to the final token:

$$S_{dist} = \frac{\sum_{(i,j)\in C_{lf}} I_l(i,j)}{|C_{lf}|}, \quad C_{lf} = \{(f, l_k) : k \in [1, C]\}. \tag{8}$$

S_{rest} calculates the mean significance of information flow among the rest words, excluding S_{agg} and S_{dist}:

$$S_{rest} = \frac{\sum_{(i,j)\in C_{tt}} I_l(i,j)}{|C_{tt}|}, \quad C_{tt} = \{(i, j) : j < i\} - C_{tl} - C_{lf}. \tag{9}$$

Hyperparameter	GNNAVI	Prefix	Adapter	LoRA	FPFT
learning rate	1e-2	1e-2	5e-5	5e-4	5e-5
optimizer	Adam	Adam	AdamW	AdamW	AdamW
epochs	50	50	50	50	50
early Stop	15	15	15	15	15
random seed	[0, 42, 312, 411, 412, 421, 520, 1218]				
virtual tokens	-	40(GPT2), 150(Llama2)	-		

Table 4: Hyperparameters for GNNAVI and baselines.

Task	Template	Label Words
SST-2	Review: $[S]$ Sentiment: $[L]$ Review: $[S_i]$ Sentiment:	Positive, Negative
EmoC	Dialogue: $[S]$ Emotion: $[L]$ Dialogue:$[S_i]$ Emotion:	Happy, Sad, Angry, Others
TREC	Question: $[S]$ Answer Type: $[L]$ Question: $[S_i]$ Answer Type:	Abbreviation, Entity, Description, Person, Location, Number
Amazon	Review: $[S]$ Sentiment: $[L]$ Review: $[S_i]$ Sentiment:	Positive, Negative
AGNews	Article: $[S]$ Answer: $[L]$ Article: $[S_i]$ Answer:	World, Sports, Business, Technology

Table 5: Template for prompt.

k	Method	#Param	SST-2	EmoC	TREC	Amazon	AGNews	Average	#Param	SST-2	EmoC	TREC	Amazon	AGNews	Average
		GPT2-XL							Llama2						
0	ICL	-	55.44	6.48	54.68	53.32	72.12	48.41	-	67.55	9.60	70.36	94.98	84.14	65.33
5	ICL	-	63.17	6.30	57.68	53.67	50.43	46.25	-	86.93	20.18	45.72	92.30	80.16	65.06
	LoRA	2.5M	91.98	50.60	75.20	88.80	**85.20**	78.36	4.2M	**95.42**	64.20	**88.40**	91.80	86.60	85.28
	Prefix	6.1M	59.13	73.46	32.92	60.00	75.40	60.18	39.3M	50.96	58.56	21.36	49.36	25.78	41.20
	Adapter	15.4M	79.82	76.00	**79.60**	**91.45**	81.25	81.62	198M	50.92	**84.05**	18.80	49.45	24.80	45.60
	FPFT	1.6B	62.13	61.30	65.28	73.00	80.82	68.51	6.7B	94.63	61.92	81.72	**95.86**	**87.58**	84.34
	GNNAVI-GCN	2.6M	**84.31**	75.48	76.72	90.90	83.16	**82.11**	16.8M	94.56	78.30	83.2	94.00	86.25	86.63
	GNNAVI-SAGE	5.1M	81.95	**78.70**	77.92	88.66	82.88	82.02	33.6M	92.91	80.12	80.80	95.66	86.06	**87.11**
10	LoRA	2.5M	**88.08**	53.20	86.40	90.60	86.80	81.02	4.2M	**94.73**	63.00	**92.80**	92.60	**90.40**	86.71
	Prefix	6.1M	51.08	77.58	38.16	65.94	61.48	58.85	39.3M	50.80	**76.98**	21.20	51.42	26.44	45.37
	Adapter	15.4M	86.70	70.65	**87.40**	90.60	86.15	84.30	198M	50.92	85.60	41.00	52.20	52.15	56.37
	FPFT	1.6B	69.01	71.90	52.48	75.82	81.34	70.11	6.7B	92.91	68.06	84.24	96.22	88.64	86.01
	GNNAVI-GCN	2.6M	84.63	**83.97**	74.80	91.57	**87.00**	**84.39**	16.8M	91.86	70.75	82.40	**96.35**	89.30	84.99
	GNNAVI-SAGE	5.1M	87.41	77.98	78.28	**91.90**	84.52	84.02	33.6M	94.06	76.02	83.96	95.76	87.64	**87.49**
20	LoRA	2.5M	85.09	69.00	86.00	**94.00**	**89.20**	84.66	4.2M	95.64	70.80	83.60	**96.20**	**90.60**	87.37
	Prefix	6.1M	56.68	**83.28**	39.20	61.22	80.62	64.20	39.3M	50.57	78.70	27.92	52.08	26.30	47.11
	Adapter	15.4M	88.42	74.65	**89.00**	89.45	86.50	85.60	198M	50.92	**85.80**	18.80	56.40	24.80	47.34
	FPFT	1.6B	73.10	70.72	68.36	77.40	80.44	74.00	6.7B	**95.32**	69.96	**88.08**	95.52	89.04	87.58
	GNNAVI-GCN	2.6M	86.93	76.23	79.67	92.70	86.07	84.32	16.8M	94.78	75.25	84.80	96.00	89.30	88.27
	GNNAVI-SAGE	5.1M	**88.67**	78.96	82.52	92.02	86.24	**85.68**	33.6M	94.56	79.92	84.56	95.64	88.54	**88.64**
50	LoRA	2.5M	89.45	74.80	54.80	**93.60**	**91.80**	80.89	4.2M	93.12	72.40	**94.40**	95.40	**91.60**	89.20
	Prefix	6.1M	50.90	**79.78**	26.72	74.42	74.40	61.24	39.3M	50.48	76.22	28.08	50.96	27.60	46.67
	Adapter	15.4M	86.75	77.85	**91.60**	90.50	88.75	87.09	198M	50.92	76.80	44.40	49.45	33.45	51.00
	FPFT	1.6B	70.60	71.68	76.40	67.84	83.10	73.92	6.7B	**95.46**	74.20	91.92	95.82	90.48	89.58
	GNNAVI-GCN	2.6M	89.49	79.50	87.93	92.40	87.43	**87.35**	16.8M	95.07	**83.05**	88.70	95.85	90.80	**90.81**
	GNNAVI-SAGE	5.1M	**90.14**	75.70	87.96	93.26	87.30	86.87	33.6M	94.72	79.04	90.72	**96.00**	90.68	90.23
100	LoRA	2.5M	89.22	**84.00**	88.40	93.20	84.80	87.92	4.2M	92.66	**86.60**	94.80	95.40	67.60	87.41
	Prefix	6.1M	56.26	72.28	32.04	69.48	51.18	56.25	39.3M	49.11	76.20	40.28	52.38	26.82	48.96
	Adapter	15.4M	86.93	82.85	**92.00**	92.40	87.60	88.36	198M	58.83	84.95	84.00	68.10	24.80	64.14
	FPFT	1.6B	72.82	73.42	68.56	78.74	84.86	75.68	6.7B	**95.07**	76.06	**96.20**	**96.20**	**91.04**	**90.91**
	GNNAVI-GCN	2.6M	89.41	81.30	90.20	92.67	87.97	88.31	16.8M	94.27	81.20	91.60	96.00	90.80	90.77
	GNNAVI-SAGE	5.1M	**90.46**	80.16	91.12	**93.28**	**88.58**	**88.72**	33.6M	94.45	81.20	90.88	96.08	90.78	90.68
200	LoRA	2.5M	**90.83**	80.80	90.80	82.00	86.20	86.13	4.2M	91.29	**86.80**	93.60	95.80	90.40	91.32
	Prefix	6.1M	50.92	80.18	69.80	59.80	79.08	67.96	39.3M	48.35	81.72	45.68	52.28	27.54	51.11
	Adapter	15.4M	88.65	80.70	**96.60**	92.30	**89.80**	89.61	198M	50.92	85.05	88.20	49.45	81.50	67.57
	FPFT	1.6B	68.97	73.70	80.16	74.82	85.34	76.60	6.7B	**95.64**	79.90	**96.76**	96.12	**91.44**	91.97
	GNNAVI-GCN	2.6M	90.67	78.82	91.88	92.94	89.20	88.70	16.8M	95.36	82.85	95.50	**96.45**	91.05	**92.24**
	GNNAVI-SAGE	5.1M	90.46	**82.68**	92.32	**93.44**	89.28	**89.64**	33.6M	95.30	81.94	94.76	95.96	90.68	91.73

Table 6: Results with different training methods (accuracy). k denotes the number of training examples per class. #Param denotes the number of trainable parameters. The best scores under the same circumstances of training examples are highlighted with **bold**.