Task Knowledge Injection via Interpolations and Reinstatement for Large Language Model Generalization

Yukun Zhao[1] **Lingyong Yan**[1] **Zhenyang Li**[1] **Shuaiqiang Wang**[1]
Zhumin Chen[2] **Zhaochun Ren**[3*] **Dawei Yin**[1*]
[1]Baidu Inc., Beijing, China [2]Shandong University, Jinan, China
[3]Leiden University, Leiden, The Netherlands
{zhaoyukun02,yanlingyong,lizhenyang02,wangshuaiqiang}@baidu.com
chenzhumin@sdu.edu.cn,z.ren@liacs.leidenuniv.nl,yindawei@acm.org

Abstract

Large language models have shown tremendous potential across various NLP tasks, and instruction tuning has been widely adopted to elicit their superior performance. However, instruction tuning may overly tailor the models to task-specific formats, potentially compromising their generalization on unseen tasks. We attribute the issue to the spurious correlations learned between inputs and targets. We propose explicit task knowledge injection to mitigate these shortcuts with latent task adaptation and knowledge reinstatement. Latent tasks serve as interpolations between new tasks and facilitate knowledge sharing with joint adaptation enabling the model to build task knowledge more smoothly. Knowledge reinstatement helps optimize building new knowledge with prior knowledge. Specifically, we retrieve input-relevant latent tasks and jointly learn the task and the relevant latent tasks. Moreover, we prompt the model to recall the forms of inputs corresponding to the target and build the task knowledge through the reinstatement of prior knowledge while learning the new task. We conduct extensive experiments on state-of-the-art large language models including Llama3.1-8B and Vicuna-13B across 1000+ instruction-following tasks to demonstrate the effectiveness of our method. The results demonstrate our method improves generalization on both in-domain and out-of-domain unseen tasks.

1 Introduction

Pre-trained large language models (Brown et al., 2020; Achiam et al., 2023; Touvron et al., 2023; OpenAI, 2024; Research, 2024) have become a cornerstone in many NLP tasks due to their impressive generalization capabilities (Von Oswald et al., 2023; Minaee et al., 2024; Lotfi et al., 2024). These models can be prompted with arbitrary demonstrations to accomplish various tasks. To further enhance their performance on downstream tasks, instruction tuning is widely adopted (Ouyang et al., 2022; Wei et al., 2021; Chung et al., 2024). Despite its success, instruction tuning may inadvertently overfit the model to specific task formats, thereby impairing its ability to generalize to new, unseen tasks (Wang et al., 2022a; Yang et al., 2024b).

Current techniques that help improve instruction tuning generalization can be categorized into four groups: (1) Data-based methods (Yang et al., 2024c; Wang et al., 2022b; Xu et al., 2023; Burns et al., 2023; Yang et al., 2024a; Zhou et al., 2024; Zhao et al., 2024; Chung et al., 2024) incorporate broad-coverage tasks, contextual demonstrations, chain-of-thought demonstrations, similar task augmentation, more complex and high-quality data into the training data to improve generalization, which requires carefully curated examples. (2) Parameter-Efficient Fine-tuning (PEFT) methods (Lester et al., 2021; Li and Liang, 2021; Zheng et al., 2024) such as LoRA (Hu et al., 2021) only utilize a small number of (extra) parameters to enable the adaptation on downstream tasks, which help improve generalization to some extent due to the implicit regularization effect. (3) Adversarial methods (Miyato et al., 2018; Jiang et al., 2019; Pan et al., 2022; Ni et al., 2024) make the model learn to handle adversarial attacks such as embedding space perturbation, which improves generalization in certain scenarios. (4) Regularization-based methods (Kirkpatrick et al., 2017; Aljundi et al., 2018; Schwarz et al., 2018; Ritter et al., 2018; Foret et al., 2021; Wang et al., 2024) employ parameter regularization and gradient regularization to penalize large changes between model parameters and conflicted gradients. The parameter regularization may lead to under- or over-constraint issues (Gu et al., 2022; Liang et al., 2024). In contrast with these methods, we aim to propose a data-efficient method to improve generalization in instruction tuning without additional data or data engineering.

* Co-corresponding authors.

Findings of the Association for Computational Linguistics: ACL 2025, pages 15070–15080

Unlike previous studies, we attribute the issue of generalization on unseen tasks to learned spurious correlations between inputs and targets (McMilin, 2022; Zhou et al., 2023; Zhao et al., 2024). Specifically, a task can have multiple verbalizations, and if a model is trained with only a few instructions, the sequence-to-sequence training paradigm may cause the model to learn shortcuts like certain words or sentence structures. We address this by explicitly injecting relevant knowledge related to current tasks to help the model build knowledge more effectively about the tasks rather than just superficial patterns.

To this end, we propose task knowledge injection with latent task adaptation and knowledge reinstatement. Our method first retrieves relevant latent tasks by comparing the task input with latent task keys. Latent tasks are interpolations between the tasks represented as tuples of keys (vectors) and corresponding knowledge (compiled as weight increments), which are initialized randomly. Then we make the model jointly learn the current task and relevant latent tasks by re-parameterizing the model parameters and weight increments. This process facilitates knowledge sharing and enables the model to build task knowledge more smoothly. Moreover, our method prompts the model to recall which forms of inputs in the existing model memory might generate the target. We make the model learn through the reinstatement of the prior knowledge during the current task learning, which helps build more comprehensive knowledge about the task.

We conduct extensive experiments on publicly available large language models, Vicuna-13B and Llama3.1-8B across 1000+ instruction-following tasks. The experimental results demonstrate that our approach improves generalization on both in-domain and out-of-domain unseen tasks.

The contributions of our paper are summarized as follows:

- We propose a novel method—task knowledge injection—to enhance generalization in instruction tuning, which is a data-efficient method.

- We propose latent task adaptation and knowledge reinstatement to establish correlations both among new tasks and between new tasks and prior knowledge, aiding the model to build task knowledge more effectively.

- We conducted extensive experiments to demonstrate the effectiveness of our method and each component.

2 Related Work

Large Language Models Generalization. Pre-trained large language models (Brown et al., 2020; Achiam et al., 2023; Touvron et al., 2023; OpenAI, 2024; Research, 2024) have demonstrated great generalization capabilities (Von Oswald et al., 2023; Minaee et al., 2024; Lotfi et al., 2024). The success can be attributed to the self-attention mechanism, large-scale parameters, and pre-training on web-scale data corpora (Jiang et al., 2024; Harun et al., 2024). Their performance on downstream tasks is achieved through in-context learning (Min et al., 2022; Brown et al., 2020). To perform better in downstream tasks, many post-training techniques like supervised fine-tuning (Ouyang et al., 2022; Wei et al., 2021; Chung et al., 2024), PPO (Schulman et al., 2017), and DPO (Rafailov et al., 2024), have been developed. However, recent studies (Wang et al., 2022a; Kumar et al., 2022; Yang et al., 2024b) point out that fine-tuning may overly tailor the model to specific tasks and formats, potentially compromising its generalization to other new tasks. In this paper, we improve large language model generalization in the fine-tuning stage.

Fine-tuning Methods for Improved Generalization. We compile the methods related to instruction-tuning, robustness, generalization, and continual learning into (1) data-based, (2) Parameter-Efficient Fine-Tuning (PEFT), (3) adversarial training, (4) regularization-based methods. (1) Data-based methods (Yang et al., 2024c; Wang et al., 2022b; Xu et al., 2023; Burns et al., 2023; Yang et al., 2024a; Zhou et al., 2024; Chung et al., 2024) incorporate broad-coverage tasks, contextual demonstrations, chain-of-thought data, synthetic data, instruction augmentation, and more complex and high-quality examples to fine-tune the model to improve the generalization. They require carefully curated examples and adjustments for the data distribution. (2) PEFT (Hu et al., 2021; Lester et al., 2021; Li and Liang, 2021; Zheng et al., 2024) utilizes only a small number of (extra) parameters to enable the adaptation to downstream tasks and can also help to extrapolate in the unseen data in some extent. (3) Adversarial training (Miyato et al., 2018; Jiang et al., 2019; Pan et al., 2022) learns to handle adversarial attacks potentially improving generalization in certain scenarios. Altinisik et al. (2023) and Zhu et al. (2019) train with embedding space perturbation and find it improves encoder-based model generalization on classification tasks. Gan et al. (2020) and Xhonneux et al. (2024) perform adversarial training in the embedding space on generative models. PACE (Ni et al., 2024) regularizes the consistent behaviors between the fine-tuned model with its perturbed version. (4) Parameter regularization (Kirkpatrick et al., 2017; Aljundi et al., 2018; Schwarz et al., 2018; Ritter et al., 2018) discourages large changes between the model parameters by adding a penalty term in the learning objective. However, this may lead to under- or over-constraint issues(Gu et al., 2022; Liang et al., 2024) due to the complex correlation between the capability of a model and its parameters. PAC-Bayes (Li and Zhang, 2021) is a layer-wise regularization method that constrains the distance traveled in each layer during fine-tuning. Regularization in the optimizers (Foret et al., 2021; Wang et al., 2024) accelerates the convergence towards flatter minima. In contrast with the previous methods, we propose a data-efficient method to improve generalization in instruction tuning without collecting or constructing additional data or referring to any other models.

3 Problem Formulation and Motivation

In this paper, we aim to fine-tune existing models to continually improve their performance on downstream tasks. Given a set of tasks $T = \{T_1, ..., T_k\}$ along with their training examples $S^{(t)} = (X^{(t)}, Y^{(t)})$ and an existing model $f_\Theta(x)$, the goal is to learn a new mapping function $g_{\Theta'(x)}$ where the parameters are initialized from Θ that generalizes better on similar downstream tasks $T^{'} = \{T_{k+1}, ..., T_m\}$. We improve the generalization of instruction tuning on the provided data.

Instruction tuning may cause the model to learn spurious correlations between inputs and targets under the sequence-to-sequence training paradigm (Wang et al., 2022a; Yang et al., 2024b), resulting in limited performance. Inspired by (Santoro et al., 2016; van Kesteren et al., 2020), which suggests that congruency in reinstatement and memory augmentation facilitates the assimilation and construction of new knowledge, we propose task knowledge injection with latent task adaptation and knowledge reinstatement. Latent tasks serve as interpolations between new tasks, allowing the model to learn new knowledge more smoothly. Joint adaptation with relevant latent tasks that share common knowledge provides useful inductive biases, helping the model build a more comprehensive understanding of the task. Fine-tuning often leads to knowledge drift, where previously learned information is overwritten. By reinstating prior knowledge while learning new tasks, the model preserves critical learned representations and strengthens its understanding of the new task in a way that aligns with its existing knowledge base.

4 Task Knowledge Injection

In this section, we introduce our task knowledge injection method. We introduce an overview first in subsection 4.1. We introduce latent task adaptation in subsection 4.2 and then introduce knowledge reinstatement in subsection 4.3.

4.1 Overview

The task knowledge injection is a new training method that includes two components, and predictions are made directly using the newly learned model parameters during inference. The first component is input-guided knowledge injection—latent task adaptation (LTA) and the second one is target-guided knowledge injection—knowledge reinstatement (KR), as shown in Figure 1. LTA first retrieves relevant latent tasks by comparing the input representations of a task with the keys of latent tasks and then makes the model jointly learn the new task and the relevant latent tasks. This joint task learning mechanism helps the model acquire more shared knowledge among similar tasks. KR prompts the model to recall which forms of inputs might generate the target, and makes the model reinstate the knowledge while learning the current task. We combine the two objectives as our final learning objective.

4.2 Latent Task Adaptation

In this paper, we aggregate training examples for tasks $T = \{T_1, ..., T_k\}$ together as $S = (X, Y)$, making our method a task-agnostic method. The learning objective is:

$$L = -\sum_{(x_i, y_i) \in S} CE(g_{\Theta'}(x_i), y_i) \qquad (1)$$

where CE denotes the cross-entropy loss.

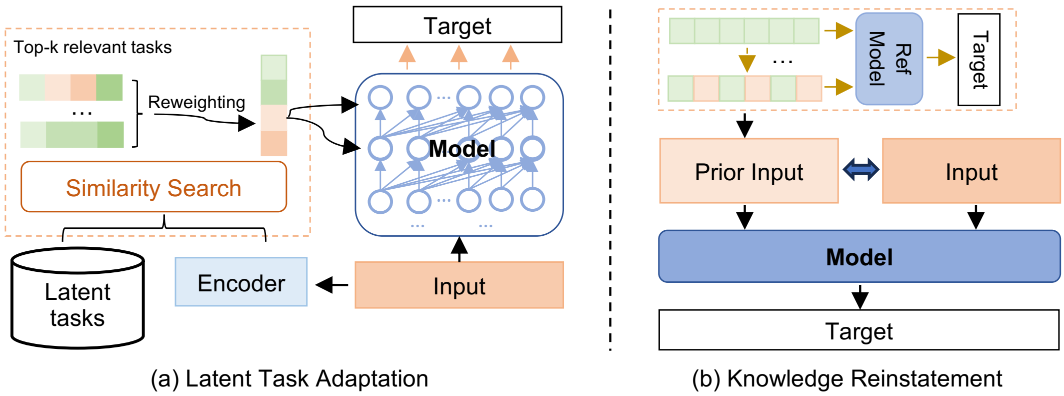

Figure 1: The overview of our task knowledge injection training method. It consists of two modules that are linearly combined to update the model parameters. Component (a) is input-guided knowledge injection, which introduces latent shared knowledge via similar input searching. Component (b) is target-guided knowledge injection, which encourages the model to reinstate knowledge relevant to the target.

For learning the downstream tasks, the model parameters are updated through:

$$\theta^{'} = \arg\min_{\theta^{'}} L, \theta^{'} = \theta + \Delta\theta \tag{2}$$

The corresponding parameter updates $\Delta\theta$ are learned from each example. We propose exploiting relevant latent tasks as interpolation knowledge to help build the task knowledge smoothly. The relevant latent tasks are those sharing similar input representations with actual examples. Thus, we use the original examples along with their relevant latent examples $\{(e_1, \Delta\theta_1), ..., (e_k, \Delta\theta_k)\}$ to update the model parameters. Each latent task is a tuple $(e_i, \Delta\theta_i)$, where e_i is the encoding of the latent task which shares similar semantics with the actual task and $\Delta\theta_i$ is the corresponding weight increment.

Latent Task Retrieval. Firstly, we initialize M latent tasks and then retrieve a small number of relevant latent tasks for each actual example for joint learning. Followed by (Lopez-Paz and Ranzato, 2017; Peng et al., 2024), we randomly initialize M orthogonal vectors $E = (e_1, ..., e_M), e_i \in \mathbf{R}^c$ as keys for the latent tasks. c is the embedding dimension of each key. Any two keys in the set are orthogonal, which helps the latent tasks cover as many task types as possible.

To initialize the corresponding M weight increments and inspired by LORA (Hu et al., 2021), we initialize each $A_i \in \mathbf{R}^{r\times k}$ and $B_i \in \mathbf{R}^{d\times r}$ with zero and normal distribution where r is much less than d helping to reduce the memory cost. Thus each weight increment $\Delta\theta_i$ is calculated as:

$$\Delta\theta_i = B_i \times A_i \tag{3}$$

For each actual training example $(x_j, y_j) \in S$, we employ RoBERTa (Liu, 2019) to encode the input as the query $p_i \in \mathbf{R}^c$. We then use KNN algorithm to retrieve top k most similar latent tasks $\{(e_1, \Delta\theta_1), ..., (e_k, \Delta\theta_k)\}$ by comparing the cosine similarity of the query p_i with the key set E where $k << M$. The weight increments are updated during the learning process while the keys are frozen.

To improve retrieval efficiency, we randomly partition the latent tasks (keys and weight increments) into C groups, with each group containing Q keys. In our setting, all tasks are mixed together. We randomly partition the training samples into C groups regardless of the task types corresponding to the partitioned latent tasks. This helps our method apply to an arbitrary number of tasks.

Joint Task Learning. We incorporate the retrieved relevant knowledge (tasks) to jointly learn the training objective as shown in Equation 1. Specifically, we have a training example (x_i, y_i) coupled with the k most relevant weight increments $\{\Delta\theta_1, ..., \Delta\theta_k\}$. The model parameter is updated and formulated as follows.

$$\Theta^{'} = \Theta + \Delta\theta_i = \Theta + \frac{\sum_{j=1}^{k} \mathbf{D}(p_i, e_j)\Delta\theta_j}{\sum_{j=1}^{k} \mathbf{D}(p_i, e_j)} \tag{4}$$

where $\mathbf{D}$ indicates the distance between the current example and the latent task, which is inferred from their cosine similarity. We use the updated parameter $\Theta^{'}$ to learn the targets where the $\Delta\theta_i$ serves as priors.

The weight increment $\Delta\theta_j$ is updated multiple times when it is chosen as one of the k most relevant examples with the actual examples. The up-

Algorithm 1 Latent Task Adaptation

```
Input: S = (X, Y), f_Θ(x), C, Q
Output: g_{Θ'}(x)
# Initialization Stage
Partition S into C groups randomly where S = ⋃_t^C S^t
for group index i = 1 to C do
    Initialize {e_{i,1}, B_{i,1}} randomly, A_{i,1} with zeros
    for key index j = 2 to Q do
        Initialize e_{i,j} orthogonal to e_{i,j−1}
        Initialize B_{i,j} randomly, A_{i,j} with zeros
    end for
end for
# Fine-tuning Stage
for all (x_i, y_i) ∈ S^t, t ∈ {1, ..., C} do
    Encode x_i via RoBERTa as p_i
    Retrieve {Δθ_1, ..., Δθ_k} via cosine similarity between p_i and E^t = {e_{t,1}, ..., e_{t,Q}}
    Obtain the weight increment Δθ_i = (Σ_j^k D(p_i, e_j)Δθ_j)/(Σ_j^k D(p_i, e_j))
    Calculate the loss L_i = L(g'_{Θ+Δθ_i}(x_i), y_i)
    Update the parameters {Θ, Δθ_1, ..., Δθ_k}
end for
```

dates of these $\{\Delta\theta_1, ..., \Delta\theta_k\}$, in turn, affect the parameter updates for the actual tasks. The complete process is summarized in Algorithm 1. The weight increments are additional parameters that are not integrated into the model parameters after training–unlike LoRA–and are treated as prior knowledge instead. The joint task learning mechanism mitigates knowledge sharing when learning new tasks, reducing reliance on spurious patterns and thereby improving generalization in downstream tasks.

4.3 Knowledge Reinstatement

In this subsection, we introduce target-guided task knowledge injection in which we make the model reinstate the prior knowledge while learning the current task.

Firstly, we steer the model memory to recall which forms of inputs lead to the target. Given a training example (x_i, y_i), we calculate $x_i^{'} = f_\Theta^{-1}(y_i)$ that have the highest probability generating y_i for the existing model f_Θ, which is similar to x_i. Directly calculating the $x_i^{'}$ is infeasible due to the massive token combinations in input space, we focus on the input embedding q_i for $x_i^{'}$ instead. We propose to train an intermediate model to learn the target-guided prior input. The input is q_i and the target is y_i. The model parameters are initialized from f_Θ. The intermediate training objective is shown as follows:

$$L_{Inter} = -\sum_{(x_i,y_i)\in S} CE(f_\Theta(q_i, y_i)) - W(q_i, emb(x_i)) \tag{5}$$

In the intermediate training process, all parameters except q_i are frozen, and q_i is initialized with the embedding result $emb(x_i)$ of x_i from f_Θ to accelerate convergence speed. $W(q_i, emb(x_i))$ is the wasserstein distance between the new form input p_i with its original version $emb(x_i)$. L_{Inter} helps the model recall prior knowledge (prompt representation) q_i that may lead to target y_i while the wasserstein distance W helps stabilize training dynamics.

After we obtain the prior input q_i for each example, we train the model to repetitively generate the target by adding the following reinstatement training objective.

$$L_{ReInst} = -\sum_{(x_i,y_i)\in S} \max(CE(g_{\Theta'}(q_i), y_i), CE(g_{\Theta'}(x_i), y_i)) \tag{6}$$

The final training objective is $L = L + \alpha \times L_{ReInst}$. To alleviate potential conflicts between learning the two objectives, we employ gradient projection PCGrad (Yu et al., 2020) during the gradient updates. The final objective helps the model learn varied forms of inputs leading to a similar target output. Combined with the previously mentioned latent task adaptation, this approach helps the model reduce reliance on specific input-target patterns, thereby improving its generalization in fine-tuning.

5 Experiments

5.1 Experimental Setup

Datasets and Metrics. We employ two datasets to evaluate the effectiveness of our method. (1) Super Natural Instructions (SNI for short) (Wang et al., 2022b), a broad-coverage instruction tuning dataset with 1600+ diverse NLP tasks. We adopt the default division of the training set and test set where the task types of the two sets are different. We randomly sample at most 100 instances for each task to build the training, validation, and testing sets. The numbers of tasks for training, validation, and testing are 700, 128, and 119 respectively. The

	SNI				FLAN			
	BLEU-4	ROUGE-1	ROUGE-2	ROUGE-L	BLEU-4	ROUGE-1	ROUGE-2	ROUGE-L
Llama-3.1-8B								
Vanilla	15.82	20.90	4.56	17.30	17.98	22.33	11.55	17.54
SFT	41.50	50.18	9.79	47.96	46.88	51.66	21.84	49.09
PACE	42.22	49.66	10.89	47.39	46.21	51.67	21.56	49.14
IACA	42.03	50.25	10.12	48.67	46.19	51.00	21.68	48.44
SLM	42.04	49.01	10.11	46.95	42.99	45.80	20.09	42.41
TKI (Ours)	**43.27**	**52.17**	**10.98**	**50.04**	**48.44**	**53.13**	**22.28**	**50.61**
Vicuna-13B								
Vanilla	27.49	28.86	8.73	25.57	23.19	26.41	13.54	21.99
SFT	41.51	49.37	9.27	46.80	46.70	51.80	20.52	48.75
PACE	41.71	49.07	9.60	46.70	45.75	49.75	20.61	46.65
IACA	40.50	46.26	9.50	44.65	44.69	48.11	19.91	45.52
SLM	41.21	48.77	8.77	46.29	46.88	51.76	20.89	48.63
TKI (Ours)	**42.12**	**49.81**	**10.44**	**47.96**	**47.58**	**52.20**	**22.71**	**49.76**

Table 1: The overall performance of the compared methods on Llama3.1-8B and Vicuna-13B on Super Natural Instructions (SNI) and FLAN. The best results are shown in bold.

test set is used to evaluate the in-domain generalization performance. (2) FLAN (Wei et al., 2022) is another collection of natural language processing tasks that differs in structures and formats with SNI. We use FLAN to evaluate out-of-domain generalization performance. Specifically, we adopt the default test set and randomly sample 2,048 instances as our test set for evaluation. Note that we use only the training data from SNI and report evaluation metrics on SNI and FLAN. We report BLEU-4, ROUGE-1, ROUGE-2, and ROUGE-L scores on the two test sets.

Models. We verify our method on two open-sourced large language models (LLMs): Vicuna-13B (v1.5) [1] and Llama-3.1-8B (Instruct Version) [2].

Baselines. We compare our method (TKI) with five state-of-the-art (SOTA) baselines: (1) the vanilla LLM without additional fine-tuning. (2) supervised fine-tuning (SFT) which fine-tunes the model to generate the target sequence based on the prompt (instruction + input). (3) PACE (Ni et al., 2024), a method that improves generalization through consistency regularization. (4) IACA (Zhao et al., 2024) is a data-augmented method that improves generalization through instruction augmentation and consistency alignment. (5) SLM (Peng et al., 2024), a SOTA continual learning method that employs joint parameterization with task-related knowledge retrieval to improve generalization. All baselines are trained on the same dataset (SNI) as our method.

Implementation Details. We use the SNI training dataset for our method. We implement our code based on LLama-Factory[3] and modify the original Transformer code to incorporate gradient projection with PCGrad. We release our codes in this url[4]. We use RoBERTa-base (Liu, 2019) to encode the query (prompt) and pre-compute the embeddings of each query, inserting them into the offline training file to accelerate training speed. We adopt `C=16, Q=12, k=2` as the default setting for joint task learning. In the intermediate learning process for knowledge reinstatement, we use the model to be fine-tuned as the reference model. The training results (token embeddings) are also inserted into the training file as offline data. This process is trained with SNI with `epochs=3, lr=5e-6`. We train our method with LORA and set the following parameters: `lora_target=all, lora_rank=8, bsz=1,` α`=1, gradient_accumulation_steps=4, lr=1e-5, epochs=3`. The other hyperparameters are choose from the SNI validation set and can be found in our code. We train our method and the baselines using 8 A100 40G GPU cards and our method takes 12

[1]https://huggingface.co/lmsys/vicuna-13b-v1.5

[2]https://huggingface.co/meta-llama/Llama-3.1-8B-Instruct

[3]https://github.com/hiyouga/LLaMA-Factory/

[4]https://github.com/yukunZhao/TK_injection

hours.

5.2 Main results

Firstly, we evaluate whether the proposed method improves generalization on unseen tasks. We report the overall performance on the SNI and FLAN test sets in Table 1. We observe that the proposed method, TKI, outperforms all baselines across all metrics on SNI, demonstrating its ability to enhance generalization on in-domain data. Moreover, TKI also achieves the best results on the unseen FLAN dataset, further validating its effectiveness on out-of-domain tasks.

Notably, the vanilla models, Vicuna-13B and LLaMA3.1-8B, exhibit different performances on FLAN and SNI. LLaMA3.1-8B performs better on FLAN than on SNI, whereas Vicuna-13B achieves higher scores on SNI than on FLAN. To verify the effectiveness of each method, we compare them against the Vanilla. After supervised fine-tuned (SFT), both models show significantly improved instruction-following ability, leading to enhanced performance on SNI as well as improved results on FLAN.

PACE and IACA are strong baselines on SNI, but their performance on the unseen dataset FLAN is comparable to SFT. In contrast, our method not only achieves significant improvements on unseen tasks within the same dataset but also performs well on an unseen dataset. This validates that our approach assimilates more comprehensive task knowledge, enabling better capture of in-domain patterns while also achieving strong generalization in out-of-domain scenarios.

5.3 Detailed Analysis

In this subsection, we conduct a detailed analysis of the effectiveness of input-guided latent task adaptation, target-guided knowledge reinstatement, the effects of the choices of different α, the necessity of gradient projection, and the hyperparameters in joint task learning. We use Llama3.1-8B as the backbone model as it achieves the best performance in the previous test sets.

Ablation Study of Latent Task Adaptation and Knowledge Reinstatement. To conduct an ablation study on the effectiveness of each component, we compare the performance of our method when excluding latent task adaptation (LTA) and knowledge reinstatement (KR) against the full TKI method on the SNI and FLAN test sets, as shown in Figure 2. On the SNI dataset, we observe that when LTA or KR is absent in TKI, the performance degrades compared to the full version TKI, confirming the effectiveness of these two components. Additionally, the decline is more noticeable when LTA is excluded compared to when KR is excluded, indicating that input-relevant latent tasks play a more critical role. On the out-of-domain dataset FLAN, the model's performance does not show a significant drop when LTA or KR is removed compared to the full version. The reason is that the model has been exposed to similar tasks and data distributions, making it sensitive to subtle changes on in-domain test set SNI. However, on the out-of-domain test set, these subtle differences may be ignored due to the larger distributional shift.

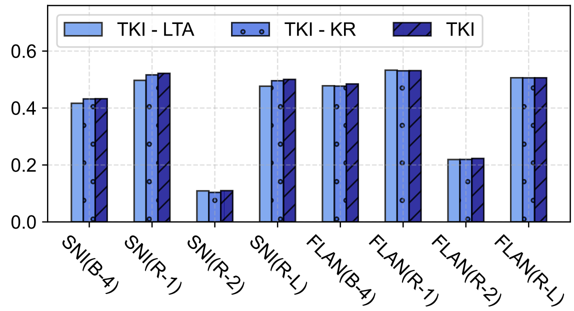

Figure 2: The comparisons of task knowledge injection (TKI) without latent task adaptation (LTA) and knowledge reinstatement (KR). B-4, R-1, R-2, and R-L are BLUE-4, ROUGE-1, ROUGE-2, and ROUGE-L metrics for short.

Furthermore, we conduct an additional study to examine the Pearson correlation between the Wasserstein distance of the input embeddings inferred from $g_{\Theta'(x)}$ and token embeddings inferred from the intermediate model and the final performance in Table 2. The p-values of all the results are less than 0.05.

BLEU-4	ROUGE-1	ROUGE-2	ROUGE-L
0.447	0.232	0.290	0.249

Table 2: The Pearson correlation values between Wasserstein distance and BLEU, ROUGE-1, ROUGE-2, and ROUGE-L on the SNI test set using Llama3.1-8B.

As shown in the table above, we observe a relatively positive correlation between the distance and the accuracy metrics. This suggests that if the samples experience less knowledge interference with prior knowledge (i.e., larger distances), their performance would be better. This underscores the necessity of incorporating knowledge reinstatement.

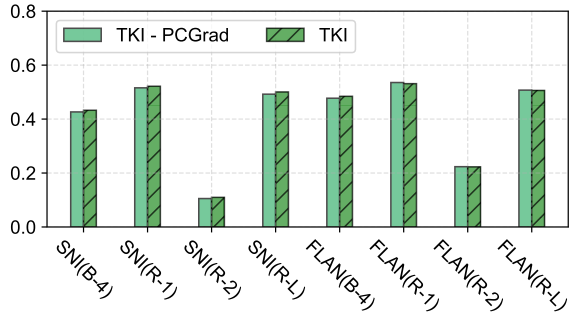

Figure 3: The comparison of task knowledge injection (TKI) with and without PCGrad. B-4, R-1, R-2, and R-L are BLUE-4, ROUGE-1, ROUGE-2, and ROUGE-L metrics for short.

The Effect of Gradient Projection. We compare the performance of using gradient projection (specifically, PCGrad) versus not using it when combining the two learning objectives in our method. As shown in Figure 3, on the SNI dataset, omitting PCGrad leads to a slight performance decline. This suggests a conflict between the naive supervised fine-tuning and knowledge reinstatement training objectives, thereby confirming the necessity of the projection gradient. On the out-of-domain FLAN dataset, however, the difference between the two approaches is negligible. The reason is that differences in gradient projection are more likely to be overlooked due to the larger distribution shift, as previously analyzed.

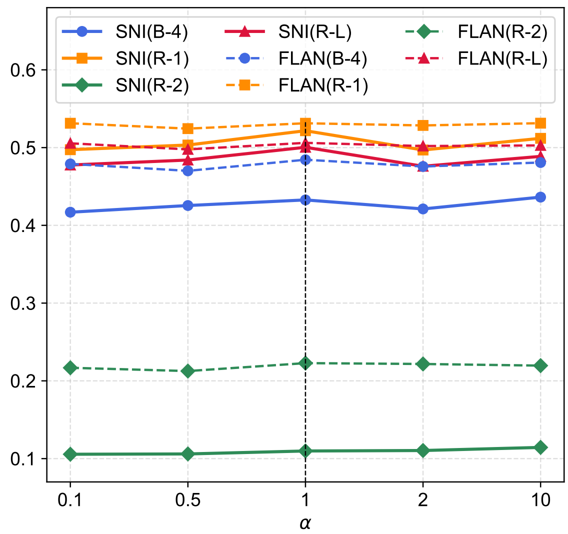

Figure 4: The comparison of different choice of α. The solid and dashed lines represent the metrics on SNI and FLAN, respectively. B-4, R-1, R-2, and R-L are BLUE-4, ROUGE-1, ROUGE-2, and ROUGE-L metrics for short. The optimal choice is marked with the vertical dashed line.

The Choice of different α. We then investigate the effect of choosing different α on task knowledge injection. We report the compared performance on SNI and FLAN, as shown in Figure 4. We observe that the model achieves its best performance when α is set to 1. As α increases from 0.1 to 1, the metrics on SNI show an upward trend. This demonstrates that a greater emphasis on knowledge reinstatement can help the model develop a better understanding of the task, thereby improving performance. While the trend on FLAN is less noticeable, the reason is that the vanilla model performs better on FLAN compared to SNI. As a result, the performance remains similar even without the reinstatement of prior knowledge on the unseen FLAN dataset. Additionally, when α continues to increase exceeding 1, the overall performance on both SNI and FLAN shows a slight decline. This indicates that the reinstatement of prior knowledge should not be overly emphasized when learning new tasks.

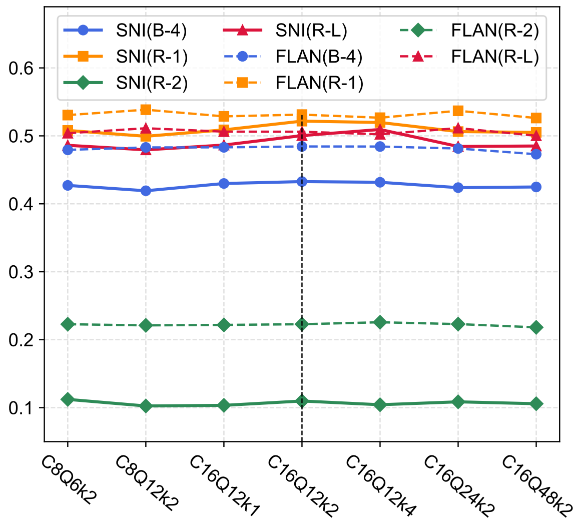

Figure 5: The compared performance of our method on SNI and FLAN when using different numbers of groups (C), keys (Q), and nearest neighbors (k) in joint task learning. The solid and dashed lines represent the metrics on SNI and FLAN, respectively. B-4, R-1, R-2, and R-L are BLUE-4, ROUGE-1, ROUGE-2, and ROUGE-L metrics for short. The optimal choice is marked with the vertical dashed line.

The hyperparameters in Joint Task Learning. Finally, we examine the effects of different cluster numbers (C), varying numbers of keys (Q) within each cluster, and different k-nearest latent tasks in joint task learning on the model performance. We report compared metrics on SNI and FLAN, where $C8Q6k2$ represents $C = 8$, $Q = 6$, and $k = 2$, with others following the same pattern.

As shown in Figure 5, the model achieves its best performance when $C = 16, Q = 12, k = 2$. The model performance across different values of C, Q, and k varies slightly. From the ROUGE-L metric perspective, when $C = 16$ and $Q = 12$ are fixed, increasing k improves performance. This suggests that incorporating more similar latent tasks helps the model acquire additional relevant knowledge, thereby enhancing its effectiveness. Conversely, when $C = 16$ and $k = 2$ are fixed, increasing Q leads to a decline in performance, likely due to weakened knowledge sharing caused by finer-grained key partitioning. Notably, the optimal hyperparameter settings depend on the scale of the training samples and should be determined heuristically.

6 Conclusion

In this paper, we propose a novel instruction-tuning method called Task Knowledge Injection. It is a data-efficient knowledge injection approach that enhances task generalization without relying on additional models or data. We conduct extensive experiments on recent large language models, including Llama3.1-8B and Vicuna-13B, across 1000+ tasks from Super Natural Instructions and FLAN datasets. The experimental results demonstrate the superior performance of our method and each component on both in-domain and out-of-domain datasets. Our method enables the model to establish correlations between new tasks and between new tasks and prior knowledge, facilitating building task knowledge more effectively, and thereby improving generalization performance. We conduct detailed analyses revealing that joint task adaptation plays a more critical role in the knowledge injection process. We also show that incorporating more relevant latent tasks leads to slightly improved performance while weakening knowledge sharing results in degraded performance.

Limitations

Although our task knowledge injection effectively improves generalization in instruction tuning, it still has several limitations. First, in latent task adaptation, we partition the latent tasks into several clusters which may limit knowledge sharing across inter-clusters. We plan to adopt a more efficient retrieval method eliminating the need for partitioning in the future, which also helps reduce the number of hyperparameters. Additionally, we employ an intermediate learning process to derive input forms for the target, which may introduce additional errors across the modules. We plan to develop an end-to-end training approach for knowledge reinstatement and explore more effective methods to acquire diverse prior knowledge.

References

Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, and 1 others. 2023. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*.

Rahaf Aljundi, Francesca Babiloni, Mohamed Elhoseiny, Marcus Rohrbach, and Tinne Tuytelaars. 2018. Memory aware synapses: Learning what (not) to forget. In *ECCV*, pages 139–154.

Enes Altinisik, Hassan Sajjad, Husrev Sencar, Safa Messaoud, and Sanjay Chawla. 2023. Impact of adversarial training on robustness and generalizability of language models. In *Findings of ACL 2023*, pages 7828–7840.

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, and 1 others. 2020. Language models are few-shot learners. In *NeurIPS*, pages 1877–1901.

Collin Burns, Pavel Izmailov, Jan Hendrik Kirchner, Bowen Baker, Leo Gao, Leopold Aschenbrenner, Yining Chen, Adrien Ecoffet, Manas Joglekar, Jan Leike, and 1 others. 2023. Weak-to-strong generalization: Eliciting strong capabilities with weak supervision. *arXiv preprint arXiv:2312.09390*.

Hyung Won Chung, Le Hou, Shayne Longpre, Barret Zoph, Yi Tay, William Fedus, Yunxuan Li, Xuezhi Wang, Mostafa Dehghani, Siddhartha Brahma, and 1 others. 2024. Scaling instruction-finetuned language models. *JMLR*, 25(70):1–53.

Pierre Foret, Ariel Kleiner, Hossein Mobahi, and Behnam Neyshabur. 2021. Sharpness-aware minimization for efficiently improving generalization. In *ICLR*.

Zhe Gan, Yen-Chun Chen, Linjie Li, Chen Zhu, Yu Cheng, and Jingjing Liu. 2020. Large-scale adversarial training for vision-and-language representation learning. In *NeurIPS*, pages 6616–6628.

Shuhao Gu, Bojie Hu, and Yang Feng. 2022. Continual learning of neural machine translation within low forgetting risk regions. In *EMNLP*, pages 1707–1718.

Md Yousuf Harun, Kyungbok Lee, Jhair Gallardo, Giri Krishnan, and Christopher Kanan. 2024. What variables affect out-of-distribution generalization in pretrained models? *arXiv preprint arXiv:2405.15018*.

Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. 2021. Lora: Low-rank adaptation of large language models. *arXiv preprint arXiv:2106.09685*.

Haoming Jiang, Pengcheng He, Weizhu Chen, Xiaodong Liu, Jianfeng Gao, and Tuo Zhao. 2019. Smart: Robust and efficient fine-tuning for pretrained natural language models through principled regularized optimization. *arXiv preprint arXiv:1911.03437*.

Jiarui Jiang, Wei Huang, Miao Zhang, Taiji Suzuki, and Liqiang Nie. 2024. Unveil benign overfitting for transformer in vision: Training dynamics, convergence, and generalization. In *NeurIPS*.

James Kirkpatrick, Razvan Pascanu, Neil Rabinowitz, Joel Veness, Guillaume Desjardins, Andrei A Rusu, Kieran Milan, John Quan, Tiago Ramalho, Agnieszka Grabska-Barwinska, and 1 others. 2017. Overcoming catastrophic forgetting in neural networks. *PNAS*, 114(13):3521–3526.

Ananya Kumar, Aditi Raghunathan, Robbie Matthew Jones, Tengyu Ma, and Percy Liang. 2022. Fine-tuning can distort pretrained features and underperform out-of-distribution. In *ICLR*.

Brian Lester, Rami Al-Rfou, and Noah Constant. 2021. The power of scale for parameter-efficient prompt tuning. *arXiv preprint arXiv:2104.08691*.

Dongyue Li and Hongyang Zhang. 2021. Improved regularization and robustness for fine-tuning in neural networks. In *NeurIPS*, pages 27249–27262.

Xiang Lisa Li and Percy Liang. 2021. Prefix-tuning: Optimizing continuous prompts for generation. In *ACL*, pages 4582–4597.

Yunlong Liang, Fandong Meng, Jiaan Wang, Jinan Xu, Yufeng Chen, and Jie Zhou. 2024. Continual learning with semi-supervised contrastive distillation for incremental neural machine translation. In *ACL*, pages 10914–10928.

Yinhan Liu. 2019. Roberta: A robustly optimized bert pretraining approach. *arXiv preprint arXiv:1907.11692*.

David Lopez-Paz and Marc'Aurelio Ranzato. 2017. Gradient episodic memory for continual learning. In *NeurIPS*.

Sanae Lotfi, Marc Anton Finzi, Yilun Kuang, Tim GJ Rudner, Micah Goldblum, and Andrew Gordon Wilson. 2024. Non-vacuous generalization bounds for large language models. In *ICML*.

Emily McMilin. 2022. Selection bias induced spurious correlations in large language models. In *ICML 2022: Workshop on Spurious Correlations, Invariance and Stability*.

Sewon Min, Patrick Lewis, Hannaneh Hajishirzi, and Luke Zettlemoyer. 2022. Rethinking the role of demonstrations: What makes in-context learning work? *arXiv preprint arXiv:2202.12837*.

Shervin Minaee, Tomas Mikolov, Narjes Nikzad, Meysam Chenaghlu, Richard Socher, Xavier Amatriain, and Jianfeng Gao. 2024. Large language models: A survey. *arXiv preprint arXiv:2402.06196*.

Takeru Miyato, Shin-ichi Maeda, Masanori Koyama, and Shin Ishii. 2018. Virtual adversarial training: a regularization method for supervised and semi-supervised learning. *TPAMI*, 41(8):1979–1993.

Yao Ni, Shan Zhang, and Piotr Koniusz. 2024. Pace: marrying generalization in parameter-efficient fine-tuning with consistency regularization. *arXiv preprint arXiv:2409.17137*.

OpenAI. 2024. Gpt-4v: Multimodal language model. *OpenAI Technical Report*.

Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, and 1 others. 2022. Training language models to follow instructions with human feedback. In *NeurIPS*, pages 27730–27744.

Lin Pan, Chung-Wei Hang, Avirup Sil, and Saloni Potdar. 2022. Improved text classification via contrastive adversarial training. In *AAAI*, volume 36, pages 11130–11138.

Bohao Peng, Zhuotao Tian, Shu Liu, Mingchang Yang, and Jiaya Jia. 2024. Scalable language model with generalized continual learning. *arXiv preprint arXiv:2404.07470*.

Rafael Rafailov, Archit Sharma, Eric Mitchell, Christopher D Manning, Stefano Ermon, and Chelsea Finn. 2024. Direct preference optimization: Your language model is secretly a reward model. In *NeurIPS*.

Google Research. 2024. Gemini: Multimodal large language model. *Google AI Blog*.

Hippolyt Ritter, Aleksandar Botev, and David Barber. 2018. Online structured laplace approximations for overcoming catastrophic forgetting. In *NeurIPS*.

Adam Santoro, Sergey Bartunov, Matthew Botvinick, Daan Wierstra, and Timothy Lillicrap. 2016. Meta-learning with memory-augmented neural networks. In *ICML*, pages 1842–1850.

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. 2017. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*.

Jonathan Schwarz, Wojciech Czarnecki, Jelena Luketina, Agnieszka Grabska-Barwinska, Yee Whye Teh, Razvan Pascanu, and Raia Hadsell. 2018. Progress & compress: A scalable framework for continual learning. In *ICML*, pages 4528–4537.

Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, Aurelien Rodriguez, Armand Joulin, Edouard Grave, and Guillaume Lample. 2023. Llama: Open and efficient foundation language models. *arXiv preprint arXiv:2302.13971*.

Marlieke TR van Kesteren, Paul Rignanese, Pierre G Gianferrara, Lydia Krabbendam, and Martijn Meeter. 2020. Congruency and reactivation aid memory integration through reinstatement of prior knowledge. *Scientific Reports*, 10(1):4776.

Johannes Von Oswald, Eyvind Niklasson, Ettore Randazzo, João Sacramento, Alexander Mordvintsev, Andrey Zhmoginov, and Max Vladymyrov. 2023. Transformers learn in-context by gradient descent. In *ICML*, pages 35151–35174.

Mingze Wang, Jinbo Wang, Haotian He, Zilin Wang, Guanhua Huang, Feiyu Xiong, Zhiyu Li, Lei Wu, and 1 others. 2024. Improving generalization and convergence by enhancing implicit regularization. *arXiv preprint arXiv:2405.20763*.

Yihan Wang, Si Si, Daliang Li, Michal Lukasik, Felix Yu, Cho-Jui Hsieh, Inderjit S Dhillon, and Sanjiv Kumar. 2022a. Two-stage llm fine-tuning with less specialization and more generalization. In *ICLR*.

Yizhong Wang, Swaroop Mishra, Pegah Alipoormolabashi, Yeganeh Kordi, Amirreza Mirzaei, Atharva Naik, Arjun Ashok, Arut Selvan Dhanasekaran, Anjana Arunkumar, David Stap, and 1 others. 2022b. Super-naturalinstructions: Generalization via declarative instructions on 1600+ nlp tasks. In *EMNLP*, pages 5085–5109.

Jason Wei, Maarten Bosma, Vincent Zhao, Kelvin Guu, Adams Wei Yu, Brian Lester, Nan Du, Andrew M Dai, and Quoc V Le. 2022. Finetuned language models are zero-shot learners. In *ICLR*.

Jason Wei, Maarten Bosma, Vincent Y Zhao, Kelvin Guu, Adams Wei Yu, Brian Lester, Nan Du, Andrew M Dai, and Quoc V Le. 2021. Finetuned language models are zero-shot learners. *arXiv preprint arXiv:2109.01652*.

Sophie Xhonneux, Alessandro Sordoni, Stephan Günnemann, Gauthier Gidel, and Leo Schwinn. 2024. Efficient adversarial training in llms with continuous attacks. *arXiv preprint arXiv:2405.15589*.

Can Xu, Qingfeng Sun, Kai Zheng, Xiubo Geng, Pu Zhao, Jiazhan Feng, Chongyang Tao, and Daxin Jiang. 2023. Wizardlm: Empowering large language models to follow complex instructions. *arXiv preprint arXiv:2304.12244*.

Haoran Yang, Hongyuan Lu, Wai Lam, and Deng Cai. 2024a. Exploring compositional generalization of large language models. In *NAACL*, pages 16–24.

Haoran Yang, Yumeng Zhang, Jiaqi Xu, Hongyuan Lu, Pheng Ann Heng, and Wai Lam. 2024b. Unveiling the generalization power of fine-tuned large language models. *arXiv preprint arXiv:2403.09162*.

Tong Yang, Yu Huang, Yingbin Liang, and Yuejie Chi. 2024c. In-context learning with representations: Contextual generalization of trained transformers. In *NeurIPS*.

Tianhe Yu, Saurabh Kumar, Abhishek Gupta, Sergey Levine, Karol Hausman, and Chelsea Finn. 2020. Gradient surgery for multi-task learning. In *NeurIPS*.

Yukun Zhao, Lingyong Yan, Weiwei Sun, Guoliang Xing, Shuaiqiang Wang, Chong Meng, Zhicong Cheng, Zhaochun Ren, and Dawei Yin. 2024. Improving the robustness of large language models via consistency alignment. In *LREC-COLING*, pages 8931–8941.

Hongling Zheng, Li Shen, Yong Luo, Tongliang Liu, Jialie Shen, and Dacheng Tao. 2024. Decomposed prompt decision transformer for efficient unseen task generalization. In *NeurIPS*.

Chunting Zhou, Pengfei Liu, Puxin Xu, Srinivasan Iyer, Jiao Sun, Yuning Mao, Xuezhe Ma, Avia Efrat, Ping Yu, Lili Yu, and 1 others. 2024. Lima: Less is more for alignment. In *NeurIPS*.

Yuhang Zhou, Paiheng Xu, Xiaoyu Liu, Bang An, Wei Ai, and Furong Huang. 2023. Explore spurious correlations at the concept level in language models for text classification. *arXiv preprint arXiv:2311.08648*.

Chen Zhu, Yu Cheng, Zhe Gan, Siqi Sun, Tom Goldstein, and Jingjing Liu. 2019. Freelb: Enhanced adversarial training for natural language understanding. *arXiv preprint arXiv:1909.11764*.