# Shuttle Between Symbolic Instructions and Neural Parameters of Large Language Models

Anonymous ACL submission

## Abstract

This paper notices that while symbolic instruction and neural parameters play different roles on steering LLMs' behavior, both instructions and parameters are the compression of task data, they are supposed be strongly correlated and can be learned to predict one from the other. Therefore, This paper proposes a novel neural network framework, SHIP (**Sh**uttle between the **I**nstructions and the **P**arameters), to model and learn the bi-directional mappings between the instructions and the parameters of LLMs. We verify that SHIP can effectively map one of the instructions/parameters to the other by evaluating it on the tasks of instruction deduction and induction. The results show that SHIP performs better than existing baseline methods in terms of deductive capabilities while significantly surpassing them in inductive capabilities. Moreover, SHIP can effectively combine the two mapping processes to perform excellent inductive reasoning. We further discuss how the latent fusing methods and latent dimensions affect SHIP's performance, and show SHIP can effectively generalize with pre-training. The code and data for this paper are released at https://anonymous.4open.science/r/Shuttle-Between-Instructions-Parameters

## 1 Introduction

In the development of Large Language Models (LLMs), practitioners primarily manipulate the model's behavior (i.e., output distribution) to solve the downstream tasks via symbolic instructions and neural parameters:

- **Instructions** serve as the explicit *external* guidelines for task solving, acting as a human-readable compression of task objectives (Yin et al., 2023). Instruction is usually leveraged through Prompting to interact with LLMs, which is flexible but superficial.
- **Parameters** store the implicit *internal* encoding of task knowledge. Conversely to prompting, Supervised Fine-Tuning (SFT) fundamentally alters the model's behavior by optimizing dense weights to embed task capabilities deeply (Wei et al., 2021).

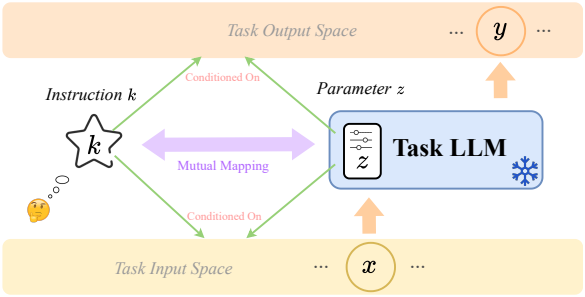


Figure 1: The basic concept of modeling the mutual mapping between symbolic instructions and neural parameters of LLMs.

Despite their distinct external representations, a deeper analysis reveals their shared intrinsic nature: instructions serve as a natural language compression devised by humans for data governing specific mapping patterns (Wang et al., 2022), whereas parameters act as a neural compression of the same task data (Delétang et al., 2023). Consequently, the instructions for a given task and the model parameters optimized on that task should theoretically exhibit a high degree of correlation. Therefore, a research question arises: **could we enable the shuttling (build the mutual mappings) between the instructions and the parameters of LLMs?**

To bridge this gap, this paper proposes SHIP (Shuttle between the Instructions and the Parameters) to unify the modeling, training, and inference of the mapping between symbolic instructions $k$ and neural parameters $z$. As shown in Figure 1, $k$ is the human-written instructions conditioned on the task data, while $z$ are some trainable parameters (detailed in Section 2 and 3.4) for the frozen Task LLM, which are also conditioned on (be trained on) the task data. Then, our core objective is to learn a set of bi-directional mappings capable of

mapping either the symbolic instructions $k$ or the neural parameters $z$ to the other.

Establishing this bi-directional mapping fundamentally transforms how we interact with LLMs, enabling two novel capabilities:

- Instructions-to-Parameters: By mapping natural language instructions to the parameter space, SHIP functions as a Hyper-network (Ivison et al., 2023), allowing for deep **adaptation** without the cost of gradient-based fine-tuning, where the model adapts its internal weights instantly based on a single instruction.
- Parameters-to-Instructions: Conversely, by mapping task-specific parameters back to the instruction space, SHIP unlocks a new form of **interpretability**. When a model's parameters are updated via SFT on new task data, SHIP can decode these parameters back into human-readable instructions. This effectively translates the "black box" learning outcome into explicit rules, revealing what the model has actually learned from the data.

Our empirical results on the Super-Natural Instructions (SNI) (Wang et al., 2022) and P3 (Sanh et al., 2021) datasets demonstrate the efficacy of SHIP. In Instructions-to-Parameters, SHIP effectively converts textual instructions into functional parameters, performing on par with standard baselines on the task of deduction (Section 3.2). More importantly, in Parameters-to-Instructions, SHIP achieves a significant improvement (over 15% on seen tasks and 10% on unseen tasks) compared to existing methods like ItD (Sun et al., 2024a) on the task of induction (Section 3.2). This confirms that the trained parameters contain rich, extractable task semantic information, that can be leveraged to interpret the task instruction. To further demonstrate the effectiveness of both the Instructions-to-Parameters and Parameters-to-Instructions mappings, this paper shows that SHIP can combine the two mappings for empowering inductive reasoning (Section 3.3). Finally, this paper discusses how latent fusion methods and latent dimensionality affect SHIP's performance, and show that SHIP can effectively generalize with pre-training.

In summary, this paper makes the following contributions:

- This paper proposes SHIP, a novel neural framework that achieves a bi-directional mapping between symbolic instructions and neural parameters of LLMs.
- This paper validates that SHIP can effectively map either the symbolic instructions $k$ or the neural parameters $z$ to the other.
- This paper further discusses how latent fusion methods and latent dimensionality affect SHIP's performance, and show that SHIP can effectively generalize with pre-training.

## 2 SHIP

In this section, we detail the proposed SHIP framework. As outlined in the Introduction section, our objective is to establish a bi-directional mapping between natural language instructions ($k$) and task-specific model parameters ($z$).

As shown in Figure 2, SHIP consists of three primary modules:

- Encoder $Enc(\cdot)$: A trainable representation encoder that maps the instruction $k$ into a probabilistic distribution in the latent space. It outputs the mean $\mu$ and covariance $\Sigma$ of a multivariate Gaussian distribution, from which the latent vector $z$ is sampled.
- Decoder $p_{dec}(\cdot)$: A trainable auto-regressive model responsible for reconstructing the instruction $k$ given the latent $z$.
- Task LLM $p_{task}(\cdot)$: A backbone frozen LLM equipped with the given task parameter $z$ that is responsible for task execution.

Here in SHIP, the latent vector $z$ is transformed and fused into the Decoder and Task LLM as lightweight adapter parameters, depending on the implementation ($z$ can take the form of Parameter Deltas, Soft Prompts (Lester et al., 2021), or LoRA weights (Hu et al., 2021), wich will be discussed later in Section 3.4).

In Section 2.1, we first show how SHIP jointly train the modules through combine the objective of VAE (Kingma, 2013) and VIB (Alemi et al., 2016). Then in Section 2.2 and 2.3, we show how SHIP leverage the bi-directional mappings for inference.

### 2.1 Joint Training Strategy

To learn the bi-directional mapping, we jointly train the Encoder and Decoder while keeping the Task LLM's base weights frozen. The training objective combines the principles of Variational Autoencoders (VAE) (Kingma, 2013) and the Variational Information Bottleneck (VIB) (Alemi et al., 2016). The training process involves a single forward pass with three distinct loss components:

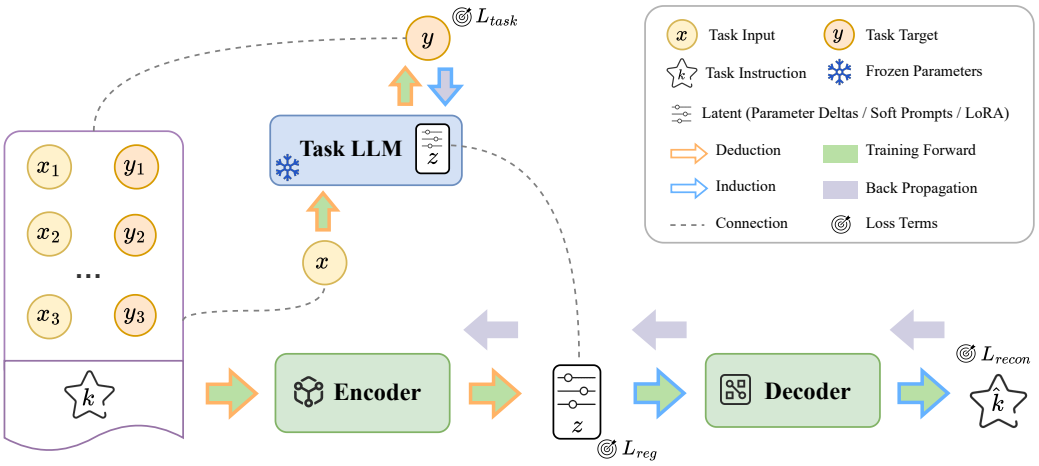


Figure 2: The framework of SHIP. The Training process is represented with filled colors and the inference process is represented with border colors.

First, SHIP uses the Encoder to encode the instruction $k$ into a high-dimension diagonal normal distribution that is parameterized by the mean $\mu$ and covariance $\Sigma$. The latent $z$ is sampled from this encoded normal distribution, where the reparametrization trick (Kingma et al., 2015) is adopted to maintain the gradient flow:

$$\begin{aligned} \mu, \Sigma &= Enc(k) \\ z &\sim \mathcal{N}(\cdot|\mu, \Sigma) \end{aligned} \tag{1}$$

Then, the Decoder attempt to reconstruct the instruction $k$ from the latent representation $z$, and calculate the reconstruction loss $L_{recon}$. This corresponds to the objective of VAE (Kingma, 2013).

$$L_{recon} = -\log p_{dec}(k|z) \tag{2}$$

Meanwhile, the latent representation $z$ is taken as the adapter parameters of the Task LLM. The Task LLM is asked to learn from the given task instance $x, y$, and calculate the task loss $L_{task}$. This corresponds to the objective of VIB (Alemi et al., 2016).

$$L_{task} = -\log p_{task}(y|z; x) \tag{3}$$

To maintain and leverage the existing well-trained natural language distribution of the Task LLM and the Decoder, in addition to latent $z$, we also provide textual condition for them: instruction $k$ for the Task LLM, and one pair of instance $x, y$ for the Decoder. So the Eq 2,3 become into:

$$L_{recon} = -\log p_{dec}(k|z; \underline{x, y}) \tag{4}$$

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

To prevent the latent $z$ from being overly complex and thus cause overfitting, we calculate the Kullback–Leibler (KL) divergence between $z$ and the standard normal distribution as the regularization loss term $L_{reg}$.

$$L_{reg} = D_{KL}(\mathcal{N}(\cdot|\mu, \Sigma)||\mathcal{N}(\cdot|0, I)) \tag{6}$$

The final objective is a weighted sum of these three terms. By minimizing this objective, SHIP learns a latent space where a single vector $z$ is both a compressed instruction representation and a functional parameter adapter.

$$L = \mathbb{E}_{(k,x,y)\sim p_{data}} w_0 L_{recon} + w_1 L_{task} + w_2 L_{reg} \tag{7}$$

## 2.2 Inference: From Instructions to Parameters

In the Instruction to Parameters inference process, the instruction $k$ is fed into the Encoder to predict the distribution parameters, from which the latent vector $z$ is sampled (Equation 1). $z$ is then injected into the frozen Task LLM as adapters.

To evaluate the quality of the generated parameters $z$, we adopt the Task LLM to perform standard auto-regressive generation to produce $\hat{y}$ based on input $x$.

$$\hat{y} \sim p_{task}(\cdot|z; x, k) \tag{8}$$

This inference process enable SHIP functions as Hyper-network to quickly adapt the Task LLM to a downstream task given its task instruction, without the need of training Task LLM with task data.

### 2.3 Inference: From Parameters to Instructions

In the Parameters to Instructions inference process, we first provide SHIP with a set of few-shot examples $T = (x_i, y_i)_{i=1}^n$ to find the optimal parameters $z^*$, and then translate $z^*$ to instruction $\hat{k}$.

The $z^*$ is obtained by optimizing the loss on the observed data:

$$z^* = \arg\min_z \frac{1}{n} \sum_{i=1}^{n} -\log p_{task}(y_i|z; x_i) \quad (9)$$

After obtaining the parameters $z^*$, 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 instructions $k$:

$$\hat{k} \sim p_{dec}(\cdot|z^*; x^*, y^*) \quad (10)$$

This inference process allow SHIP to interpret what the Task LLM actually learn after training on the data of a downstream task in a human-readable way.

## 3 Experiments

In the following sections, we first introduce the experiment settings (Section 3.1), then, we conduct a series of experiments to answer one core research question:

- **RQ:** Can SHIP learn the bi-directional mappings between the instructions and the parameters? (Section 3.2 and 3.3)

In addition, we discuss further properties of SHIP in Section 3.4:

- **D1:** Do the textual conditions benefit SHIP?
- **D2:** What is the impact of different strategies for fusing $z$ into the Task LLM?
- **D3:** How does the dimensionality of $z$ affect the performance of SHIP?
- **D4:** Do SHIP learn to generalize across tasks with pre-training?

### 3.1 Setting

**Hyper-parameters.** We employ Qwen-2.5-7b-Instruct (Team, 2024) as the base language model $M$. Task LLM is set to $M$ itself. The Encoder and Decoder are $M$ with two LoRAs (Hu et al., 2021) of rank 128 and 1, respectively. The dimension of $z$ is set to 17920 and the default fusing method of $z$ to Task LLM is to take $z$ as local Parameter Deltas (the indices of parameters are randomly selected from the entire parameters at the beginning). We will later discuss the choice of latent dimension and fusion methods in Section 3.4. All other baselines that need training (later introduced in §3.2,3.3) 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 1.0, 1.0, 1e-3, respectively.

**Dataset.** 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.

**Training.** Under the above settings, a single run of the experiment is conducted on three NVIDIA A100 GPUs. To evaluate whether SHIP can generalize through training on massive general data like LLMs do, we trained two versions of SHIP:

- **SHIP-domain**. SHIP that trained on the seen tasks of SNI and P3 for 10 epochs.
- **SHIP-pretrain**. SHIP that trained on around 437k general instruction following data (Appendix A) for 1 epoch.

In Appendix B, we demonstrate the training curves of SHIP to show that it effectively learns and converges on the given task data.

**Evaluation Tasks.** To effectively evaluate whether SHIP can map instructions and parameters to each other, we selected the following evaluation tasks:

- **Deduction**. Given the task instruction $k$ and task inputs $x$, deduction requires the model to predict the target $y$.
- **Induction**. Given multiple instances $(x_i, y_i)$ that satisfy the same instruction, induction demands the model to predict the task instruction $k$.
- **Inductive Reasoning**. In this task, models are only provided with few-shot examples $x_1, y_1; x_2, y_2; ...; x_n, y_n$ and an input $x$, and asked to infer the output $y$ based on them.

For the evaluation of the prediction results of all tasks, this paper adopts gpt-4o-mini[1] as a judge to

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

Table 1: The induction & deduction performance of SHIP 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.

| 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 * | 33.05 | 20.32 | 31.11 | 27.78 | 30.67 | 12.78 | 30.65 | 9.53 |
| vanilla SFT | 38.82 | 80.75 | 33.29 | 38.89 | 43.77 | 67.22 | 45.27 | 19.05 |
| TAGI | 38.18 | - | 38.89 | - | 46.28 | - | 45.75 | - |
| ItD | - | 83.42 | - | 33.33 | - | 74.44 | - | 14.29 |
| SHIP-domain | 39.47 | **95.72** | 37.56 | **44.44** | **50.44** | **83.33** | **56.19** | **23.81** |
| SHIP-pretrain * | **39.90** | 36.90 | **41.11** | 38.89 | 50.00 | 15.56 | 51.43 | **23.81** |

determine whether the prediction is correct (aligns with the ground truth). The prompts of three tasks for the inference model and judge are shown in the Appendix C.

### 3.2 Induction & Deduction

In this section, we first verify that SHIP can effectively map one of the instructions/parameters to the other by evaluating SHIP on separate deduction and induction tasks.

We adopt the following methods for comparison:

- **prompting**. This method simply prompts the LLM $M$ with the instructions $k$ and input $x$ to infer the target $y$ (deduction) and prompts the LLM $M$ with multiple instances $(x, y)$ to infer the instructions $k$ (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 instruction into the Task LLM through the hyper-network. It first trains the "reference" parameters of the Task LLM on the training data, and then leverages the (instruction, 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 instruction 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.
- **SHIP**. For the task of deduction, SHIP leverage its instructions-to-parameters inference (Section 2.2) to predict the task output $y$. For the task of induction, SHIP will first train the Task LLM to converge, then leverage its parameters-to-instructions inference (Section 2.3) to predict the instruction $k$.

We first compare the accuracy of SHIP on deduction and induction with baselines. As shown in Table 1, while SHIP-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 SHIP-pretrain also demonstrates competitive performance on two datasets although it is not trained on the in-domain data. These results indicate that SHIP demonstrates excellent *sample-level generalization* and *task-level generalization* abilities on both tasks of deduction and induction.

### 3.3 Inductive Reasoning

Having verified the effectiveness of SHIP to map one of the instructions/parameters to the other on separate deduction and inductions tasks, in this section, we further show that SHIP can effectively combine the two mappings for the *inductive reasoning* task. We adopt the following inductive reasoning 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 $y$.

efficient-intelligence/

Table 2: The inductive reasoning results of SHIP and baselines on SNI and P3.

| Dataset | | SNI | | P3 | |
|---|---|---|---|---|---|
| method | | seen task (90%) | unseen task (10%) | seen task (90%) | unseen task (10%) |
| ICL | | 9.41 | 10.00 | 14.67 | 22.86 |
| Instruction Induction | | 23.96 | 13.33 | 38.56 | 40.95 |
| SHIP-domain | SFT | 12.03 | 14.44 | 27.00 | 25.24 |
| | Refined | **40.32** | 24.44 | **49.67** | **57.14** |
| SHIP-pretrain | SFT | 18.56 | 15.00 | 17.56 | 23.81 |
| | Refined | 35.19 | **26.67** | 44.78 | 49.52 |

- **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 the LLM with the query $x$ and instruction $k$ to perform inductive reasoning.
- **SHIP-SFT**. With the well-trained SHIP, we fine-tune the Task LLM on the demonstrations, to obtain the converged parameters $z^*$. We use this fine-tuned $z^*$ for the inference $p_{task}(\cdot|z^*; x)$.
- **SHIP-Refined**. In this method, we combine the inductive & deductive abilities of SHIP to perform inductive reasoning. We first leverage the $z^*$ in SHIP-SFT to decode the induced instructions $\hat{k}$ (Section 2.3). Then, follow the inference process in Section 2.2, we again encoded the instruction $\hat{k}$ into $\hat{z}$, and finally infer $y$ with $p_{task}(\cdot|\hat{z}; x)$.

As illustrated in Table 2, the direct fine-tuned $z^*$ (SHIP-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 SHIP effectively integrates its *deductive* and *inductive* capabilities to facilitate *inductive reasoning*.

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

- **Ground truth**. We use the SHIP to encode the annotated $k$ of the dataset into $z$.
- **SHIP-SFT**. The trained $z^*$ after **SHIP-SFT**.
- **SHIP-Refined**. The $\hat{z}$ that is obtained by **SHIP-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 3 and Appendix E, 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 SHIP effectively refines the trained latent, adapting it to better align with true semantic representations, thereby enhancing its inductive reasoning performance.

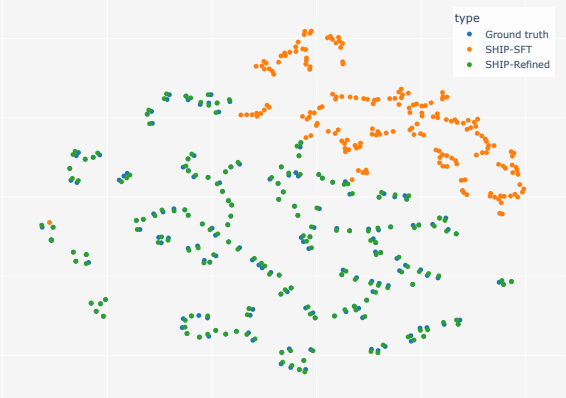


Figure 3: The latent $z$ of SHIP-domain on SNI.

### 3.4 Discussions

**Ablation of Textual Conditions**. To verify the effectiveness of the textual condition proposed in §2.1, we conduct an ablation study of SHIP by dropping these parts. As shown in Table 3, omitting the textual condition $k$ prevents the Task LLM from explicitly perceiving task instructions. However, since considerable task adaptability is still injected via the encoded $z$, SHIP's performance on deduction declines only slightly. Building on this, further omitting the textual conditions $x$ and $y$ forces the Decoder to reconstruct the task instruction $k$ relying solely on the neural $z$. This presents a

Table 3: The ablation results of SHIP on SNI and P3.

| Dataset | SNI | | | | P3 | | | |
|---|---|---|---|---|---|---|---|---|
| method | seen task (90%) | | unseen task (10%) | | seen task (90%) | | unseen task (10%) | |
| | deduction | induction | deduction | induction | deduction | induction | deduction | induction |
| SHIP-domain | **39.47** | **95.72** | **37.56** | **44.44** | **50.44** | **83.33** | **56.19** | 23.81 |
| - textual condition $k$ | 21.39 | 90.37 | 4.44 | 38.89 | 39.78 | 82.78 | 42.86 | **33.33** |
| - textual condition $k; x, y$ | 22.89 | 32.62 | 8.89 | 11.11 | 41.11 | 36.11 | 38.10 | 4.76 |

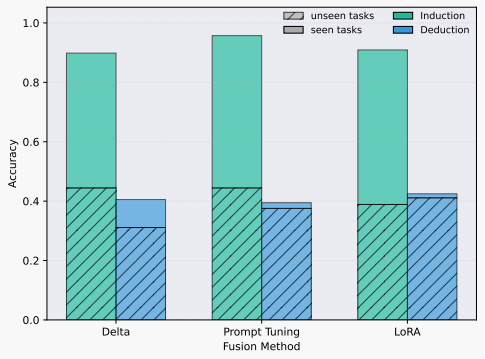


Figure 4: The induction & deduction performance of SHIP-domain using different fusion methods.

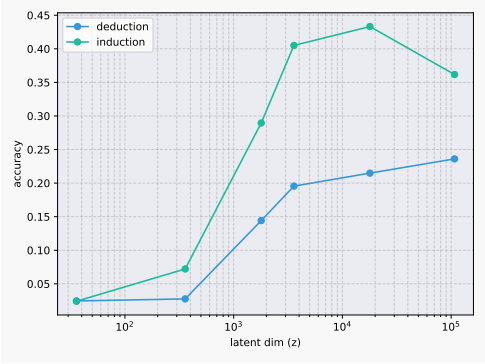


Figure 5: The induction & deduction performance of SHIP-domain with different latent dimensions.

significant challenge for the Decoder (as an LLM), resulting in a marked drop in SHIP's induction performance. Nevertheless, we observe that even in the complete absence of textual conditions, SHIP still completes deduction and induction tasks with a certain success rate. This indicates that the bidirectional mapping trained within the SHIP framework possesses substantial abstract representation capabilities, which can be integrated with textual representations (by providing textual conditions) to further enhance downstream tasks.

**Comparison of Fusion Methods**. To investigate which way of fusing $z$ into the task LLM is better, we compared the inference performance when the $z$ of the same dimensionality is incorporated into the task LLM as local Parameter Deltas (Delta), as soft prompts (Prompt Tuning, Lester et al., 2021), and as low-rank adapters (LoRA, Hu et al., 2021).

As shown in Figure 4, all fusion methods achieve excellent performance on both the induction and deduction tasks, indicating that our proposed SHIP framework can effectively accommodate different ways of fusing $z$. Comparing the approaches, p-tuning places $z$ entirely on the input side of the Task LLM and has a simpler structure, making it the easiest for the decoder to perform induction. LoRA, as a mainstream PEFT method, delivers the strongest deduction performance when fused into the Task LLM.

**Choice of Latent Dimension**. We also test different latent dimensions for the latent vector $z$ to investigate the impact of the bottleneck capacity on SHIP's performance. To more directly demonstrate the impact of $z$, we selected a version of SHIP without any textual condition for testing. As shown in Figure 5, we vary the latent dimension from $10^2$ to $10^5$ and evaluate the model on both deduction and induction tasks in SNI. The results indicate a positive correlation between the latent dimension and the model's performance. Specifically, when the dimension is low (e.g., $10^2$ or $10^3$) the limited capacity hinders the effective compression of task information, resulting in suboptimal performance. Conversely, increasing the dimension allows for a richer representation of the task semantics, leading to significant improvements. The performance begins to saturate around a dimension of $10^4$, which justifies our default setting of 17,920 to ensure sufficient capacity for the bi-directional mapping.

**Generalization with Scaling Up**. To visualize the generalization process of SHIP, we pretrain it with a weaker base model Llama-2-7b-chat (Touvron et al., 2023) 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 6. From the curve, it is evident that SHIP's induction and deduction capabilities

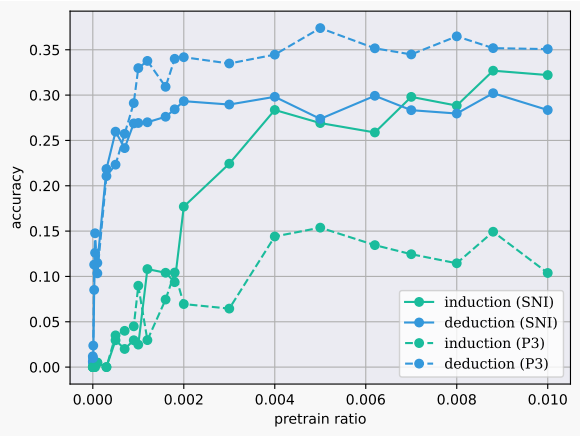


Figure 6: The OOD induction & deduction performance of SHIP-pretrain with respect to the ratio of used pretrained data.

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."

# 4 Related Work

## 4.1 Instruction-based LLM Deduction

Given the task instruction and an input, how to enable LLM to faithfully 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.

SHIP 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.

## 4.2 Instruction-oriented LLM Induction

For the sake of interpretability and generalization, some previous works also try to induce instruction from task observations through LLMs. Instruction Induction (Honovich et al., 2023) proposes to explicitly induce the task instruction from the given few-shot samples to enhance models' inductive reasoning performance. However, 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 task of induction. To improve LLMs' capability of induction, methods such as Hypothesis Search (Wang et al., 2023), Hypothesis Refinement (Qiu et al., 2023) modeled induction as a a sentence generation task, and they then proposed to filter or refine the generated instruction through verifying on observed samples or self-improvements. Further, ItD (Sun et al., 2024a) have made attempts to enhance the intrinsic inductive abilities of LLMs through self-data-augmentation and 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. SHIP introduces *parameter-based induction*, and our experiments have demonstrated that this approach significantly outperforms the previous series of data-based induction methods.

# 5 Conclusion

This paper proposes SHIP, a novel neural network framework that is designed to unify the modeling, training, and inference of the bi-directional mappings between the instruction and the parameters of the LLMs. We show that SHIP can effectively map either the symbolic instructions $k$ or the neural parameters $z$ to the other through a series of experiments on the tasks of deduction, induction, and inductive reasoning. Further discussions also illustrate SHIP's properties on different fusing methods and latent dimensions, and capability of generalization with pre-training.

## Limitations

The scope of symbols (to map to or be mapped from the neural parameters) is limited to *instruction* in this work, while other forms of symbolic task information such as *rules* may compress more difficult and informative instructions. We will expand and scale up SHIP to this scope in the future.

## Ethics Statement

This paper proposes SHIP, a novel neural network framework that integrates VAE and VIB, designed to uniformly model, learn, and infer the bi-directional mappings between symbolic instructions and neural parameters 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*.

Samuel R Bowman, Luke Vilnis, Oriol Vinyals, Andrew M Dai, Rafal Jozefowicz, and Samy Bengio. 2015. Generating sentences from a continuous space. *arXiv preprint arXiv:1511.06349*.

Grégoire Delétang, Anian Ruoss, Paul-Ambroise Duquenne, Elliot Catt, Tim Genewein, Christopher Mattern, Jordi Grau-Moya, Li Kevin Wenliang, Matthew Aitchison, Laurent Orseau, et al. 2023. Language modeling is compression. *arXiv preprint arXiv:2309.10668*.

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

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*.

Linlu Qiu, Liwei Jiang, Ximing Lu, Melanie Sclar, Valentina Pyatkin, Chandra Bhagavatula, Bailin Wang, Yoon Kim, Yejin Choi, Nouha Dziri, et al. 2023. Phenomenal yet puzzling: Testing inductive reasoning capabilities of language models with hypothesis refinement. *arXiv preprint arXiv:2310.08559*.

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*.

Qwen Team. 2024. Qwen2.5: A party of foundation models.

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*.

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*.

Wenpeng Yin, Qinyuan Ye, Pengfei Liu, Xiang Ren, and Hinrich Schütze. 2023. Llm-driven instruction following: Progresses and concerns. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing: Tutorial Abstracts*, pages 19–25.

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*.

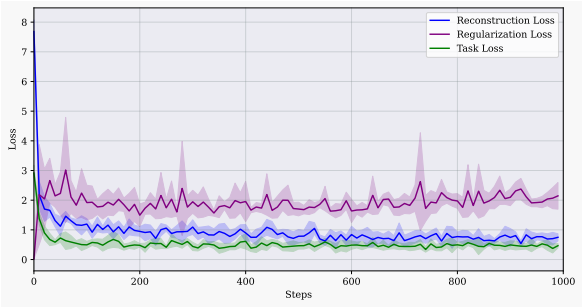


Figure 7: The loss curve of SHIP trained on SNI with respect to the training steps.

## A Instruction-following Data for Pretraining SHIP

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

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

## B Training Curve of SHIP

In Figure 7, we present the training loss curves for the three components during the training of SHIP-pretrain, plotted against the training steps. The results are averaged over 6 runs. The initially increased regularization loss (KL term) shows that SHIP is learning and storing increased latent patterns for the reconstruction and task loss. This is verified in many VAE practices (Bowman et al., 2015). The concurrent decrease in both reconstruction loss and task loss demonstrates that SHIP effectively trains the function of the Encoder and Decoder, i.e., learns the mappings between the instructions and the parameters.

## C Prompts for the LLM Judge and Baselines

## D Examples of Deduction and Induction

## E Other Latent Distribution t-SNE Results

**Prompt for the LLM Judge in Deduction and Inductive Reasoning**

**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 instruction, the input, and the target answer of the current cases, respectively.

**Prompt for the LLM Judge in 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 instruction of the current cases.

**Prompt for the Baselines in Deduction**

**Role: System**
$\{k\}$
**Role: User**
$\{x\}$

Figure 10: The prompt for the Baselines in Deduction.

**Prompt for the Baselines in Induction**

**Role: System**
I gave a friend an instruction and an input. The friend read the instruction and wrote an output for the input.
**Role: User**
Here is the input-output pair:
Input: $\{x_1\}$
Output: $\{y_1\}$
...
Input: $\{x_5\}$
Output: $\{y_5\}$
The instruction was:

Figure 11: The prompt for the Baselines in Induction.

**Prompt for the Baselines in Inductive Reasoning**

**Role: User**
$\{x_1\}$
**Role: Assistant**
$\{y_1\}$
...
**Role: User**
$\{x_5\}$
**Role: Assistant**
$\{y_5\}$
**Role: User**
$\{x\}$

Figure 12: The prompt for the Baselines in Inductive Reasoning.

**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 Sydney.
Target $y$: Canberra
Score: 0

**Case 3:**
Input $x$: Bangladesh
Prediction $\hat{y}$: New Delhi is the capital city of Bangladesh.
Target $y$: Dhaka
Score: 0

**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 13: An example case of the 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 14: An example case of the induction task.

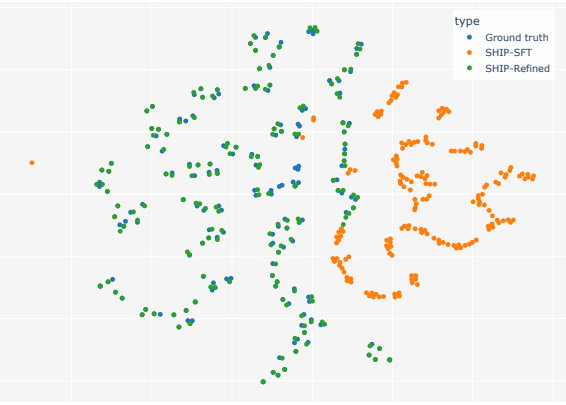


Figure 15: The latent $z$ of SHIP-domain on P3.

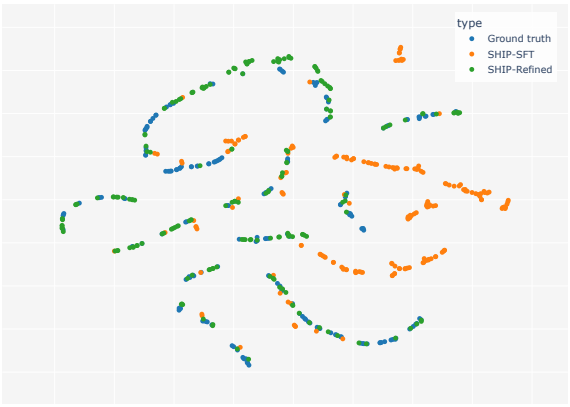


Figure 16: The latent $z$ of SHIP-pretrain on SNI.

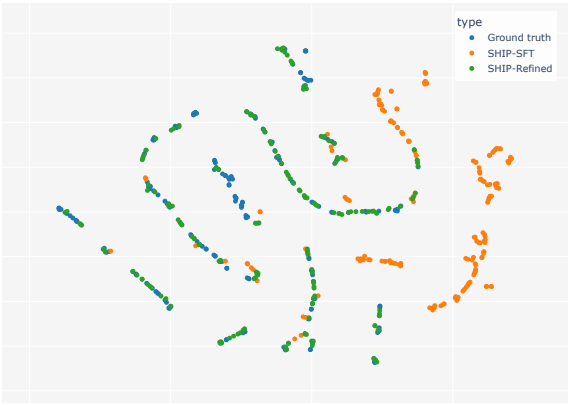


Figure 17: The latent $z$ of SHIP-pretrain on P3.