From Instance Training to Instruction Learning: Task Adapters Generation from Instructions

Huanxuan Liao[1,2], **Shizhu He**[1,2]*, **Yao Xu**[1,2], **Yuanzhe Zhang**[1,2],
Yanchao Hao[3], **Shengping Liu**[4], **Kang Liu**[1,2], **Jun Zhao**[1,2]

[1] The Key Laboratory of Cognition and Decision Intelligence for Complex Systems,
Institute of Automation, Chinese Academy of Sciences, Beijing, China
[2] School of Artificial Intelligence, University of Chinese Academy of Sciences, Beijing, China
[3] Platform and Content Group, Tencent, Beijing, China   [4] Unisound, Beijing, China
liaohuanxuan2023@ia.ac.cn {shizhu.he, yao.xu, kliu, jzhao}@nlpr.ia.ac.cn

Abstract

Large language models (LLMs) have acquired the ability to solve general tasks by utilizing instruction finetuning (IFT). However, IFT still relies heavily on instance training of extensive task data, which greatly limits the adaptability of LLMs to real-world scenarios where labeled task instances are scarce and broader task generalization becomes paramount. Contrary to LLMs, humans acquire skills and complete tasks not merely through repeated practice but also by understanding and following instructional guidelines. This paper is dedicated to simulating human learning to address the shortcomings of instance training, focusing on instruction learning to enhance cross-task generalization. Within this context, we introduce **T**ask **A**dapters **G**eneration from **I**nstructions (**TAGI**), which automatically constructs the task-specific model in a parameter generation manner based on the given task instructions without retraining for unseen tasks. Specifically, we utilize knowledge distillation to enhance the consistency between TAGI developed through *Learning with Instruction* and task-specific models developed through *Training with Instance*, by aligning the labels, output logits, and adapter parameters between them. TAGI is endowed with cross-task generalization capabilities through a two-stage training process that includes hypernetwork pretraining and finetuning. We evaluate TAGI on the Super-Natural Instructions and P3 datasets. The experimental results demonstrate that TAGI can match or even outperform traditional meta-trained models and other hypernetwork models, while significantly reducing computational requirements. Our code will be available at https://github.com/Xnhyacinth/TAGI.

1 Introduction

Large language models (LLMs) have acquired the ability to solve general tasks by utilizing instruction finetuning (IFT), which describes different tasks in the same natural language format [3; 6; 23]. However, IFT still relies heavily on instance training of extensive task data {(*Description*, *[Demonstrations]*, *Source*, *Target*)} [37; 39], which faces significant limitations in adapting LLMs to real-world scenarios where labeled task instances are scarce and broader task generalization becomes paramount.

Therefore, for better cross-task generalization, the "zero-shot" learning ability of LLMs is crucial for real-world applications: models learned with instructions can achieve non-trivial performance on unseen tasks with just a single instruction that provides a comprehensive description of the task (e.g.,

---

*Corresponding author

"*You will be given sentences in which your task is to recognize the name of a person.*"). Traditionally, achieving this capability involves meta-training the model by associating each input with specific task instructions [21; 37]. For example, GPT-3 [25] has demonstrated strong "zero-shot" capabilities through meta-training. However, these methods heavily depend on the foundation model's abilities and are inefficient for various unseen tasks [22; 44], as they require reprocessing extensive task instructions and some supplementary task data (e.g., examples from few-shot instances) for each input (see the top of Figure 1).

In recent years, researchers have begun to explore meta-learning to enhance the cross-task generalization capabilities of LLMs, aiming to construct flexible, reusable and robust task-specific models [1; 34]. For example, task-specific models such as Adapter [11], LoRA [12], and Prefix [17] have been constructed by a hypernetwork [8]. This approach significantly enhances task generalization by processing instructions efficiently, reducing redundant computations [27]. However, these methods heavily depend on a substantial corpus of training instances, which can hinder their capacity to efficiently learn and construct task-specific models based on provided instructions [13].
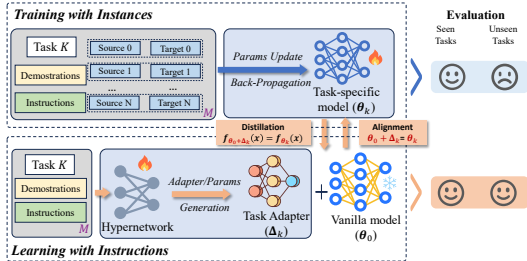

Figure 1: Comparison of the typical **Training with Instance** and the proposed **Learning with Instruction**: The former involves training the model at the instance level with parameter updates, while the latter generates a task-specific adapter at the task level with parameter generation.

In fact, contrary to LLMs, humans acquire skills and complete tasks not only through repeated practice but also by understanding and following instructional guidelines [15]. For example, a tourist with basic knowledge of riding vehicles can easily learn to use new ones abroad for the first time with the help of travel guides. This paper aims to mimic the way humans learn skills by understanding instructions. This shift represents a modest evolution in task model construction, transitioning from traditional instance training models to a contemporary approach focused on instruction learning. By providing task instructions, the novel paradigm offers an automated solution for generating task-specific adapters and seamlessly integrating them into the base model. This approach aims to streamline the development of task-specific models while enhancing their ability to generalize across diverse tasks with instructions.

Guided by this goal, we introduce **T**ask **A**dapters **G**eneration from **I**nstructions (TAGI), which converts instructions to task-specific adapters using a hypernetwork. Under the knowledge distillation framework [10; 36], we enable models to the "*Learning with Instruction*" paradigm in a manner analogous to the "*Training with Instance*" paradigm. TAGI will enhance the alignment between the task-specific model θ_k (acting as the teacher) and the vanilla LLM θ_0 combined with the generated task adapters Δ_k (acting as the student) (see the bottom of Figure 1). This alignment is achieved not only through instance training but also by incorporating parameter learning for task-specific models based on instructions. Specifically, we align the student under two distinct paradigms, encompassing not just the targets and logits, but also the adapters' parameters by an L2 regularization within instruction, which represents the enhancement of the understanding of instructions and the ability to generate more efficient task-specific adapters. Moreover, TAGI endows the model with task generalization capabilities through a two-stage training process: hypernetwork pretraining on standard text pretraining data (e.g., C4 [29]), followed by finetuning on meta-training tasks. This allows it to generalize effectively across unseen tasks without sacrificing performance.

We evaluate TAGI on the Super-Natural Instructions (SNI) [37] and P3 [30] datasets. Experimental results demonstrate its ability to effectively generate adapters for unseen tasks, surpassing meta-trained models by 2% in SNI and 5% in P3, while significantly reducing computational demands by 60%, and outperforming other hypernetwork models by 7%. Notably, our method does not require additional parameter updating or gradient back-propagation, and it avoids the inefficiency of repeatedly encoding instructions during inference. We summarize our contributions as follows:

- We propose a novel model construction paradigm by imitating human learning abilities, **Learning with Instruction**, for the cross-task generalization of the LLMs. To the best of our knowledge, it is the first time that a task-specific model has been generated based on instruction learning, and its capabilities and parameters are distilled from a teacher model trained on instance learning.

- We used a knowledge distillation framework to develop task-specific models within the instruction learning paradigm. By aligning model parameters comprehensively, the TAGI method improves the model's ability to understand instructions and solve unseen tasks more accurately and efficiently.

- Comprehensive quantitative and qualitative assessments have highlighted the effectiveness of TAGI on two publicly available large-scale instruction datasets, with lower inference costs.

2   Related Work

TAGI draws inspiration from previous research on instruction following, hypernetworks and knowledge distillation. In this section, we will delve into the pioneering work in these areas.

Instruction Following is often used to evaluate the cross-task generalization of LLMs, and it is dedicated to handling any task described in natural language. Recent findings suggest that additional finetuning of LLMs with instructions substantially improves their zero-shot capabilities [6; 38; 39]. Moreover, large-scale multi-task meta-training has been shown to equip models with the ability to address new tasks in zero- or few-shot scenarios, facilitated by standard task formats and prompts [30; 44] alongside providing concise task instructions and select examples [24; 37]. However, the instructions and examples can significantly escalate the computational burden compared to task-specific models. Existing works attempt to mitigate this issue involved creating adapters to separately process instructions and examples [13; 42] with reduced performance. To overcome these limitations, we introduce a new paradigm that draws on instruction-based learning, simulating instance training to enhance the perception and processing capabilities of LLMs for handling unseen tasks.

Hypernetworks [8; 31] are neural networks that generate weights for other neural networks [4], which are designed to use fewer parameters to dynamically build task-specific models [9; 33]. Notable works such as HyperTuning [27], HINT [13], and Hypter [42] have all adopted hypernetworks to convert task instructions and demonstrations into adapters for LLMs. And MEND [5] utilizes hypernetworks to compress demonstrations for distilled vectors. Although they all avoided processing lengthy instructions repeatedly and utilized adapters to make training and testing more cost-effective [19], they still have a performance loss compared to meta-training [7]. The proposed method TAGI incorporates the utilization of hypernetworks, which are instrumental in generating task-specific adapters that are seamlessly integrated into LLMs. Compared to existing models based on hypernetworks, TAGI not only trains at the instance level but also incorporates knowledge distillation to supervise the adapters generated by hypernetworks, thereby achieving both efficiency and effectiveness.

Knowledge Distillation is a technique where a smaller model (student) learns to mimic the predictions of a larger model (teacher), aiming to retain performance while reducing computational resources [10]. Indeed, the application of knowledge distillation is the essential difference between the proposed method in this paper and other hypernetwork-based methods such as HINT [13] and Hypter [42]. Recently, some works [32] utilize knowledge distillation to finetune small language models such as T5 [29], enabling them to act as LLMs with pre-prompting without any given prompts. Compared with the typical knowledge distillation methods of LLMs, the proposed method TAGI in this paper further utilizes model parameter alignment and aims to mimic another learning paradigm of human skill learning. We not only calculate the Kullback–Leibler (KL) divergence [14] between teacher and student models [10], but also compute the L2 regularization between the generated adapter by instruction learning and task-specific models by instance training.

3   Methods

3.1   Problem Setting

Cross-task Generalization: Given a set of tasks $\mathcal{T} = \{\mathcal{T}_1, ..., \mathcal{T}_{|\mathcal{T}|}\}$, where each task $\mathcal{T}_i$ contains a set of (*source*, *target*) samples $\mathcal{D}_i = \{(s_1, t_1), ..., (s_n, t_n)\}$. We categorize these tasks into three distinct non-overlapping groups for validating out-of-distribution generalization: meta-train ($\mathcal{T}_{train}$), meta-valid ($\mathcal{T}_{valid}$), and meta-test ($\mathcal{T}_{test}$), assuming all tasks adhere to a text-to-text format. For example, $\mathcal{T}_{train}$ comprises tasks like translation and question answering, the $\mathcal{T}_{valid}$ and $\mathcal{T}_{test}$ encompass tasks such as paraphrasing and natural language inference respectively. Within the $\mathcal{T}_{train}$, the goal is to utilize the data for training and transfer knowledge to facilitate learning to resolve the test tasks. For all methods discussed, aside from the original unsupervised pretraining of the language model
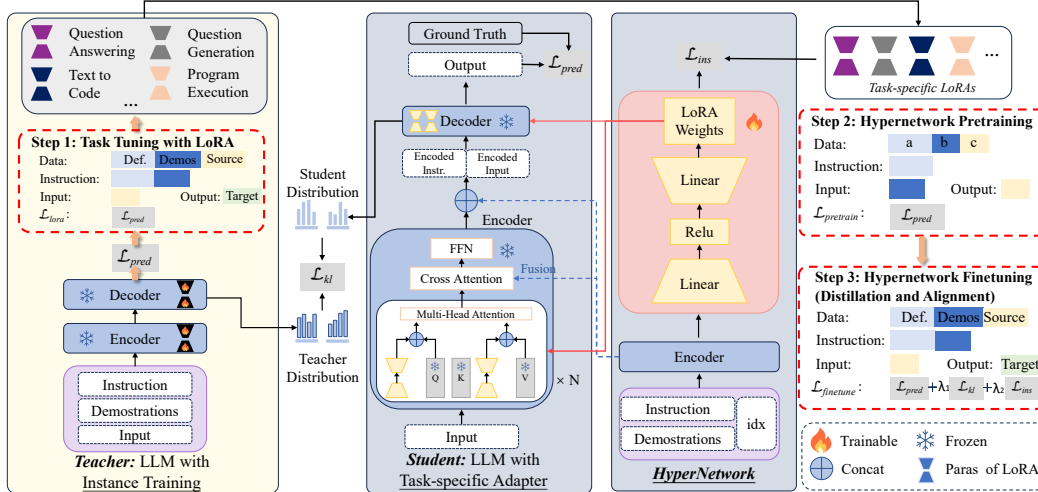
Figure 2: Overview of TAGI. The hypernetwork takes instruction as input and generates adapters subsequently integrated into the vanilla LLM, and constructed the task-specific model as student. After training the task models through instances on multiple basic tasks as a teacher, TAGI constructs task-specific models by aligning the labels, output logits, and adapter parameters between teacher and student models. To improve compliance with task instructions and the efficacy of weight generation, TAGI undergoes a two-stage hypernetwork training process: hypernetwork pretraining and finetuning. a-c are random divisions of the sampled sentences from pretraining datasets.

backbone on separate corpora, the model learning primarily takes place through multi-task training on the $\mathcal{T}_{train}$.

3.2 Task Adapters Generation from Instructions (TAGI)

In this section, we will introduce the detailed method of TAGI. For each (unseen) task, TAGI consists of two core components: a **hypernetwork** § 3.2.1 which receives task instructions and generates parameter-efficient adapters, and a task-specific model which combines the **vanilla LLM** and the generated adapters from hypernetwork.

Unlike traditional meta-training methods, we transition from *training with instance* to *learning with instruction*, which not only addresses efficiency issues at the instance level but also incorporates parameter alignment for the task-specific model parameters at the instruction level. Specifically, the complete process is shown in Figure 2, we initially train the LoRA modules § 3.2.2 on various upstream tasks (seen tasks) with task datasets of meta-train ($\mathcal{T}_{train}$). Specifically, for N distinct upstream tasks, we independently train N LoRA modules, with each module denoted as Δ_i for task $\mathcal{T}_i \in \mathcal{T}$, presumed to represent the optimal model for its respective task. Subsequently, TAGI is committed to building proprietary models for downstream tasks (unseen tasks). Its training process is bifurcated into two primary phases: hypernetwork pretraining § 3.2.3 and hypernetwork finetuning § 3.2.4 which encompasses distillation and alignment.

3.2.1 Hypernetwork for Converting Instructions into LoRA

A pivotal element of our model is the hypernetwork that converts task instructions (descriptions and demonstrations) into a parameter-efficient module. Our hypernetwork comprises two crucial components: the **encoder**, derived from the vanilla LLM[2], is designed to minimize encoding biases by converting task instructions into a continuous contextual representation. This representation is then fused with LLM input and concated with encoded input for the decoder. Additionally, the **adapter generator**, utilizing a basic MLP design [18], is both lightweight and efficient, effectively converting encoded instructions into parameter-efficient modules.

---

[2]We find that re-using the encoder from the vanilla LLM works well [13].

Encoder: Prior studies simply concatenated encoded instructions with inputs, overlooking the interactions between them. To address this, we integrated a hierarchical cross-attention layer into the encoder of the LLM to refine the input representation with embedded instruction details. Specifically, for an input x and its corresponding task instruction i_x, we initially employ the encoder within the hypernetwork to encode the instruction into representations $\mathbf{I}_x \in \mathbb{R}^{s \times d}$. Then, we feed the x into the model and obtain the output representation $\mathbf{S}_l$ from the self-attention sublayer in the l-th layer. Ultimately, $\mathbf{S}_l$ is processed through the l-th cross-attention layer, resulting in a text representation that is enriched with instruction information:

$$\mathbf{F}_l = \text{CrossAttentionLayer}_l(\mathbf{S}_l, \mathbf{I}_x) \tag{1}$$

where CrossAttentionLayer$_l$ conducts multi-head attention on the query, key, and value matrices, followed by residual connection and layer normalization. The final input to the decoder is the concatenation of the encoded instruction and the encoded fusion input, i.e., $(\mathbf{I}_x; \mathbf{F}_l)$.

Adapter Generator: Considering the efficiency and effectiveness, we utilize a two-layer multi-layer perceptron (MLP) to generate parameter-efficient modules (e.g., LoRA) for the encoded instruction. To differentiate between the query $\mathcal{Q}$ and value $\mathcal{V}$ matrices as well as the layers, we introduce layer ids $\text{idx}_l^{\{\mathcal{Q},\mathcal{V}\}} \in \{0, \ldots, 2 \times \#\text{blocks}\}$ as positional information. We use a unique network for each layer and share it between $\mathcal{Q}$ and $\mathcal{V}$ (i.e., one network is used for a certain layer LoRA generation).

$$\text{LoRA}_l^{\{\mathcal{Q},\mathcal{V}\}} = \text{MLP}_l(\mathbf{I}_{x_k}; \text{idx}_l^{\{\mathcal{Q},\mathcal{V}\}} \mid \text{idx}_l^{\mathcal{Q}} = 2l, \text{idx}_l^{\mathcal{V}} = 2l + 1) \tag{2}$$

where $\text{LoRA}_l^{\mathcal{Q}}$ and $\text{LoRA}_l^{\mathcal{V}}$ are the l-th LoRA of $\mathcal{Q}$ and $\mathcal{V}$, respectively.

3.2.2 LoRA Tuning for Task-specific Models

LoRA [12] efficiently reduces the number of trainable parameters by decomposing the update of the LLM's attention weight matrix (denoted as $\mathbf{W}_0 \in \mathbb{R}^{d \times k}$) into low-rank matrices. Specifically, LoRA updates the weight matrix as $\mathbf{W}_0 + \delta\mathbf{W} = \mathbf{W}_0 + \mathbf{AB}$, with $\mathbf{A} \in \mathbb{R}^{d \times r}$ and $\mathbf{B} \in \mathbb{R}^{r \times k}$ being trainable low-rank matrices of rank r, significantly smaller in dimensions than d and k. We finetune a robust baseline to derive the LoRA parameters Δ_i for task-specific models for i-th task, facilitating LLM instruction learning and parameter alignment. SNI is categorized into 60 types based on task types, while P3 encompasses 36 categories, corresponding to 60 and 36 parameter modules, respectively.

3.2.3 Hypernetwork Pretraining for Preliminary Generalization

Previous research [5; 27] has demonstrated that pretraining hypernetworks can substantially improve the model's cross-task generalization capabilities. Adhering to the HINT [13], we pretrain the hypernetwork on C4 [29] before finetuning it on a diverse multi-task prompt dataset. As illustrated in the right segment of Figure 2, given an input sequence, we partition it into randomly sized segments a, b, and c, where a is fed into the hypernetwork, b into the LLM, and c is the segment to predict. During this stage, training is conducted by minimizing the cross-entropy loss $\mathcal{L}_{\text{pred}}$, aiming to ensure that the hypernetwork learns to recognize instructions to enhance generalization ability.

$$\mathcal{L}_{\text{pred}} = \log P_{(\text{LLM+Hypernetwork}(a))}(c \mid b) \tag{3}$$

3.2.4 Hypernetwork Finetuning for Instruction Learning

At this stage, TAGI is finetuned on a multi-task prompt dataset, enabling it to learn the generation of optimal parameters from task instructions, thereby ensuring effective generalization to future unseen tasks. Similar to the pretraining phase, task instructions (alongside some few-shot samples) replace a, the main input replaces b, and the target replaces c. In each iteration, the hypernetwork generates LoRA parameters and encodes the instructions. LoRA is a parameter-efficient module (i.e., inserting into the model), and the encoded instructions are integrated with the encoder's embeddings for information fusion and concatenated with the fused encoding input during decoding. Beyond the standard $\mathcal{L}_{\text{pred}}$, we employ knowledge distillation for instruction learning: a strong baseline combining complete task instructions and input, serves as the teacher, while the model incorporating generated LoRA parameters with the input, acts as the student. The KL divergence $\mathcal{L}_{\text{kl}}$ measures the discrepancy in word probability distributions between the two models as an implicit learning outcome, and the MSE loss $\mathcal{L}_{\text{ins}}$ calculates the difference between the generated parameters and

those of task-specific parameter-efficient modules as an explicit learning intermediate result. The formulation of finetuning is as follows:

$$\mathcal{L}_{\text{ins}} = \text{MSE}(\Delta_i, \text{Hypernetwork}(a)) \tag{4}$$

$$\mathcal{L}_{\text{kl}} = \text{KL}(P_{(\text{LLM}+\Delta_i)}(x \mid (a;b)) \parallel P_{(\text{LLM}+\text{Hypernetwork}(a))}(x \mid b)) \tag{5}$$

$$\mathcal{L}_{\text{finetune}} = \mathcal{L}_{\text{pred}} + \lambda_1 \mathcal{L}_{\text{kl}} + \lambda_2 \mathcal{L}_{\text{ins}} \tag{6}$$

where $a \in \mathcal{T}_i$, Δ_i is the optimal LoRA modules of the i-th task, λ_1 and λ_2 are the hyper-parameter to control the importance of distillation in finetuning.

4 Experiments

We first present the datasets (§ 4.1) and baselines (§ 4.2) used in our evaluation and then discuss three research questions (RQs):

RQ1: Can the proposed instruction learning paradigm effectively learn the ability of instance training? Can it support cross-task generalization of LLMs? (§ 4.4)

RQ2: How many foundation tasks does TAGI need to learn to achieve better results? (§ 4.5)

RQ3: What is the impact of different modules and learning stages on TAGI? (§ 4.7)

4.1 Datasets

To demonstrate the generality of our method, we evaluate our approach on two popular multi-task instruction datasets[3]: **Super-Natural Instructions (SNI)** [37] and **T0 split of P3 (P3)** [30].

SNI comprising over 1,600 task datasets, each dataset includes a task definition and a set of fixed positive and negative demonstrations. We follow the previous research [13; 27] and examine two settings: only using the task definition as the input to the hypernetwork (**'Def'**), and using the definition along with two few-shot positive examples (**'Def + 2 Pos'**). We only use the English tasks in the dataset and the model's generation is evaluated on a set of 119 unseen tasks using ROUGE-L.

P3 composed of 62 task datasets, the T0 model is trained with these tasks divided into meta-training and meta-test sets. The format of the prompts takes into consideration 0-shot reasoning and typically includes instructions or possible answer options. We follow the precedent work [41] by using the T0 training subset 36 tasks to train our model. The evaluation is conducted based on the accuracy scores of multiple-choice questions for unseen 11 tasks in the meta-test set (MTest11).

4.2 Baselines

We compare the characteristics of **TAGI** against eight primary groups of baselines (as shown in Table 1): *1)* **No FT**: models without finetuning. *2)* **HyperTuning** [27]: models that use hypernetwork to convert demonstrations into adapters without instruction fusion. *3)* **Hypter** [42]: models based on hypernetwork do not use pretraining. *4)* **HINT** [13]: models pretrain hypernetwork and concat instruction. *5)* **T0** and **Tk-Instruct**: strong baselines fully finetuned on P3 and SNI respectively with instruction concatenated. *6)* **Full FT**: models

Table 1: Compare the characteristics of all comparison methods and the proposed TAGI. More comparisons can be seen in C.1.

Method	Pre-Train	Instr. Fus.	Low Infer. Cost	Instr. Learning	Unseen Task
Simple FT	✗	✔	✗	✗	✗
T0 [30] / Tk-Instruct [37]	✗	✔	✗	✗	✔✔✔
Hypter [42]	✗	✗	✔	✗	✔
HyperTuning [27]	✔	✗	✔	✗	✔
HINT [13]	✔	✗	✔	✗	✔✔
TAGI (Ours)	✔	✔	✔	✔	✔✔✔

fineuned on target tasks. *7)* **Decoder-only model**: decoder-only models fully finetuned like GPT-2 [28] and OPT [43]. *8)* **FiD-ICL** [41]: ICL method use encoder intermediate fusion.

---

[3]We provide the full list of datasets and more details in the A.2.

Table 2: RougeL results on Super-Natural Instructions. The best results are in bold, while the second-best are underlined. ∗, † means that those results are from HINT [13] and [27] respectively, "-" means not reported. ‡ indicates that there is no parameter alignment loss in the hypernetwork finetuning because the model is too large, leading to a significant amount of time required for LoRA tuning for each task. The Average Relative FLOPs cost is calculated relative to Tk-Instruct. We use the number of FLOPs required by each model to process one task (containing 100 examples).

| Method | Def (Zero-shot) | | | Def + 2 Pos. (Few-shot) | | | Avg. Rel. |
	Base (250M)	XL (3B)	XXL (11B)	Base (250M)	XL (3B)	XXL (11B)	FLOPs
No FT	8.8	14.3	26.2	9.4	13.6	30.5	×1.0
Tk-Instruct†	**35.3**	<u>48.0</u>	**53.6**	<u>42.1</u>	54.0	**62.0**	×1.0
# Decoder-only model							
GPT-2 XL (1.5B)*	-	38.2	-	-	45.3	-	**×0.33**
OPT (13B)*	-	-	44.8	-	-	51.5	×0.36
# Hypernetwork-based model							
Hypter*	12.1	16.8	15.5	10.6	14.2	13.4	×0.35
HyperTuning†	-	38.9	-	-	48.6	-	×<u>0.34</u>
HINT*	<u>33.3</u>	47.2	51.1	41.8	53.2	56.4	×0.37
TAGI (Ours)	**35.3**	**48.4**	<u>52.3</u> ‡	**42.5**	**56.3**	<u>58.4</u> ‡	×0.39

4.3 Implementations

We limit our scope to encoder-decoder models for our experiments[4]. We use T5-LM-Adapt[5] and T0 [30] as initializations in our experiments. The two model groups have the same architectural framework but differ in weight; T0 uses T5-LM-Adapt for initialization and undergoes multi-task training on the P3 meta-training set. For **SNI**, only T5-LM-Adapt is considered, and three different sizes are tested: Base (250M), XL (3B), and XXL (11B), with the teacher model being TK-Instruct [37]. For **P3**, we experimented with two sets of models of three different sizes: Base (250M), Large (800M), and XL (3B) with the only template as input, while the teacher model being FiD-ICL [41] with 16-shot examples. The A.4 contains more implementation details and experimental settings.

4.4 Main Results

Super-Natural Instructions. We report the performance and inference costs of TAGI models and baselines in Table 2. Our analysis and findings yield several key insights:

• Firstly, **methods lacking finetuning exhibit subpar performance**. As shown in the first row of the table, the performance of No FT is significantly lower than other baseline methods by approximately 30 points (except for Hypter), which underscores the critical role of inductive bias, introduced during meta-training, in enhancing the model's instructional adherence and cross-task generalization.

• Secondly, **TAGI demonstrates notable improvements over other hypernetwork-based baselines, with only a marginal increase in inference overhead** (see Table 2 last column). We find that TAGI still outperforms the advanced method HINT (≥ 2 points) while achieving similar computational savings. This highlights the efficacy of instruction learning with knowledge distillation. The underperformance of HINT and Hypertuning may stem from their sole reliance on cross-entropy with the target during meta-training, lacking explicit supervision of intermediate task-specific module parameters and implicit supervision of the teacher outcome. This deficiency impedes their ability to fully leverage instruction tasks for generating superior adapter parameters during meta-test.

• Thirdly, **TAGI consistently matches or even surpasses robust baselines in both zero- and few-shot settings**. Comparing TAGI with multi-task finetuning approaches such as Full FT and TK-Instruct, we observe that TAGI achieves comparable performance (0 − 2.3 points) except for 11B while utilizing approximately 2.5 × fewer FLOPs. TAGI's performance on the 11B model is somewhat lacking, potentially attributable to either insufficient training due to resource limitations or a decrement in performance stemming from the omission of parameter alignment constraints due to time constraints[6]. In alignment with prior research, TAGI significantly surpasses GPT-2 and OPT-13B in comparative analyses with decoder-only models (≥ 10 points in GPT2 and ≥ 7 points

---

[4]We have discussed in detail the encoder-decoder and decoder-only models in B.1.

[5]https://huggingface.co/google/t5-xl-lm-adapt

[6]We discuss the trend and possible reasons in B.2

Table 3: Average accuracy results over T0 evaluation tasks after training on the T0 P3 train set. α means results are from [41]. $^\heartsuit$ trained by us followed the Tk-Instruct (meta-training) [37]. Our method uses only template inputs without demonstrations yet achieves competitive performance with ICL-based methods using 16 shots, with much-reduced inference overhead. The Average Relative Inference Time is calculated relative to the Metatrain. We use the inference time required by each model to process all 11 test tasks with batch_size of 1.

Method	T5-LM			T0			Avg. Rel. Infer. Time
	Base (250M)	Large (800M)	XL (3B)	Base (250M)	Large (800M)	XL (3B)	
# MTest11 Avg.							
Zero-shot	43.9	41.5	42.6	49.1	52.4	57.6	×1.0
Full FT	44.6	45.5	47.2	**51.9**	**56.6**	**61.4**	×1.0
Metatrain $^\heartsuit$	44.1	52.4	53.1	50.1	52.4	56.8	×1.0
# ICL-based method							
Concat-ICL$^\alpha$	44.2	47.6	-	48.6	53.2	-	×4.1
FiD-ICL$^\alpha$	**47.0**	**55.2**	**60.0**	51.0	53.4	58.2	×1.9
Ensemble-ICL$^\alpha$	44.6	54.5	52.6	49.9	53.7	57.7	×13.2
# Hypernetwork-based model							
Hypter*	-	-	-	-	-	56.2	-
HINT*	-	-	-	-	-	60.3	-
TAGI (Ours)	45.6	54.7	58.9	50.8	53.8	58.8	**×0.88**
# HyperT5 Avg. (Without SCloze dataset)							
FiD-ICL$^\alpha$	**46.9**	**55.8**	60.6	**51.7**	53.9	58.5	×1.9
HyperTuning†	-	54.6	59.6	-	-	-	-
TAGI (Ours)	46.7	56.0	59.8	**51.7**	54.6	59.2	**×0.88**

in OPT-13B), affirming the superiority of encoder-decoder models within similar meta-learning frameworks. Overall, TAGI fulfills its objective by enhancing cross-task generalization capabilities through instruction learning and striking an optimal balance between performance and efficiency.

P3. We report results on the T0 evaluation set in Table 3, with full results in C.2.

• Firstly, examining the ICL-based methods presented in the middle section, it is evident that all three ICL aggregation strategies achieve superior performance. This underscores the utility of instructions and demonstrations in aiding LLMs. However, these methods require concatenating extensive demonstrations during both training and inference, which significantly increases computational demands and reduces efficiency (×2 - ×13.2 inference time). In contrast, **TAGI by leveraging solely task instructions one time, attains comparable or superior accuracy levels while significantly curtailing computational burdens (×0.88)**. TAGI demonstrates a slight disadvantage (merely 1.2 points) to FiD-ICL [41] on T5-LM, yet it outperforms other methods (≥ 1 point). For T0, it is only 1.5 points lower than Full FT and exceeds all ICL-based methods. Notably, TAGI does not require the 16 examples like the ICL-based method, nor does it necessitate repeated processing of instructions like the baselines, significantly reducing inference overhead.

• A comparison of the first three lines of results indicates that for large or XL models, **initializing with T5-LM outperforms T0**. We hypothesize that the process of training T5-LM to transition into T0 might result in the dilution of world knowledge or the diminishment of certain specific capabilities, thereby attenuating the benefits derived from meta-training. Conversely, for models of base size, T0 serves as a more effective initialization point.

• Furthermore, **TAGI outperforms competing hypernetwork models**[7]. By comparing the last two columns, it is evident that the performance in MTest11 surpasses HINT and Hypertuning by 0.5 and 4.6 points respectively. Additionally, in the HyperT5 evaluation, the performance exceeds Hypertuning by 1 point. This aligns with prior findings, suggesting that instruction learning augments the hypernetwork's task comprehension and its capacity to generate task-specific adapters.

4.5 Varying Number of Meta-Training Tasks

A fundamental component of our methodology is incorporating parameter alignment in instruction learning. Consequently, it is imperative to examine the effect of varying the number of tasks on which

---

[7]Because HINT is designed for TPU and Hypertuning is not open-sourced, we didn't calculate their inference time. However, based on SNI experiments, it can be inferred that the trend of time expenditure is consistent.
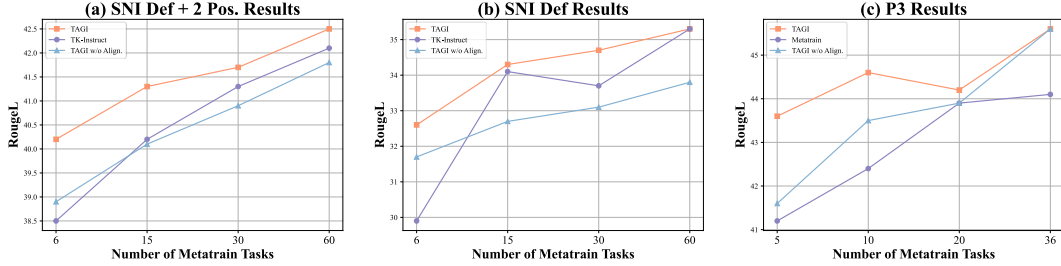
Figure 3: The performance of different numbers of meta-training tasks. The backbone model is T5-LM-Base, all trained for 20,000 steps.
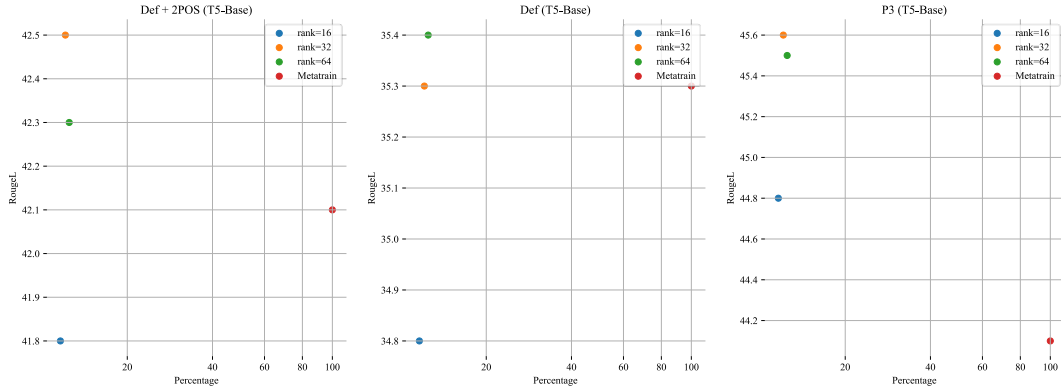

Figure 4: The percentage of generated parameters (%) against performance (RougeL). The backbone model is T5-LM-Base, all trained for 20,000 steps.

parameter alignment is applied on outcomes and its influence on the generalization capabilities of LLMs. To this end, we conduct a comprehensive experimental analysis to compare the efficacy of instruction learning with parameter alignment across a spectrum of task quantities against instruction learning devoid of parameter alignment. Tasks are organized in descending order based on the number of datasets encompassed within each. Subsequently, a predetermined number of tasks are sequentially selected for meta-training purposes. This approach allows us to systematically evaluate the impact of parameter alignment on learning and generalization as the number of tasks varied.

From Figure 3, we find that, firstly, **an increase in the number of tasks correlates with improved performance across all methods**, suggesting that meta-training across a broader array of tasks enhances the model's instruction-following capabilities. However, the practical limitations of sourcing a sufficient quantity of tasks for meta-training must be acknowledged. Secondly, it was observed that the TAGI model exhibits lower overall performance in the absence of parameter alignment for instruction learning, yet it demonstrates a smaller relative standard deviation and less variability in performance in response to the number of tasks. This pattern aligns with the expected outcomes of instruction learning, highlighting the **efficacy of our approach in bolstering the model's ability to adhere to task instructions and generate task-specific adapters**.

4.6 Parameter Size against Performance

We analyzed the proportion of generated parameter sizes relative to the total parameter size during the generation of various ranks, and compared this to the performance of the full meta-training fine-tuning method, as demonstrated in Figure 4 and Table 7. We can find that **TAGI requires only about 10% of the parameters to outperform full meta-training fine-tuning which indicates that the limited parameters generated by the Hypernetwork serve as an optimal solution for task completion**. The ability to adaptively construct models tailored to specific tasks removes the necessity for additional fine-tuning, underscoring TAGI's effectiveness and efficiency.

9

4.7 Ablation Study

To evaluate the significance of each component within the TAGI model, we conducted a series of experiments across two meta-task datasets utilizing the T5-LM-XL (3B) model. The results as depicted in the Table 4, highlight that the **instructions fusion plays a pivotal role in enhancing model performance**. This process facilitates dynamic interaction between the input and the instructions, enriching the model's input with additional contextual information, reminiscent of the substantial benefits observed with ICL. Moreover, **pretraining emerges as a critical phase**, markedly improving the capabilities of models that have not undergone pretraining, thereby significantly enhancing their proficiency in interpreting and executing task instructions. Furthermore, the systematic removal of **various components during the finetuning phase indicates a consistent decline in performance**, underscoring the integral contribution of each component to the model's overall efficacy.

Table 4: Ablation study of TAGI model. All models utilized are T5-LM-XL (3B) and training for 20,000 steps. The P3 dataset was selected by the HyperT5 evaluation.

Method	Def	Def + 2Pos.	P3
TK-Instruct	48.0	54.0	-
TK-Instruct-LoRA	47.5	54.6	-
TK-Instruct-Prefix	42.6	54.2	-
Hypertuning	38.9	48.6	59.6
HINT	47.2	53.2	60.3
TAGI	**48.4**	**56.3**	**60.6**
Ablation Study			
w/o pretraining	47.1	55.6	58.3
w/o Instr. Fus.	35.1	40.6	44.2
w/o $\mathcal{L}_{ce}$	47.6	55.4	59.8
w/o $\mathcal{L}_{kl}$	45.7	53.9	57.3
w/o $\mathcal{L}_{ins}$	47.5	55.2	59.4
w/o Hypernetwork	43.8	50.7	-

Compared to meta-learning methods such as LoRA fine-tuning (rank=32) "Tk-Instruct-LoRA", prefix fine-tuning (num_virtual_tokens=32) "Tk-Instruct-prefix", and full fine-tuning "Tk-Instruct", our TAGI method enhances task comprehension and utilization which achieved through a hypernetwork that dynamically generates adapter LoRA insertions into the LLM based on input, leads to better cross-task generalization capabilities. Notably, prefix fine-tuning excels in the Def + 2Pos scenario, likely due to its effective integration of information from positive examples. Conversely, the Def scenario performs less satisfactorily, indicating that instructions alone are insufficient for optimal results. Comparative analysis with other hypernetwork models reveals that TAGI's ablation performance remains robust, affirming the effectiveness of each step in bolstering TAGI's operational efficiency.

5 Conclusions

In this paper, we introduce an innovative method of instruction learning designed to emulate instance training. This approach enables the model to achieve specified tasks and learn from instructions on how to address a category of problems. The proposed TAGI seamlessly integrates instruction into the input and processes the instruction simultaneously, thereby ensuring minimal inference overhead. Concurrently, we employ a knowledge distillation framework to facilitate instruction learning for distilling skills and aligning task-specific models. This allows the hypernetwork to transform task instructions into an efficient module inserted into the LLMs, thereby boosting generalization performance. Remarkably, TAGI consistently equals or surpasses the efficacy of conventional meta-training approaches while requiring fewer FLOPs and obviating the need for additional model parameters updating or gradient back-propagation. Future work will investigate more potent hypernetwork pretraining techniques and develop superior instruction fusion methods to augment the hypernetwork's expressive capability, thereby enhancing the model's ability to generalize to unseen tasks. Moreover, future work will investigate various task type classifications and the generalization effects of cross-modal tasks in instruction learning.

6 Acknowledgements

References

[1] Jonathan Baxter. *Learning to Learn*. Springer US, 1998.

[2] Christos Baziotis, Mikel Artetxe, James Cross, and Shruti Bhosale. Multilingual machine translation with hyper-adapters, 2022.

[3] Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel Ziegler, Jeffrey Wu, Clemens Winter, Chris Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. Language models are few-shot learners. In H. Larochelle, M. Ranzato, R. Hadsell, M.F. Balcan, and H. Lin, editors, *Advances in Neural Information Processing Systems*, volume 33, pages 1877–1901. Curran Associates, Inc., 2020.

[4] Vinod Kumar Chauhan, Jiandong Zhou, Ping Lu, Soheila Molaei, and David A. Clifton. A brief review of hypernetworks in deep learning, 2023.

[5] Tong Chen, Qirun Dai, Zhijie Deng, and Dequan Wang. Demonstration distillation for efficient in-context learning, 2024.

[6] Hyung Won Chung, Le Hou, Shayne Longpre, Barret Zoph, Yi Tay, William Fedus, Yunxuan Li, Xuezhi Wang, Mostafa Dehghani, Siddhartha Brahma, Albert Webson, Shixiang Shane Gu, Zhuyun Dai, Mirac Suzgun, Xinyun Chen, Aakanksha Chowdhery, Alex Castro-Ros, Marie Pellat, Kevin Robinson, Dasha Valter, Sharan Narang, Gaurav Mishra, Adams Yu, Vincent Zhao, Yanping Huang, Andrew Dai, Hongkun Yu, Slav Petrov, Ed H. Chi, Jeff Dean, Jacob Devlin, Adam Roberts, Denny Zhou, Quoc V. Le, and Jason Wei. Scaling instruction-finetuned language models, 2022.

[7] Budhaditya Deb, Guoqing Zheng, and Ahmed Hassan Awadallah. Boosting natural language generation from instructions with meta-learning, 2022.

[8] David Ha, Andrew Dai, and Quoc V. Le. Hypernetworks, 2016.

[9] Yun He, Huaixiu Steven Zheng, Yi Tay, Jai Gupta, Yu Du, Vamsi Aribandi, Zhe Zhao, YaGuang Li, Zhao Chen, Donald Metzler, Heng-Tze Cheng, and Ed H. Chi. Hyperprompt: Prompt-based task-conditioning of transformers, 2022.

[10] Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531*, 2015.

[11] Neil Houlsby, Andrei Giurgiu, Stanislaw Jastrzebski, Bruna Morrone, Quentin de Laroussilhe, Andrea Gesmundo, Mona Attariyan, and Sylvain Gelly. Parameter-efficient transfer learning for nlp, 2019.

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

[13] Hamish Ivison, Akshita Bhagia, Yizhong Wang, Hannaneh Hajishirzi, and Matthew Peters. Hint: Hypernetwork instruction tuning for efficient zero-shot generalisation. *ACL*, 2023.

[14] James M. Joyce. *Kullback-Leibler Divergence*, pages 720–722. Springer Berlin Heidelberg, Berlin, Heidelberg, 2011.

[15] Sharon Kim, Mahjabeen Raza, and Edward Seidman. *Improving 21st-century teaching skills: The key to effective 21st-century learners*. Springer US, 2019.

[16] Quentin Lhoest, Albert Villanova del Moral, Yacine Jernite, Abhishek Thakur, Patrick von Platen, Suraj Patil, Julien Chaumond, Mariama Drame, Julien Plu, Lewis Tunstall, Joe Davison, Mario Šaško, Gunjan Chhablani, Bhavitvya Malik, Simon Brandeis, Teven Le Scao, Victor Sanh, Canwen Xu, Nicolas Patry, Angelina McMillan-Major, Philipp Schmid, Sylvain Gugger, Clément Delangue, Théo Matussière, Lysandre Debut, Stas Bekman, Pierric Cistac, Thibault Goehringer, Victor Mustar, François Lagunas, Alexander Rush, and Thomas Wolf. Datasets: A community library for natural language processing. In Heike Adel and Shuming Shi, editors, *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 175–184, Online and Punta Cana, Dominican Republic, November 2021. Association for Computational Linguistics.

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

[18] Huanxuan Liao, Shizhu He, Yao Xu, Yuanzhe Zhang, Kang Liu, Shengping Liu, and Jun Zhao. Imagination augmented generation: Learning to imagine richer context for question answering over large language models. *arXiv preprint arXiv:2403.15268*, 2024.

[19] Haokun Liu, Derek Tam, Mohammed Muqeeth, Jay Mohta, Tenghao Huang, Mohit Bansal, and Colin Raffel. Few-shot parameter-efficient fine-tuning is better and cheaper than in-context learning, 2022.

[20] Shayne Longpre, Le Hou, Tu Vu, Albert Webson, Hyung Won Chung, Yi Tay, Denny Zhou, Quoc V. Le, Barret Zoph, Jason Wei, and Adam Roberts. The flan collection: Designing data and methods for effective instruction tuning, 2023.

[21] Sewon Min, Mike Lewis, Luke Zettlemoyer, and Hannaneh Hajishirzi. MetaICL: Learning to learn in context. In Marine Carpuat, Marie-Catherine de Marneffe, and Ivan Vladimir Meza Ruiz, editors, *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 2791–2809, Seattle, United States, July 2022. Association for Computational Linguistics.

[22] Swaroop Mishra, Daniel Khashabi, Chitta Baral, Yejin Choi, and Hannaneh Hajishirzi. Reframing instructional prompts to gptk's language, 2022.

[23] Swaroop Mishra, Daniel Khashabi, Chitta Baral, and Hannaneh Hajishirzi. Cross-task generalization via natural language crowdsourcing instructions. In *ACL*, 2022.

[24] Swaroop Mishra, Daniel Khashabi, Chitta Baral, and Hannaneh Hajishirzi. Cross-task generalization via natural language crowdsourcing instructions, 2022.

[25] Long Ouyang, Jeff Wu, Xu Jiang, Diogo Almeida, Carroll L. Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, John Schulman, Jacob Hilton, Fraser Kelton, Luke Miller, Maddie Simens, Amanda Askell, Peter Welinder, Paul Christiano, Jan Leike, and Ryan Lowe. Training language models to follow instructions with human feedback, 2022.

[26] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Köpf, Edward Yang, Zach DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. *PyTorch: an imperative style, high-performance deep learning library*. Curran Associates Inc., Red Hook, NY, USA, 2019.

[27] Jason Phang, Yi Mao, Pengcheng He, and Weizhu Chen. Hypertuning: Toward adapting large language models without back-propagation, 2022.

[28] Alec Radford, Jeff Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. Language models are unsupervised multitask learners. 2019.

[29] Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. Exploring the limits of transfer learning with a unified text-to-text transformer. *Journal of Machine Learning Research*, 21(140):1–67, 2020.

[30] Victor Sanh, Albert Webson, Colin Raffel, Stephen H. Bach, Lintang Sutawika, Zaid Alyafeai, Antoine Chaffin, Arnaud Stiegler, Teven Le Scao, Arun Raja, Manan Dey, M Saiful Bari, Canwen Xu, Urmish Thakker, Shanya Sharma Sharma, Eliza Szczechla, Taewoon Kim, Gunjan Chhablani, Nihal Nayak, Debajyoti Datta, Jonathan Chang, Mike Tian-Jian Jiang, Han Wang, Matteo Manica, Sheng Shen, Zheng Xin Yong, Harshit Pandey, Rachel Bawden, Thomas Wang, Trishala Neeraj, Jos Rozen, Abheesht Sharma, Andrea Santilli, Thibault Fevry, Jason Alan Fries, Ryan Teehan, Tali Bers, Stella Biderman, Leo Gao, Thomas Wolf, and Alexander M. Rush. Multitask prompted training enables zero-shot task generalization, 2022.

[31] Jürgen Schmidhuber. Learning to control fast-weight memories: An alternative to dynamic recurrent networks. *Neural Computation*, 4:131–139, 1992.

[32] Charlie Snell, Dan Klein, and Ruiqi Zhong. Learning by distilling context, 2022.

[33] Yi Tay, Zhe Zhao, Dara Bahri, Donald Metzler, and Da-Cheng Juan. Hypergrid transformers: Towards a single model for multiple tasks. In *International Conference on Learning Representations*, 2021.

[34] Sebastian Thrun and Lorien Y. Pratt. Learning to learn: Introduction and overview. In *Learning to Learn*, 1998.

[35] Thomas Wang, Adam Roberts, Daniel Hesslow, Teven Le Scao, Hyung Won Chung, Iz Beltagy, Julien Launay, and Colin Raffel. What language model architecture and pretraining objective work best for zero-shot generalization?, 2022.

[36] Wenhui Wang, Furu Wei, Li Dong, Hangbo Bao, Nan Yang, and Ming Zhou. Minilm: Deep self-attention distillation for task-agnostic compression of pre-trained transformers. *Advances in Neural Information Processing Systems*, 33:5776–5788, 2020.

[37] Yizhong Wang, Swaroop Mishra, Pegah Alipoormolabashi, Yeganeh Kordi, Amirreza Mirzaei, Atharva Naik, Arjun Ashok, Arut Selvan Dhanasekaran, Anjana Arunkumar, David Stap, Eshaan Pathak, Giannis Karamanolakis, Haizhi Lai, Ishan Purohit, Ishani Mondal, Jacob Anderson, Kirby Kuznia, Krima Doshi, Kuntal Kumar Pal, Maitreya Patel, Mehrad Moradshahi, Mihir Parmar, Mirali Purohit, Neeraj Varshney, Phani Rohitha Kaza, Pulkit Verma, Ravsehaj Singh Puri, Rushang Karia, Savan Doshi, Shailaja Keyur Sampat, Siddhartha Mishra, Sujan Reddy A, Sumanta Patro, Tanay Dixit, and Xudong Shen. Super-NaturalInstructions: Generalization via declarative instructions on 1600+ NLP tasks. In Yoav Goldberg, Zornitsa Kozareva, and Yue Zhang, editors, *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pages 5085–5109, Abu Dhabi, United Arab Emirates, December 2022. Association for Computational Linguistics.

[38] Jason Wei, Maarten Bosma, Vincent Zhao, Kelvin Guu, Adams Wei Yu, Brian Lester, Nan Du, Andrew M. Dai, and Quoc V Le. Finetuned language models are zero-shot learners. In *International Conference on Learning Representations*, 2022.

[39] Orion Weller, Nicholas Lourie, Matt Gardner, and Matthew E. Peters. Learning from task descriptions. In Bonnie Webber, Trevor Cohn, Yulan He, and Yang Liu, editors, *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1361–1375, Online, November 2020. Association for Computational Linguistics.

[40] Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Remi Louf, Morgan Funtowicz, Joe Davison, Sam Shleifer, Patrick von Platen, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven Le Scao, Sylvain Gugger, Mariama Drame, Quentin Lhoest, and Alexander Rush. Transformers: State-of-the-art natural language processing. In Qun Liu and David Schlangen, editors, *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 38–45, Online, October 2020. Association for Computational Linguistics.

[41] Qinyuan Ye, Iz Beltagy, Matthew Peters, Xiang Ren, and Hannaneh Hajishirzi. FiD-ICL: A fusion-in-decoder approach for efficient in-context learning. In Anna Rogers, Jordan Boyd-Graber, and Naoaki Okazaki, editors, *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 8158–8185, Toronto, Canada, July 2023. Association for Computational Linguistics.

[42] Qinyuan Ye and Xiang Ren. Learning to generate task-specific adapters from task description, 2021.

[43] Susan Zhang, Stephen Roller, Naman Goyal, Mikel Artetxe, Moya Chen, Shuohui Chen, Christopher Dewan, Mona Diab, Xian Li, Xi Victoria Lin, Todor Mihaylov, Myle Ott, Sam Shleifer, Kurt Shuster, Daniel Simig, Punit Singh Koura, Anjali Sridhar, Tianlu Wang, and Luke Zettlemoyer. Opt: Open pre-trained transformer language models, 2022.

[44] Ruiqi Zhong, Kristy Lee, Zheng Zhang, and Dan Klein. Adapting language models for zero-shot learning by meta-tuning on dataset and prompt collections, 2021.

[45] Ahmet Üstün, Arianna Bisazza, Gosse Bouma, Gertjan van Noord, and Sebastian Ruder. Hyper-x: A unified hypernetwork for multi-task multilingual transfer, 2022.

A Experimantal Settings

A.1 Problem Setting

Meta-Training and Inference: Our methodology rigorously adheres to the protocol outlined in MetaICL [21]. In the meta-train phase, we commence by selecting a task $\mathcal{T}$ from $\mathcal{T}_{train}$, followed by the sampling of k support examples $\{(x_i^{(s)}, y_i^{(s)})\}$ and m query examples $\{(x_i^{(q)}, y_i^{(q)})\}$ from the chosen task. The proposed hypernetwork is then adjusted to minimize the overall loss, focusing on generating a task model that can accurately predict the target sequences (e.g., answer) for source sequences (e.g. question). During the meta-test/inference phase, for each novel task in $\mathcal{T}_{test}$, we employ instructions to create the task-specific adapter, to optimize the model's performance across all query examples $\{(x_i^{(q)}, y_i^{(q)})\}$.

Table 5: Number of samples in given splits for each dataset.

Dataset	Examples per Task	Train	Test
Super-Natural Instructions	100	75,417	11,810
P3	-	90,897,454	2,940,068
P3 (Sampling)	1000	290,000	2,940,068

A.2 Datasets

During the pretraining phase, we utilized the C4 dataset [29], truncating each sequence to 1024 tokens. For the training phase, we employed Super-Natural Instructions (SNI) [37] and P3 datasets [30] for meta-training and meta-test. For SNI, we adhered to the default settings [13; 37], which include 100 examples per task for both the training and test splits. For P3, we used the data and prompts provided by T0. All prompts related to the meta-training tasks were included in the meta-training process, while the meta-test phase utilized evaluation prompts specified by T0 [30]. We treated ANLI R1, R2, and R3 as three distinct tasks, resulting in 11 tasks for the original meta-test in P3 (Meta-Test-11). Due to resource constraints, we deviated from the sampling procedures of prior work, opting to sample 1000 examples per task for each prompt template. This approach yielded a smaller dataset size, as detailed in Table 5. For further information on P3 refer to [30]. Additionally, to facilitate comparison with the Hypertuning method, we excluded the StoryCloze task from the evaluation since it was not included in the datasets for the HyperT5 evaluation.

A.3 Split Sizes for Varying Number of Meta-Training Tasks

As shown in Table 11 and Table 12, we present a comprehensive list of the two datasets, including the number of tasks or templates contained in each task and the task divisions from § 4.5 experiments. The divisions in the table are cumulative; thus, the second division includes both the first and the second divisions. For SNI, tasks were sorted in descending order based on the number of tasks they contained and then divided into specified sizes (6, 15, 30, 60). For P3, we selected a specified number of tasks (5, 10, 20, 36) based on the task classification in the original paper, which includes categories such as Multiple-Choice QA, Closed-Book QA, Summarization, Structure-To-Text, Paraphrase Identification, Sentiment, Topic Classification, and Extractive QA.

We obtain all our data from huggingface datasets [16]. In the following, we provide the dataset links:

- Super-Natural Instructions: https://github.com/allenai/natural-instructions
- P3: https://huggingface.co/datasets/bigscience/P3

Additionally, the Super-Natural Instructions dataset (previously known as Natural Instructions-v2) has undergone some changes over time. In our experiments, we use the v2.6 version.

A.4 Implementations

Our implementations are based on huggingface transformers v4.23.1 [40] using PyTorch v1.13.1 [26] and deepspeed[8] v0.10.0. All experiments were conducted on 4 A100 NVIDIA GPUs, each equipped with 80GB of memory, and eight A6000 NVIDIA GPUs with 48GB of memory. Unless otherwise specified, the rank of LoRA generated by the hypernetwork is 32, and we use the Adamw optimizer with a learning rate of 5e-5 and a linear warmup rate of 0.02. We pre-train all models for 50,000 steps using C4 [29] with a batch size of 8 samples and sequences of length 1024.

A.5 T0-Base/Large/3B

T0 [30] provides model checkpoints only in sizes 3B and 11B. Additionally, HINT [13] and FiD-ICL [41] re-pretrained T0 and found that the model was not sufficiently trained, achieving better results after reproduction. Therefore, we used the T0 model [9] reproduced by FiD-ICL to conduct a series of experiments.

Table 6: Hyperparameters for Training TAGI Models and LoRA Tuning.

| | LoRA Tuning | Pretraining | Finetuning | | | | | |
| | | | SNI | | | P3 | | |
			Base (250M)	XL(3B)	XXL (11B)	Base (250M)	Large (800M)	XL(3B)
Max Input Len	1024	1024	1024	1024	1024	512	512	512
Max Output Len	128	-	128	128	128	64	64	64
Optimizer	adamw	adafactor	adamw	adamw	adamw	adamw	adamw	adamw
Learning Rate	1e-4	1e-3	1e-4	5e-5	5e-5	1e-4	1e-4	5e-5
precision	bf16	float32	bf16	bf16	bf16	bf16	bf16	bf16
# Training Steps	10000	50000	20000	20000	20000	20000	20000	20000
# Warmup Steps	-	-	# 2% of total training steps					
Batch Size	8	8	8	2	1	8	4	2
Gradient Accumulation	2	1	2	4	2	2	4	4
LoRA Rank	# 32							

A.6 Hyperparameter

The complete stable hyperparameter set used for training runs can be found in Table 6.

B Additional Experiments and Findings

B.1 Why we choose Enc-Dec Models?

Previous work has suggested that models with an encoder-decoder (enc-dec) structure have advantages over decoder-only (dec-only) models in terms of task generalization and instruction-following capabilities [20; 35; 41]. Therefore, in our experiments, we only considered models with an enc-dec structure (T5-LM and T0). Our experimental results demonstrated that enc-dec models indeed have an advantage when compared, although dec-only models might have higher computational efficiency due to their ability to cache KV and have fewer layers. However, our method, TAGI, significantly improves performance in various aspects with only a slight increase in computational overhead. We encode the task instructions only once based on the original computation.

B.2 T5-LM-XXL Training Trend

In this section, we detail how the performance of the T5-LM-XXL (11B) model surpasses the hypernetwork models but falls short of the meta-trained strong baseline Tk-Instruct by 1-4 points, as mentioned earlier in § 4.4. The primary reason is insufficient training; when replicating the Tk-Instruct experiment, our results were significantly lower than reported when finetuning for only

---

[8]https://github.com/microsoft/DeepSpeed
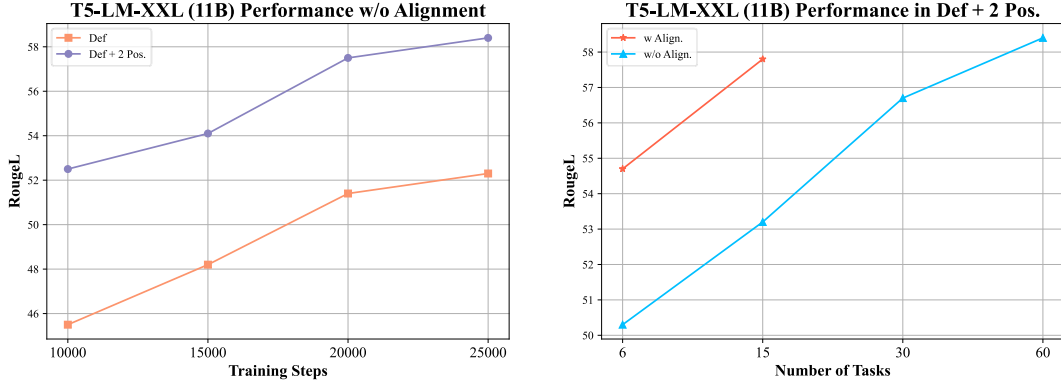[9]https://huggingface.co/qinyuany/fid-icl-t0-large

Figure 5: Analysis of T5-LM-XXL (11B).

20,000 steps. Consequently, we analyzed the performance of our TAGI model at different finetuning steps. As shown on the left side of Figure 5, performance steadily improves with more steps with substantial growth. Thus, we reasonably predict that increasing the steps to 50,000 or more could surpass Tk-Instruct. Another possible reason is the lack of parameter alignment for the 11B model due to limited resources. Our previous analysis has shown that parameter alignment is crucial, with larger models benefiting more. Therefore, we analyzed performance with a small number of tasks for parameter alignment. As shown on the right side of Figure 5, performance with parameter alignment for 6 and 15 tasks is better than without alignment. Based on these trends, it can be inferred that performance with full task parameter alignment could surpass Tk-Instruct.

B.3 Analysis on Hyperparameters

To explore the optimal hyperparameter settings for our experiments, we conducted a series of tests and error analyses using the T5-LM-Base (800M) model. The findings presented in Table 7 reveal that variations in hyperparameters can lead to performance fluctuations, particularly with higher learning rates or reduced finetuning steps. Given the varying pre-training conditions of models of different sizes, a size-specific analysis is essential; however, details on larger models are omitted here due to resource limitations.

We observed that different settings of LoRA minimally affect performance, leading us to select a balanced size of 32. Similarly, the impact of the warmup ratio is negligible; thus, based on our experience, we chose a warmup ratio of one percent of the maximum finetuning steps. While more finetuning steps generally correlate with improved performance, excessive finetuning can result in overfitting on meta-training tasks, thereby diminishing generalizability. Moreover, increased finetuning steps require greater computational resources. Consequently, we determined that the optimal number of finetuning steps is 20,000 based on our experimental outcomes.

B.4 How λ_1 and λ_2 are tuned?

In the experiment, we set λ_1 and λ_2 to two different values: $\lambda_1 = 5$ and $\lambda_2 = \text{sigmoid}(\mathcal{L}_{\text{ins}})$. The effects of these different λ values on the results are illustrated in Figure 6 and Table 8. We maintained all other conditions constant and only varied λ to perform an ablation experiment at Def+2Pos. scenario.

B.5 Inference Cost

To analyze the computational efficiency of the TAGI model compared to the standard instruction training model (full fine-tuning), let's consider a scenario where we have to process n samples, each of length i, along with a task instruction of length t. We assume the output sequence length is negligible and thus ignore it in our computations.

In a typical full fine-tuning setup, such as Tk-Instruct, each input is concatenated with the task instruction, requiring the model to process the combined input sequence. If we denote the number

16

Table 7: Performance variation due to different hyperparameters. The base model is T5-LM-Base, and all experiments follow the previous hyperparameter settings, changing only the target hyperparameter, where underlines indicate experimental defaults.

| Method	Learning Rate				LoRA Rank			Training Steps			Warmup Ratio		
	5e-5	1e-4	3e-4	1e-3	16	32	64	15000	20000	25000	0.01	0.02	0.03
SNI													
Def + 2 Pos.													
Tk-Instruct [37]	41.3	41.8	42.2	38.9	-	-	-	41.4	41.8	42.1	41.5	41.8	40.6
TAGI (Ours)	42.1	42.5	40.3	39.7	41.8	42.5	42.3	41.8	42.5	42.4	42.3	42.5	41.9
Def													
Tk-Instruct [37]	35.0	34.2	32.6	31.7	-	-	-	34.4	34.2	34.5	35.0	34.2	34.3
TAGI (Ours)	34.3	35.3	33.5	31.8	34.8	35.3	35.4	34.2	35.3	35.4	34.8	35.3	34.9
P3													
MTest11 Avg.													
Metatrain	43.3	44.1	43.6	40.9	-	-	-	44.0	44.1	44.3	44.2	44.1	43.6
TAGI (Ours)	44.0	45.6	44.0	41.6	44.8	45.6	45.5	44.3	45.6	45.2	45.1	45.6	44.8
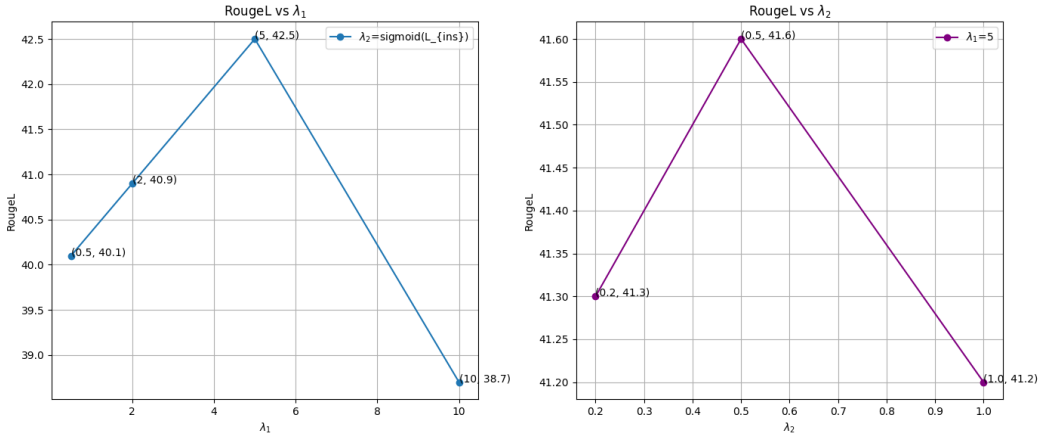
Figure 6: Ablation study on λ hyperparameters. The backbone model is T5-Base.

of FLOPs required to process a single token with an encoder-decoder model as N, where N is the total number of model parameters, then the total computation cost for all samples can be estimated as: $\text{FLOPs}_{\text{standard}} = N \cdot n(t + i)$ Here, each of the n samples includes both the instruction and the sample input, leading to $n(t + i)$ tokens being processed.

Our TAGI model, on the other hand, processes the task instruction only once, regardless of the number of samples. This unique feature significantly reduces the computation required, especially as the number of samples or the length of the instruction increases. The total computation cost in this model is given by: $\text{FLOPs}_{\text{TAGI}} = N \cdot (t + ni)$ In this case, the instruction length t is processed only once, and each sample is processed separately, resulting in a total of $(t + ni)$ tokens being processed.

C   Extended Results

C.1   Characteristics Comparison of the Proposed TAGI and Other Baselines

Here, we report a full comparison of methods and the proposed TAGI in Table 9, also visualized in Table 1. In this report, we compare various methods across eight dimensions. **Finetuning on target tasks** yields good performance; however, it necessitates retuning when applied to unseen tasks and fails to address these effectively. **Strong baseline meta-training methods** excel at handling unseen tasks by enabling models to solve problems based on task-specific instructions. Nevertheless, these methods are limited to instance-level operations and entail repetitive processing of concatenated instructions and comprehensive finetuning, resulting in significant parameter updates and high inference costs.

Table 8: Ablation study on λ hyperparameters. The backbone model is T5-Base.

λ_1	λ_2	RougeL
0.5	$sigmoid(\mathcal{L}_{\text{ins}})$	40.1
2	$sigmoid(\mathcal{L}_{\text{ins}})$	40.9
5	$sigmoid(\mathcal{L}_{\text{ins}})$	**42.5**
10	$sigmoid(\mathcal{L}_{\text{ins}})$	38.7
5	0.2	41.3
5	0.5	41.6
5	1.0	41.2

Hypter [42] initially introduced the approach of considering tasks at the task level, treating identical tasks as a unified entity, and employing a hypernetwork to generate adapters that represent specific task models from instructions. Building on this, **Hypertuning** [27] uses demonstrations to generate adapters and pretrains the hypernetwork to boost its expressive capabilities. Both strategies avoid the direct input of instructions and rely on hypernetwork, which reduces parameter updates and lowers computational demands during inference. However, they suffer from notable performance degradation due to the lack of instructional information in the input.

HINT [13] addresses this issue by appending instructions post-encoder, thus eliminating redundant computations. Although these methods facilitate learning at the task level, they do not engage in instruction-based learning, i.e., they do not explicitly supervise the hypernetwork's generation process to aid in understanding instructions and generating parameters.

The proposed TAGI rectifies these deficiencies by integrating cross-attention for enhanced information fusion and supervised learning of adapter weights within HINT. This innovation aids in generalizing to unseen tasks without increasing the computational burden.

Table 9: Compare the characteristics of all comparison methods and the proposed TAGI.

Method	Meta-Train	Pre-Train	Instr. Concat.	Instr. Fus.	Low Up. Params	Low Infer. Cost	Instr. Learning	Unseen Task
Simple FT	✗	✗	✔	✔	✗	✗	✗	✗
T0 [30] / Tk-Instruct [37]	✔	✗	✔	✔	✗	✗	✗	✔✔✔
Hypter [42]	✔	✗	✗	✗	✔	✔	✗	✔
HyperTuning [27]	✔	✔	✗	✗	✔	✔	✗	✔
HINT [13]	✔	✔	✔	✗	✔	✔	✗	✔✔
TAGI (Ours)	✔	✔	✔	✔	✔	✔	✔	✔✔✔

C.2 P3 Full Results

Table 10 reports the per-task performance and average accuracy on P3 reported in Table 3.

D Limitations

Large Language Models. Due to computational constraints, most of our experiments were conducted using models with $\leq 3B$ parameters. Given the complexity of our research, we restricted our focus to encoder-decoder models, which have demonstrated superior performance in cross-task generalization [35], which we explore further in B.1. Consequently, it remains uncertain whether instruction learning can be effectively scaled to larger models ($\geq 7B$ parameters) or commonly used decoder-only models. However, since our method preserves the original model parameters without compromising performance, we anticipate its applicability to broader research in the future.

Training Costs. Although TAGI is computationally efficient during inference, its training cost is significantly higher. This is due to the additional requirements beyond the foundation laid by previous work, including the introduction of knowledge distillation, running a hypernetwork to generate adapters for each batch, and pre-training some downstream task-specific models. Consequently, while

Table 10: Main Full P3 Results. "-" means not reported. † and ‡ mean the results are from FiD-ICL [41] and Hypertuning [27] respectively. ◇ Computed as the average of R1/R2/R3 (except for HyperT5 rows where the numbers are quoted). More ICL-based results and details can be seen FiD-ICL [41].

Method	ANLI◇	(R1)	(R2)	(R3)	HSwag	CB	COPA	RTE	WiC	WSC	WGD	SCloze	MTest11 Avg.	HyperT5 Avg.
Random	33.4	33.4	33.4	33.4	25.0	50.0	50.0	52.7	50.0	63.5	50.0	50.0	44.7	46.8
# Base(250M)														
T5-LM †	33.4	33.3	33.5	33.5	24.7	44.3	54.3	47.9	49.7	57.9	49.8	54.1	43.9	45.2
T5-LM Full FT †	33.8	34.5	33.4	33.5	24.8	66.5	45.7	51.1	53.7	46.3	49.8	50.9	44.6	46.5
T5-LM Metatrain	31.0	30.3	29.5	33.1	25.0	40.5	52.6	51.2	50.2	58.4	47.4	66.6	44.1	44.6
T5-LM-FiD †	33.0	32.4	33.1	33.4	26.7	42.5	58.8	54.6	51.1	57.9	50.3	76.3	47.0	46.9
T5-LM-TAGI	32.1	31.5	31.7	33.1	25.0	44.5	54.7	53.7	52.3	60.5	50.8	64.0	45.6	46.7
T0 †	32.3	31.5	32.4	33.1	26.5	45.8	65.9	69.3	51.6	56.7	51.2	76.1	49.1	49.9
T0 Full FT †	33.5	32.6	33.9	33.9	29.1	73.2	66.3	68.0	53.1	50.9	51.0	79.0	51.9	53.1
T0 Metatrain	32.1	31.5	31.5	33.2	29.5	50.4	64.2	68.2	47.7	61.6	52.8	80.8	50.1	50.8
T0-FiD †	32.7	31.7	32.9	33.6	26.2	54.9	68.2	68.1	51.9	60.3	51.3	82.3	51.0	51.7
T0-TAGI	32.7	31.1	31.9	35.0	29.8	49.3	67.1	70.0	49.0	61.2	54.4	79.6	50.8	51.7
# Large(800M)														
T5-LM †	32.7	32.1	33.4	32.7	25.3	33.8	50.5	49.0	51.0	50.4	50.5	47.8	41.5	42.9
T5-LM Full FT †	34.1	35.1	33.6	33.6	26.1	65.4	47.1	51.7	53.7	47.5	49.9	56.5	45.5	46.9
T5-LM Metatrain	31.3	30.0	30.5	33.4	27.0	60.4	77.6	71.9	47.0	56.4	54.8	87.2	52.4	53.3
T5-LM-FiD †	34.4	33.9	33.4	35.8	28.3	60.2	81.1	72.6	50.7	63.7	55.6	91.6	55.2	55.8
T5-LM-TAGI	33.7	33.5	32.5	35.1	27.8	62.9	79.0	76.1	52.9	57.9	58.2	86.2	54.7	56.0
T0 †	34.1	32.2	34.2	36.0	26.1	56.8	76.6	65.3	50.8	56.4	53.9	88.4	52.4	52.5
T0 Full FT †	35.3	34.5	35.4	36.2	33.1	80.1	80.8	69.2	54.1	53.2	56.3	90.0	56.6	57.8
T0 Metatrain	32.9	31.5	31.8	35.5	24.5	59.4	77.0	65.1	48.8	56.7	57.6	88.0	52.4	52.8
T0-FiD †	33.4	31.8	32.8	35.7	26.1	60.7	77.6	67.1	52.1	59.1	54.7	89.5	53.4	53.9
T0-TAGI	32.7	31.5	32.9	36.6	27.3	61.3	79.6	68.7	48.2	59.9	56.4	89.4	53.8	54.6
HyperT5-Prefix ‡	33.4	-	-	-	32.3	60.1	73.9	71.5	51.1	63.0	51.1	-	-	54.6
HyperT5-LoRA ‡	33.6	-	-	-	33.0	49.5	74.2	67.4	52.0	64.0	52.9	-	-	53.3
# XL(3B)														
T5-LM †	32.7	32.2	33.4	32.7	24.6	32.7	53.1	48.8	50.8	57.6	50.9	51.4	42.6	43.9
T5-LM Full FT †	34.6	35.5	34.3	33.9	27.1	67.8	54.8	50.7	53.7	47.7	50.7	63.3	47.2	48.4
T5-LM Metatrain	32.7	31.5	32.3	34.3	33.3	59.5	74.8	69.5	52.6	53.8	54.2	88.4	53.1	53.8
T5-LM-FiD †	39.3	39.8	37.6	40.4	31.4	67.0	92.3	78.8	50.4	64.5	61.2	96.5	60.0	60.6
T5-LM-TAGI	37.7	37.8	36.1	39.3	32.0	68.2	89.4	76.6	53.6	61.2	59.6	94.2	58.9	59.8
T0 †	38.0	38.4	35.7	40.0	26.5	67.7	82.2	80.1	53.5	57.3	57.8	94.0	57.6	57.9
T0 Full FT †	38.5	37.5	38.8	39.2	38.7	81.9	88.0	80.1	55.9	59.5	61.4	95.0	61.4	63.0
T0 Metatrain	37.0	37.3	33.2	40.4	24.8	66.9	81.9	78.9	52.7	60.2	55.6	92.8	56.8	57.3
T0-FiD †	38.6	39.0	36.5	40.5	28.5	62.9	87.4	74.6	52.1	62.7	61.0	95.5	58.2	58.5
T0-TAGI	38.7	39.5	35.6	41.0	26.5	68.7	87.8	78.2	52.2	61.8	59.8	95.6	58.8	59.2
HyperT5-Prefix ‡	38.7	-	-	-	33.6	69.6	88.4	79.5	53.1	57.6	56.6	-	-	59.6
HyperT5-LoRA ‡	35.3	-	-	-	30.8	66.4	83.3	68.5	50.3	60.0	56.1	-	-	56.4

TAGI may be highly efficient for inference and suitable for users with limited resources, training a unique TAGI model presents considerable challenges.

Datasets. In the SNI study, our investigation was limited to tasks in English, leaving the generalization capabilities in a multilingual context unexplored. However, given the proven effectiveness of hypernetwork methods in achieving multilingual generalization [2; 45], we are optimistic about the potential directions for our future research in this domain. Furthermore, in P3, we adopted the methodologies of T0 [30] and FiD-ICL [41], concentrating primarily on natural language processing (NLP) tasks amenable to ranking classification. This focus included tasks related to classification and multiple-choice questions but excluded other types of generative tasks. Looking ahead, we aim to develop new research resources and broaden our experimental scope and evaluations to encompass a more diverse array of categories.

Table 11: Meta-Train dataset of Super-Natural Instructions.

Task	# Num of Task
First Split (6 Tasks)	
Question Answering	157
Program Execution	90
Question Generation	51
Sentiment Analysis	42
Misc.	36
Toxic Language Detection	32
Second Split (15 Tasks)	
Text Categorization	28
Commonsense Classification	23
Text Matching	17
Named Entity Recognition	17
Information Extraction	17
Wrong Candidate Generation	15
Text Completion	14
Question Understanding	13
Text to Code	12
Third Split (30 Tasks)	
Summarization	12
Dialogue Generation	11
Word Semantics	10
Story Composition	9
Speaker Identification	9
Pos Tagging	9
Linguistic Probing	9
Fill in The Blank	8
Text Quality Evaluation	7
Stereotype Detection	7
Sentence Composition	7
Negotiation Strategy Detection	7
Gender Classification	7
Coherence Classification	6
Word Relation Classification	5
Fourth Split (60 Tasks)	
Explanation	5
Text Simplification	4
Sentence Perturbation	4
Paraphrasing	4
Mathematics	4
Intent Identification	4
Dialogue State Tracking	4
Code to Text	4
Sentence Ordering	3
Fact Verification	3
Answer Verification	3
Translation	2
Style Transfer	2
Stance Detection	2
Speaker Relation Classification	2
Question Decomposition	2
Number Conversion	2
Irony Detection	2
Grammar Error Detection	2
Spelling Error Detection	1
Spam Classification	1
Sentence Expansion	1
Sentence Compression	1
Punctuation Error Detection	1
Preposition Prediction	1
Poem Generation	1
Entity Relation Classification	1
Entity Generation	1
Discourse Relation Classification	1
Discourse Connective Identification	1

Table 12: P3 dataset tasks. † means evaluation without story_cloze.

Task	# Num of Prompts
Meta-Train	
First Split (5 Tasks)	
cosmos_qa	13
kilt_tasks_hotpotqa	5
amazon_polarity	9
cnn_dailymail_3.0.0	9
common_gen	9
Second Split (10 Tasks)	
glue_mrpc	7
adversarial_qa_dbert	5
ag_news	7
dream	5
gigaword	9
Third Split (20 Tasks)	
paws	12
wiki_qa	11
ropes	12
quoref	11
dbpedia_14	4
multi_news	6
imdb	10
quail	13
quartz	8
wiki_bio	5
Fourth Split (36 Tasks)	
adversarial_qa_dbidaf	5
adversarial_qa_droberta	5
duorc_SelfRC	9
duorc_ParaphraseRC	9
cos_e_v1.11	11
qasc	8
sciq	5
glue_qqp	6
social_i_qa	6
wiki_hop_original	9
wiqa	8
app_reviews	4
rotten_tomatoes	10
yelp_review_full	7
samsum	7
xsum	10
Meta-Test	
super_glue_wsc.fixed	
winogrande_winogrande_xl	
super_glue_cb	
super_glue_rte	
anli(r1/r2/r3)	
super_glue_copa	
hellaswag	
super_glue_wic	
story_cloze †	

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: The abstract and introduction accurately reflect the paper's contributions and scope.

   Guidelines:

   - The answer NA means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A No or NA answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: We can find the limitations in D.

   Guidelines:

   - The answer NA means that the paper has no limitation while the answer No means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory Assumptions and Proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

   Answer: [NA]

Justification: Our paper does not include theoretical results.

Guidelines:

- The answer NA means that the paper does not include theoretical results.
- All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
- All assumptions should be clearly stated or referenced in the statement of any theorems.
- The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
- Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
- Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental Result Reproducibility**

Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

Answer: [Yes]

Justification: We can reproduce the main experimental results following our settings in A and 4.

Guidelines:

- The answer NA means that the paper does not include experiments.
- If the paper includes experiments, a No answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
- If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
- Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
- While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
  (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
  (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
  (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
  (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

Answer: [Yes]

Justification: We'll open source the code to an anonymous site `https://anonymous.4open.science/r/TAGI` and put it on github after review.

Guidelines:

- The answer NA means that paper does not include experiments requiring code.
- Please see the NeurIPS code and data submission guidelines (`https://nips.cc/public/guides/CodeSubmissionPolicy`) for more details.
- While we encourage the release of code and data, we understand that this might not be possible, so "No" is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
- The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (`https://nips.cc/public/guides/CodeSubmissionPolicy`) for more details.
- The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
- The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
- At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
- Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental Setting/Details**

Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer, etc.) necessary to understand the results?

Answer: [Yes]

Justification: We can find the experimental settings (hyperparameters and datasets) in 4.3 and A.

Guidelines:

- The answer NA means that the paper does not include experiments.
- The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
- The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment Statistical Significance**

Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

Answer: [Yes]

Justification: We examined the effect of different hyperparameters on results in B.3.

Guidelines:

- The answer NA means that the paper does not include experiments.
- The authors should answer "Yes" if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
- The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
- The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
- The assumptions made should be given (e.g., Normally distributed errors).

- It should be clear whether the error bar is the standard deviation or the standard error of the mean.
- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g. negative error rates).
- If error bars are reported in tables or plots, The authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments Compute Resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

   Justification: We can find it in 4.3 and A.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code Of Ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics https://neurips.cc/public/EthicsGuidelines?

   Answer: [Yes]

   Justification: All of our studies follow the NeurIPS Code of Ethics.

   Guidelines:

   - The answer NA means that the authors have not reviewed the NeurIPS Code of Ethics.
   - If the authors answer No, they should explain the special circumstances that require a deviation from the Code of Ethics.
   - The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader Impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [NA]

    Justification: There is no societal impact of the work performed.

    Guidelines:

    - The answer NA means that there is no societal impact of the work performed.
    - If the authors answer NA or No, they should explain why their work has no societal impact or why the paper does not address societal impact.
    - Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.

- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pretrained language models, image generators, or scraped datasets)?

Answer: [NA]

Justification: The paper poses no such risks.

Guidelines:

- The answer NA means that the paper poses no such risks.
- Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
- Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
- We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

Answer: [Yes]

Justification: We follow their open-source protocols in all our uses.

Guidelines:

- The answer NA means that the paper does not use existing assets.
- The authors should cite the original paper that produced the code package or dataset.
- The authors should state which version of the asset is used and, if possible, include a URL.
- The name of the license (e.g., CC-BY 4.0) should be included for each asset.
- For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
- If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, paperswithcode.com/datasets has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
- For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.

- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New Assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

    Answer: [NA]

    Justification: This paper does not release new assets.

    Guidelines:

    - The answer NA means that the paper does not release new assets.
    - Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
    - The paper should discuss whether and how consent was obtained from people whose asset is used.
    - At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and Research with Human Subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

    Answer: [NA]

    Justification: This paper does not involve crowdsourcing nor research with human subjects.

    Guidelines:

    - The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
    - Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
    - According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional Review Board (IRB) Approvals or Equivalent for Research with Human Subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

    Answer: [NA]

    Justification: This paper does not involve crowdsourcing nor research with human subjects.

    Guidelines:

    - The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
    - Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
    - We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
    - For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.