VaiBot: Shuttle between the Instructions and Parameters of Large Language Models

Anonymous ACL submission

Abstract

The interaction with LLMs through instructions has been extensively investigated in the research community. However, previous studies have treated the emergence of instructions and the training of LLMs on task data as separate processes, overlooking the inherent unity between the two. This paper proposes a novel neural network framework, VaiBot, that integrates VAE and VIB, designed to uniformly model, learn, and infer both instruction *deduction* and instruction *induction* tasks of LLMs. Through experiments, we demonstrate that VaiBot performs on par with existing baseline methods in terms of deductive capabilities while significantly surpassing them in inductive capabilities. We also find that VaiBot can scale up using general instruction-following data and exhibits excellent one-shot induction abilities. We finally synergistically integrate the deduction and induction processes of VaiBot for the task of *inductive reasoning*. Through t-SNE dimensionality reduction, we observe that its inductive-deductive process significantly improves the distribution of training parameters, enabling it to outperform baseline methods in inductive reasoning tasks. The code and data for this paper can be found at https://anonymous.4open.science/r/VaiBot-021F.

1 Introduction

With the rise of Large Language Models (LLMs), an increasing number of researchers and application scenarios are beginning to explore interacting with LLMs through *instructions*. Instructions are a type of natural language that delineates task objectives, characterized by a high level of abstraction and refined task knowledge.

Existing research has approached the interaction between instructions and LLMs from two objectives: *deduction* and *induction*. Specifically, given an instruction k, task inputs x_i, and targets y_i, deduction requires the model to generate the target y_i based on the instruction k and input x_i; whereas induction demands the model to predict the task instruction k based on a large number of x_i and y_i as observations.

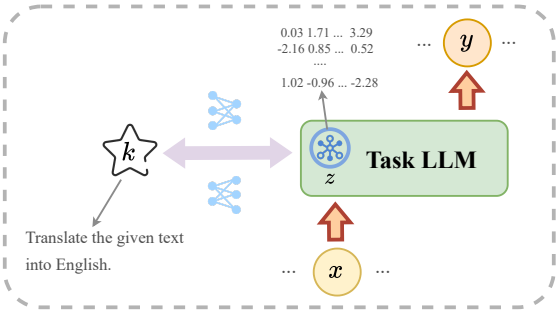

Figure 1: The basic concept of unifying the modeling of instruction deduction and induction of LLMs.

However, the previous studies have treated the instruction *deduction* and *induction* of LLMs as separate processes (§2), overlooking the inherent unity between the two. In fact, instructions are a compression of task data by humans through natural language, while the parameterized gradient descent training of the LLMs, also constitutes a compression of task data. Therefore, the parameters of an LLM after training on specific tasks should exhibit a high degree of correlation with the task instructions. Consequently, we propose to learn the mutual mapping between the instructions and the parameters, aiming to unify the modeling of instruction deduction and induction of LLMs (Figure 1).

Building upon this concept, we introduce VaiBot (**V**ariational **a**utoencoder and **i**nformation **Bot**tleneck) , a novel framework that integrates the Variational Autoencoder (VAE) (Kingma, 2013) and Variational Information Bottleneck (VIB) (Alemi et al., 2016) to uniformly model, learn, and infer both instruction deduction and induction tasks

of LLMs. VaiBot operates by first encoding the instruction k into a latent representation z via an encoder. This latent z then serves a dual purpose: it is utilized by a decoder to reconstruct the original instruction k, thereby facilitating the induction process (VAE). Concurrently, z is employed as supplementary parameters for a Task LLM, aiming to predict the target y_i from the given input x_i, which is a deduction process (VIB). The two objectives together with the regularization term are end-to-end jointly optimized to learn the functions of the encoder, decoder, and Task LLM.

We initially conducted a series of experiments to validate the effectiveness of VaiBot in both deduction and induction tasks. The experimental results revealed that VaiBot's deductive capabilities are on par with those of supervised fine-tuning (SFT) and meta-learning. In terms of inductive capabilities, VaiBot achieved more than a 40% improvement in in-distribution performance and over a 20% improvement in out-of-distribution performance compared to traditional induction methods (§2.2). Additionally, by conducting induction tasks with varying numbers of observed samples, we found that VaiBot exhibits superior 1-shot induction capabilities compared to data-based induction methods.

We further trained VaiBot using general instruction-following data and observed that, as the volume of training increased, it developed the ability to generalize across tasks for both deduction and induction, demonstrating that the architecture can effectively scale up.

Finally, we synergistically combine the deductive and inductive processes of VaiBot to perform *inductive reasoning*. Compared to baseline methods such as ICL and Instruction Induction (Honovich et al., 2023), VaiBot is much more effective in conducting inductive reasoning. Through t-SNE dimensionality reduction, we observed that the induction-deduction process of VaiBot significantly improved the distribution of the latent, thereby enabling the Task LLM to achieve superior reasoning performance.

In summary, this paper makes the following contributions:

- We propose a novel neural network framework, VaiBot, that integrates VAE and VIB, designed to uniformly model, learn, and infer both instruction deduction and instruction induction tasks of LLMs.
- We demonstrate that VaiBot performs on par with existing baseline methods in terms of deductive capabilities while significantly surpassing them in inductive capabilities. We also find that VaiBot can scale up using general instruction-following data and exhibits excellent one-shot induction abilities.
- We synergistically integrate the deduction and induction processes of VaiBot. Through t-SNE dimensionality reduction, we observe that its induction-deduction process significantly improves the distribution of training parameters, enabling it to outperform baseline methods in inductive reasoning tasks.

2 Related Work

2.1 Instruction-based LLM Deduction

Given an instruction, how to ask LLM to perform deduction based on it, i.e. instruction following, has been widely considered by researchers. Previous studies such as IFEval (Zhou et al., 2023), InfoBench (Qin et al., 2024), and RuleBench (Sun et al., 2024b) have been instrumental in evaluating the capacity of large models to follow the instructions, also demonstrating that instruction fine-tuning (IFT) can significantly bolster this capability.

Different from the prompt-level instruction-following paradigm, Meta-Learning methods like Hint (Ivison et al., 2023) and TAGI (Liao et al., 2024) have tried training a hyper-network to encode the instruction into some extra parameters of LLMs to execute the instruction. However, these Meta-Learning methods rely heavily on supervised training conducted in advance on each subtask to obtain (instruction, parameter) pairs as training data for the hyper-network.

VaiBot employs a similar hyper-network architecture that maps instructions to LLMs' parameters, but it further integrates a reconstruction process, enabling the training of this hyper-network to no longer depend on pre-prepared (parameter, instruction) pairs. Instead, it can be trained on general instruction-following datasets.

2.2 Instruction-oriented LLM Induction

For the sake of interpretability and generalization, some previous works also try to induce instruction from task observations through LLMs. Some evaluation studies (Mirchandani et al., 2023; Gendron et al., 2023; Mitchell et al., 2023) have consistently demonstrated that current LLMs are poor at the

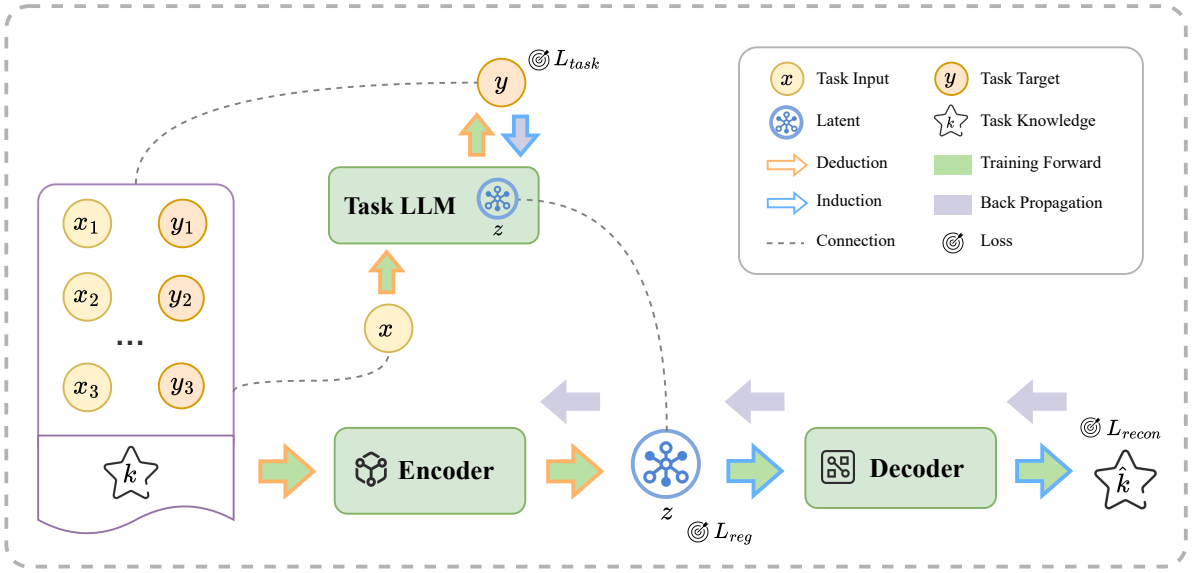

Figure 2: The framework of VaiBot. The Training process is represented with filled colors and the inference process is represented with border colors.

task of induction. To improve LLMs' capability of induction, methods such as Hypothesis Search (Wang et al., 2023) and ItD (Sun et al., 2024a) have modeled induction as a sequence generation task, attempting to enhance the inductive abilities of large models through approaches like sampling-selecting and augmenting-finetuning.

However, these methods are confined to *data-based induction* and overlook the fact that the parameters of neural networks, once trained to converge on task data, provide highly indicative cues for the objectives of induction. VaiBot introduces *parameter-based induction*, and our experiments have demonstrated that this approach significantly outperforms the previous series of data-based induction methods.

3 VaiBot

VaiBot is trained to map a given textual knowledge k to a latent z, which not only can serve as the extra parameters of an LLM, to solve the downstream task (VIB); but can also used for reconstruct the textual knowledge k (VAE). It is mainly composed of three models:

- **Encoder**. A textual encoder that encode the knowledge k to the latent z. This mapping is denoted as $Enc(\cdot)$.
- **Decoder**. An auto-regressive decoder that decode the latent z to the textual knowledge k. The distribution of decoder is denoted as $p_{dec}(\cdot)$.
- **Task LLM**. An LLM that solve the downstream task. The distribution of Task LLM is denoted as $p_{task}(\cdot)$.

In the following part of this section, we will introduce how these three models are jointly trained and how they are used for neural-symbolic bidirectional inference.

3.1 Training

As shown in Figure 2, our training data consists of triples (k, x, y), where k is the textual knowledge, x, y are the input-target pairs that y can be inferred from x using the textual knowledge k.

First, VaiBot uses the Encoder to encode the knowledge k into a high-dimension diagonal normal distribution, and calculate the regularization loss J_{reg} using Kullback–Leibler (KL) divergence.

$$\mu, \Sigma = Enc(k) \quad (1)$$

$$L_{reg} = D_{KL}(\mathcal{N}(\cdot|\mu, \Sigma)||\mathcal{N}(\cdot|0, I)) \quad (2)$$

Then, the latent z is sampled from the encoded normal distribution, here, the reparametrization trick (Kingma et al., 2015) is adopted to maintain the gradient flow. VaiBot then uses the Decoder to attempt to reconstruct the knowledge k from the latent representation z, and calculate the reconstruction loss L_{recon}. This corresponds to the objective of VAE.

$$z \sim \mathcal{N}(\cdot|\mu, \Sigma) \quad (3)$$

$$L_{recon} = -\log p_{dec}(k|z) \quad (4)$$

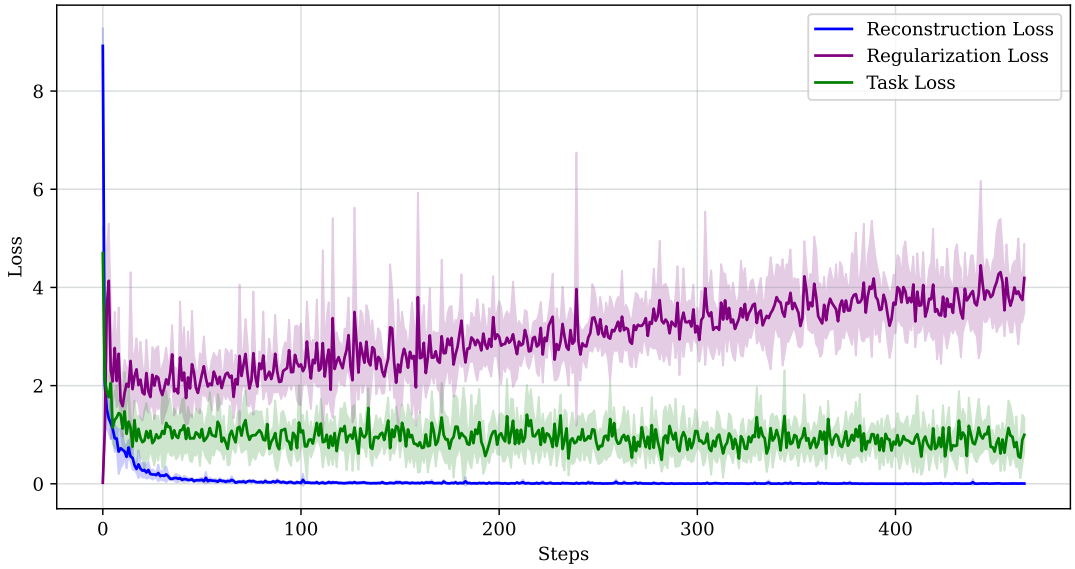

Figure 3: The loss curve of VaiBot trained on SNI with respect to the training steps.

Meanwhile, the latent representation z is taken as the extra parameters of the Task LLM. The Task LLM is asked to infer on the given task instance x, y, and calculate the task loss L_{task}. This corresponds to the objective of VIB.

$$L_{task} = -\log p_{task}(y|z;x) \tag{5}$$

To maintain and leverage the existing well-trained natural language distribution of the Task LLM and auto-regressive Decoder, we add textual condition: instruction k for the Task LLM, and one pair of instance x, y for the Decoder. So the Eq 4,5 become into:

$$L_{recon} = -\log p_{dec}(k|z;\underline{x,y}) \tag{6}$$
$$L_{task} = -\log p_{task}(y|z;x,\underline{k}) \tag{7}$$

The final objective function is the weighted sum of the three loss terms, and we minimize it under the distribution of training data.

$$L = \mathbb{E}_{(k,x,y)\sim p_{data}} w_0 L_{reg} + w_1 L_{task} + w_2 L_{recon} \tag{8}$$

3.2 Symbolic to Neural Inference

As indicated by the orange border arrows in the Figure 2, given a textual knowledge k, to perform the inference on the task input x, VaiBot encodes the knowledge k into the latent representation z, and uses the Task LLM to generate an output via auto-regressive generation.

$$\mu, \Sigma = Enc(k) \tag{9}$$
$$z \sim \mathcal{N}(\cdot|\mu, \Sigma) \tag{10}$$
$$\hat{y} \sim p_{task}(\cdot|z;x,k) \tag{11}$$

3.3 Neural to Symbolic Inference

As indicated by the blue border arrows in the Figure 2, given the multiple instances $T = (x_i, y_i)_{i=1}^n$, to infer their shared knowledge k, VaiBot first fine-tune the Task LLM on the instances T to obtain the converged latent representation z^*. Here, we adopt an *indirect training* trick to fine-tune the Task LLM, which is to create a trainable tensor $\tilde{k}$, and then encode $\tilde{k}$ into the normal distribution to get the trainable latent representation $\tilde{z}$, instead of directly initializing the $\tilde{z}$, taking it as the leaf parameters of the computation graph.

$$\tilde{\mu}, \tilde{\Sigma} = Enc(\tilde{k}) \tag{12}$$
$$\tilde{z} \sim \mathcal{N}(\cdot|\tilde{\mu}, \tilde{\Sigma}) \tag{13}$$
$$J_{task}^{\tilde{k}}(x,y) = -\log p_{task}(y|x;\tilde{z}) \tag{14}$$

Through minimizing $J_{task}^{\tilde{k}}$ on training task samples x, y, we obtain the converged $\tilde{k}$. However, what we want is the converged latent representation z^*:

$$k^* = \arg\min_{\tilde{k}} \frac{1}{n} \sum_{i=1}^{n} J_{task}(x_i, y_i) \tag{15}$$
$$\mu^*, \Sigma^* = Enc(k^*) \tag{16}$$
$$z^* \sim \mathcal{N}(\cdot|\mu^*, \Sigma^*) \tag{17}$$

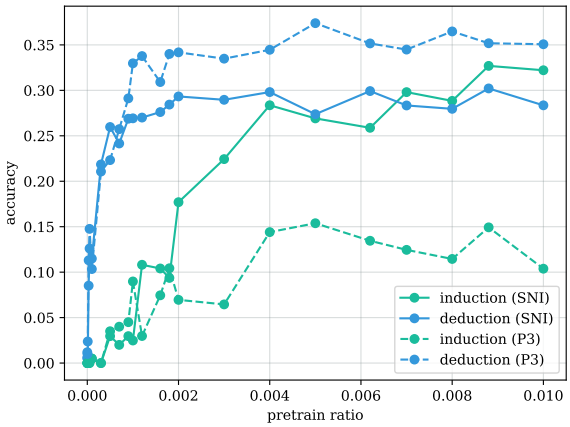

Figure 4: The OOD induction & deduction performance of VaiBot-pretrain with respect to the ratio of used pretrained data.

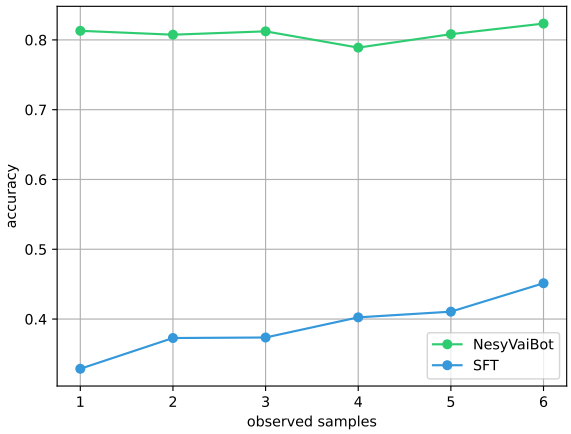

Figure 5: The induction performance of VaiBot and SFT on SNI with respect to the number of observed samples. The accuracy is the average accuracy over all seen and unseen tasks.

Finally, we randomly sample a pair of (x^*, y^*) from T to leverage the well-trained natural language distribution of the Decoder. Under this condition, we can decode the trained parameters z^* into explainable textual knowledge k:

$$\hat{k} \sim p_{dec}(\cdot|z^*; x^*, y^*) \tag{18}$$

4 Experiment Settings & Training

In this section, we introduce the experiment settings and the training of the VaiBot. We employ Llama-2-7b-chat (Touvron et al., 2023) as the base language model M. Task LLM is M itself while Encoder, Decoder is M with two LoRA (Hu et al., 2021) of rank 16 and 1, respectively. To facilitate efficient batch training & inference, we adopt prompt tuning (Lester et al., 2021) as the additional parameters of the Task LLM. The number of soft tokens is set to 10, and thus the dimension of z is $10 \times 4096 = 40960$. All other baselines that need training (later introduced in §5,6) will take the z of the same size as the training parameters for fair comparisons. The weights of the loss terms w_0, w_1, w_2 are set to 1e-3, 1.0, 1.0, respectively.

We adopt two popular multi-task instruction datasets: Super-Natural Instructions (SNI, Wang et al. 2022) and T0 split of P3 (P3, Sanh et al. 2021) for evaluation. We first split each dataset into seen tasks (90%) and unseen tasks (10%). For each subtask, we only leave 5 instances as test samples, and use the rest as training samples. Therefore, for methods that are trained on seen tasks, the test results on seen tasks reflect their *sample-level generalization* ability, while the test results on unseen tasks reflect their *task-level generalization* ability.

Besides training VaiBot with data from seen tasks for 10 epochs (VaiBot-in-domain), we additionally adopt around 437k instruction following data (Appendix A) to pretrain VaiBot for 1 epoch (VaiBot-pretrain). Note that, here we mean "pretrain" by training VaiBot to newly learn to generate and leverage the latent z from general textual data, instead of randomly initializing the base model M.

In Figure 3, we present the training loss curves for the three components during the training of VaiBot-in-domain on the SNI dataset, plotted against the training steps. The concurrent decrease in both reconstruction loss and task loss demonstrates that VaiBot effectively integrates the training processes of VAE and VIB. Regarding the regularization loss, the weight term w_0 controls the trade-off between reconstruction fidelity and the quality of disentanglement within the learned latent z (Higgins et al., 2017). Our experimental exploration of various w_0 values for the regularization term revealed that the performance of VaiBot remains robust across different settings, indicating a relative insensitivity to this parameter.

5 Induction & Deduction

In this section, we verify the effectiveness of VaiBot on separate *deduction* and *induction* tasks. In the deduction task, the model is provided with task knowledge k and input x, and asked to generate the target y; In the induction task, the model is pro-

Dataset	SNI				P3			
method	seen tasks (90%)		unseen tasks (10%)		seen tasks (90%)		unseen tasks (10%)	
	deduction	induction	deduction	induction	deduction	induction	deduction	induction
prompting *	12.70	20.63	12.21	26.32	21.78	2.78	23.28	4.76
vanilla SFT	29.42	49.20	28.56	27.78	35.89	45.00	37.31	19.05
TAGI	32.02	-	23.33	-	36.33	-	47.62	-
ItD	-	43.85	-	33.33	-	33.33	-	28.57
VaiBot-in-domain	33.26	85.56	21.11	44.44	48.67	78.33	58.10	28.57
VaiBot-pretrain *	30.37	36.36	32.22	50.00	38.22	20.00	49.52	19.05

Table 1: The induction & deduction performance of VaiBot and baselines on SNI and P3. Methods marked with * are not trained on seen tasks. - indicates that the method is not applicable to that task.

vided with 5 test samples $\{x, y\}$ as the observation, and asked to generate the task knowledge k. For the evaluation of both tasks, this paper adopts an external LLM (gpt-4o-mini[1]) as a judge to determine whether the prediction is correct, the prompts for the judge are shown in the Appendix B and the examples of deduction and induction are shown in Appendix C.

We adopt the following methods as the baselines:

- **prompting**. This method simply prompts the LLM M with the task knowledge k and input x to infer the target y (deduction) and prompts the LLM M with multiple instances (x, y) to infer the task knowledge k (induction).
- **vanilla SFT**. Based on the prompting method, we fine-tune the LLM M based on the training data of seen tasks to learn the task of deduction and induction.
- **TAGI**. TAGI (Liao et al., 2024) is a typical meta-learning-based method that fuses the knowledge into the Task LLM through hyper-network. It first trains the "reference" parameters of the Task LLM on the training data, and then leverages the (knowledge, parameters) pairs to train the hyper-network. TAGI can only be used in the *deduction* task.
- **ItD**. ItD (Sun et al., 2024a) is a recently proposed method that can empower the induction ability of the language model. It first decomposes the joint distribution of $p(x, y, k)$ with a deduction perspective into the knowledge prior $p(k)$ and deduction likelihood $p(y|x, k)p(x|k)$, and sample from them. Then, it fine-tunes the language model with the sampled data in the form of induction: $p(k|x, y)$. ItD can only be used in the *induction* task.

[1]https://openai.com/index/gpt-4o-mini-advancing-cost-efficient-intelligence/

5.1 Comparison with Baselines

We first compare the accuracy of VaiBot on deduction and induction with baselines. As shown in Table 1, while VaiBot-in-domain demonstrates competitive deduction ability compared to SFT and TAGI, it shows impressive induction ability compared to other data-based induction methods, not only outperforming ItD and vanilla SFT on the seen tasks, but also on the unseen tasks by a large margin. Moreover, the VaiBot-pretrain also demonstrates competitive performance on two datasets although it is not trained on the in-domain data. These results indicate that VaiBot demonstrates excellent *sample-level generalization* and *task-level generalization* abilities on both tasks of deduction and induction.

5.2 Ablations

To verify the effectiveness of textual condition and indirect training trick proposed in §3.1, we conduct an ablation study of VaiBot by dropping these parts. As shown in Table 2, if dropping the textual condition x, y for the Decoder, or tuning z without the indirect training trick, the induction performance will greatly decrease; if dropping the textual condition k for the Task LLM, the deduction performance will be harmed to some extent. These findings verify that the textual conditions and indirect training tricks we adopt are beneficial for NestVaiBot.

5.3 Generalization with Scaling Up

To visualize the generalization process of VaiBot, we pretrain it using varying proportions of the entire pretraining dataset and evaluate its performance on the induction and deduction tasks for SNI and P3. The resulting performance curve is depicted in Figure 4. From the curve, it is evident that Vai-

Dataset	SNI				P3			
method	seen task (90%)		unseen task (10%)		seen task (90%)		unseen task (10%)	
	deduction	induction	deduction	induction	deduction	induction	deduction	induction
VaiBot-in-domain	33.26	85.56	21.11	44.44	48.67	78.33	58.10	28.57
w/o textual condition x, y	31.65	0.53	16.67	0.00	45.67	11.67	54.29	4.76
w/o textual condition k	32.94	84.49	4.44	22.22	48.22	80.00	56.19	33.33
w/o indirect training	29.52	1.59	14.74	0.00	38.00	7.78	49.52	4.76
VaiBot-pretrain	30.37	36.36	32.22	50.00	38.22	20.00	49.52	19.05
w/o textual condition x, y	28.77	0.53	27.78	0.00	37.00	0.56	47.62	0.00
w/o textual condition k	18.29	36.90	10.00	44.44	24.44	19.44	31.43	28.57
w/o indirect training	28.98	2.14	26.67	0.00	40.72	4.42	48.57	3.57

Table 2: The ablation results of VaiBot on SNI and P3.

Dataset		SNI		P3	
method		seen task (90%)	unseen task (10%)	seen task (90%)	unseen task (10%)
ICL		10.91	14.44	13.22	22.86
Instruction Induction		12.80	7.37	17.00	27.62
VaiBot-in-domain	SFT	11.98	5.56	27.00	30.48
	Refined	33.37	3.33	46.89	59.05
VaiBot-pretrain	SFT	3.42	3.33	10.89	21.90
	Refined	21.39	20.00	26.56	33.33

Table 3: The inductive reasoning results of VaiBot and baselines on SNI and P3.

Bot's induction and deduction capabilities improve progressively as the volume of pretraining data increases. Notably, the deduction ability exhibits rapid growth and early convergence with increasing pretraining data, whereas the induction ability converges at a later stage. This observation aligns with the perspective highlighted in prior works (Bang et al., 2023; Tang et al., 2023; Sun et al., 2024a), which posit that "induction is harder than deduction for LLMs." These findings further validate the inherent complexity of inductive reasoning compared to deductive reasoning in the context of large language models.

5.4 Few-shot Induction

To further highlight the superiority of VaiBot, we conduct a comparative analysis between VaiBot and SFT across varying numbers of observed samples. Specifically, we train SFT with 1 to 6 observed samples and evaluate both VaiBot and SFT using the corresponding number of testing observations. As illustrated in Figure 5, VaiBot achieves nearly optimal induction performance even when observing just 1 sample, whereas SFT requires a larger number of observed samples to enhance its induction capabilities. These results underscore VaiBot's superiority in few-shot induction, demonstrating its ability to perform effectively even in one-shot induction scenarios.

6 Inductive-Deductive Collaborative Reasoning

To verify whether VaiBot can effectively collaborate the *induction* and *deduction* processes, we further consider the *inductive reasoning* task. In this task, models are asked to infer y with input x and some few-shot demonstrations $x_1, y_1; x_2, y_2; ...; x_n, y_n$. Compared with the task of deduction, *inductive reasoning* provides no task knowledge k to the model, and the model is supposed to *induce* the task knowledge from the given observations and then apply it to the test input. We adopt the following methods for comparison:

- **ICL**. We adopt in-context learning (ICL) as the basic method of inductive reasoning. Specifically, we splice the observations x_i, y_i and the input x together into a prompt: $x_1, y_1; x_2, y_2; ...; x_n, y_n; x$, and let the LLM to generate the correspond y.
- **Instruction Induction**. Instruction Induction (Honovich et al., 2023) proposed to explicitly induce textual instruction k from the observations $x_1, y_1; x_2, y_2; ...; x_n, y_n$, and then prompt

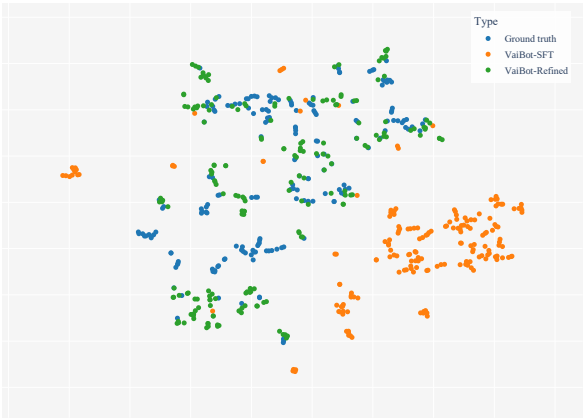

Figure 6: The t-SNE result of latent z on SNI.

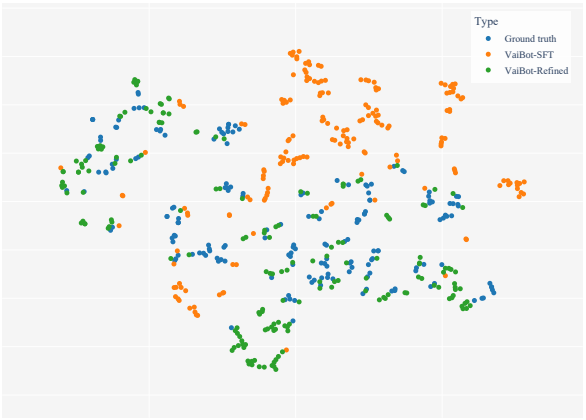

Figure 7: The t-SNE result of latent z on P3.

the LLM with the query x and instruction k to perform inductive reasoning.

- **VaiBot SFT**. With the well-trained VaiBot, First, follow the inference process in §3.3, we fine-tune the Task LLM on the demonstrations, to obtain the converged parameters z^*. We use this fine-tuned z^* for the Neural to Symbolic Inference $p_{task}(\cdot|z^*; x)$ (§3.3).
- **VaiBot Refined**. In this method, we collaborate the inductive & deductive ability of VaiBot to perform inductive reasoning. We first leverage the z^* to decode the induced task knowledge $\hat{k}$. Then, follow the inference process in §3.2, we again encoded the knowledge $\hat{k}$ into $\hat{z}$, and finally infer y with $p_{task}(\cdot|\hat{z}; x)$. Note that although we have obtained the textual knowledge k and we have proved it beneficial for deduction §5.2, we do not add it as the additional textual condition (i.e. $p_{task}(\cdot|\hat{z}; x, k)$) as we want to directly compare the quality of z.

6.1 Comparison with Baselines

As illustrated in Table 3, the direct fine-tuned z^* (VaiBot SFT) demonstrates limited effectiveness in assisting the Task LLM to predict y based on x. However, a significant improvement is observed when z^* is first decoded into $\hat{k}$ and subsequently re-encoded into $\hat{z}$. This approach substantially enhances the Task LLM's performance with $\hat{z}$, surpassing the ICL baseline by a considerable margin. These findings suggest that VaiBot effectively integrates its *deductive* and *inductive* capabilities to facilitate *inductive reasoning*.

6.2 Semantic Distribution of the Latent

To elucidate why the decode-encode collaborative process of z significantly enhances VaiBot's inductive reasoning capabilities, we generate and analyze three distinct types of z:

- **Ground truth**. We use the VaiBot to encode the annotated k of the dataset into z.
- **SFT**. The trained z^* after **VaiBot SFT**.
- **Refined**. The $\hat{z}$ that is obtained by **VaiBot Refined**.

We employ t-SNE (Van der Maaten and Hinton, 2008) for dimensionality reduction, projecting all z into a 2D plane and differentiating their types with distinct colors. As depicted in Figure 6 and Figure 7, the trained latent from SFT significantly deviates from the ground truth latent. However, by performing the induction-deduction collaborative process, the refined latent becomes markedly closer to and aligned with the ground truth (green and blue). These findings demonstrate that VaiBot effectively refines the trained latent, adapting it to better align with true semantic representations, thereby enhancing its inductive reasoning performance.

7 Conclusion

This paper proposes VaiBot, a novel neural network framework that integrates VAE and VIB, designed to uniformly model, learn, and infer both instruction deduction and instruction induction tasks of LLMs. A series of experiments are conducted to verify the effectiveness of VaiBot, which performs on par with existing baseline methods in terms of deductive capabilities while significantly surpassing them in inductive capabilities. Moreover, by combining the process of induction and deduction in VaiBot, we find that VaiBot can perform excellent inductive reasoning through refining the latent.

Limitations

The scope of deduction and induction is limited to *instruction* in this work, while other forms of task information such as *rules* may compress more difficult and informative task knowledge. We will expand and scale up VaiBot to this scope in the future.

Ethics Statement

This paper proposes VaiBot, a novel neural network framework that integrates VAE and VIB, designed to uniformly model, learn, and infer both instruction deduction and instruction induction tasks of LLMs. All experiments are conducted on publicly available datasets. Thus there is no data privacy concern. Meanwhile, this paper does not involve human annotations, and there are no related ethical concerns.

References

Alexander A Alemi, Ian Fischer, Joshua V Dillon, and Kevin Murphy. 2016. Deep variational information bottleneck. *arXiv preprint arXiv:1612.00410*.

Yejin Bang, Samuel Cahyawijaya, Nayeon Lee, Wenliang Dai, Dan Su, Bryan Wilie, Holy Lovenia, Ziwei Ji, Tiezheng Yu, Willy Chung, et al. 2023. A multitask, multilingual, multimodal evaluation of chatgpt on reasoning, hallucination, and interactivity. *arXiv preprint arXiv:2302.04023*.

Gaël Gendron, Qiming Bao, Michael Witbrock, and Gillian Dobbie. 2023. Large language models are not abstract reasoners. *arXiv preprint arXiv:2305.19555*.

Irina Higgins, Loic Matthey, Arka Pal, Christopher P Burgess, Xavier Glorot, Matthew M Botvinick, Shakir Mohamed, and Alexander Lerchner. 2017. beta-vae: Learning basic visual concepts with a constrained variational framework. *ICLR (Poster)*, 3.

Or Honovich, Uri Shaham, Samuel R. Bowman, and Omer Levy. 2023. Instruction induction: From few examples to natural language task descriptions. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1935–1952, Toronto, Canada. Association for Computational Linguistics.

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*.

Hamish Ivison, Akshita Bhagia, Yizhong Wang, Hannaneh Hajishirzi, and Matthew Peters. 2023. HINT: Hypernetwork instruction tuning for efficient zero- and few-shot generalisation. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 11272–11288, Toronto, Canada. Association for Computational Linguistics.

Diederik P Kingma. 2013. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*.

Durk P Kingma, Tim Salimans, and Max Welling. 2015. Variational dropout and the local reparameterization trick. *Advances in neural information processing systems*, 28.

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

Huanxuan Liao, Yao Xu, Shizhu He, Yuanzhe Zhang, Yanchao Hao, Shengping Liu, Kang Liu, and Jun Zhao. 2024. From instance training to instruction learning: Task adapters generation from instructions. *arXiv preprint arXiv:2406.12382*.

Suvir Mirchandani, Fei Xia, Pete Florence, Brian Ichter, Danny Driess, Montserrat Gonzalez Arenas, Kanishka Rao, Dorsa Sadigh, and Andy Zeng. 2023. Large language models as general pattern machines. *arXiv preprint arXiv:2307.04721*.

Melanie Mitchell, Alessandro B Palmarini, and Arseny Moskvichev. 2023. Comparing humans, gpt-4, and gpt-4v on abstraction and reasoning tasks. *arXiv preprint arXiv:2311.09247*.

Yiwei Qin, Kaiqiang Song, Yebowen Hu, Wenlin Yao, Sangwoo Cho, Xiaoyang Wang, Xuansheng Wu, Fei Liu, Pengfei Liu, and Dong Yu. 2024. Infobench: Evaluating instruction following ability in large language models. *arXiv preprint arXiv:2401.03601*.

Victor Sanh, Albert Webson, Colin Raffel, Stephen H Bach, Lintang Sutawika, Zaid Alyafeai, Antoine Chaffin, Arnaud Stiegler, Teven Le Scao, Arun Raja, et al. 2021. Multitask prompted training enables zero-shot task generalization. *arXiv preprint arXiv:2110.08207*.

Wangtao Sun, Haotian Xu, Xuanqing Yu, Pei Chen, Shizhu He, Jun Zhao, and Kang Liu. 2024a. Itd: Large language models can teach themselves induction through deduction. *arXiv preprint arXiv:2403.05789*.

Wangtao Sun, Chenxiang Zhang, XueYou Zhang, Xuanqing Yu, Ziyang Huang, Pei Chen, Haotian Xu, Shizhu He, Jun Zhao, and Kang Liu. 2024b. Beyond instruction following: Evaluating inferential rule following of large language models. *arXiv preprint arXiv:2407.08440*.

Xiaojuan Tang, Zilong Zheng, Jiaqi Li, Fanxu Meng, Song-Chun Zhu, Yitao Liang, and Muhan Zhang. 2023. Large language models are in-context semantic reasoners rather than symbolic reasoners. *arXiv preprint arXiv:2305.14825*.

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*.

Laurens Van der Maaten and Geoffrey Hinton. 2008. Visualizing data using t-sne. *Journal of machine learning research*, 9(11).

Ruocheng Wang, Eric Zelikman, Gabriel Poesia, Yewen Pu, Nick Haber, and Noah D Goodman. 2023. Hypothesis search: Inductive reasoning with language models. *arXiv preprint arXiv:2309.05660*.

Yizhong Wang, Swaroop Mishra, Pegah Alipoormolabashi, Yeganeh Kordi, Amirreza Mirzaei, Anjana Arunkumar, Arjun Ashok, Arut Selvan Dhanasekaran, Atharva Naik, David Stap, et al. 2022. Super-naturalinstructions: Generalization via declarative instructions on 1600+ nlp tasks. *arXiv preprint arXiv:2204.07705*.

Jeffrey Zhou, Tianjian Lu, Swaroop Mishra, Siddhartha Brahma, Sujoy Basu, Yi Luan, Denny Zhou, and Le Hou. 2023. Instruction-following evaluation for large language models. *arXiv preprint arXiv:2311.07911*.

A Instruction-following Data for Pretraining VaiBot

We collect and process the instruction-following data from the following HuggingFace datasets for the pretraining of VaiBot:

- `xzuyn/manythings-translations-alpaca`
- `MBZUAI/LaMini-instruction`
- `tatsu-lab/alpaca`
- `silk-road/alpaca-data-gpt4-chinese`
- `yizhongw/self_instruct`

B Prompts for the LLM Judge

Prompt for Deduction

Role: System
Here are an instruction, an input, an reference answer and a predicted answer. Please help me determine if the predicted answer is correct. Only return “True" or “False".
Role: User
instruction: $\{k\}$
input: $\{x\}$
reference answer: $\{y\}$
predicted answer: $\{\hat{y}\}$

Figure 8: The prompt for the external LLM to judge if the deduction result $\hat{y}$ is correct for the current case. k, x, y stands for the knowledge, the input, and the target answer of the current cases, respectively.

Prompt for Induction

Role: System
Here are two instructions described in natural language. Please help me determine if these two instructions are equivalent. Only return “True" or “False".
Role: User
transformation A: $\{k\}$
transformation B: $\{\hat{k}\}$

Figure 9: The prompt for the external LLM to judge if the induction result $\hat{k}$ is correct for the current case. k stands for the knowledge of the current cases.

C Examples of Deduction and Induction

Example for Deduction

Instruction k:
In this task, you are given a country name and you need to return the capital city of the given country

Case 1:
Input x: Angola
Prediction $\hat{y}$: The capital city of Angola is Luanda.

Target y: Luanda
Score: 1

Case 2:
Input x: Australia
Prediction $\hat{y}$: The capital city of Australia is Canberra.

Target y: Canberra
Score: 1

Case 3:
Input x: Bangladesh
Prediction $\hat{y}$: Dhaka is the capital city of Bangladesh.

Target y: Dhaka
Score: 1

Case 4:
Input x: Armenia
Prediction $\hat{y}$: Yerevan.

Target y: Yerevan
Score: 1

Case 5:
Input x: Barbados
Prediction $\hat{y}$: The capital city of Barbados is Bridgetown.

Target y: Bridgetown
Score: 1

Figure 10: An example case of deduction task.

Example for Induction

Observations:
Input x: Angola
Target y: Luanda

Input x: Australia
Target y: Canberra

Input x: Bangladesh
Target y: Dhaka

Input x: Armenia
Target y: Yerevan

Input x: Barbados
Target y: Bridgetown

Ground truth instruction k:
In this task, you are given a country name and you need to return the capital city of the given country

Predicted instruction $\hat{k}$:
Name the capital of the given country.

Score:
1

Figure 11: An example case of induction task.