PromptTrans: Eliciting In-Context Learning from Smaller Language Models by Translating Demonstrations to Prompts

Anonymous ACL submission

Abstract

Large language models (LMs) like GPT-3 have shown remarkable in-context learning ability: by concatenating demonstration examples as the input context, the model is able infer on an unseen task without further training. For smaller LMs like T5-large, however, in-context learning performance is abysmally poor. This poses a question on the design of the in-context learning framework - Is there a better way to condition on demonstrations, rather than simply concatenating them in text? Towards this, we propose PROMPTTRANS[1], a parameter-efficient tuning framework (3.4% of backbone LM parameters) to translate demonstrations to soft prompts and augment the input context with the translated soft prompts. We meta-train PROMPTTRANS on 120 tasks and evaluate on 40 unseen tasks from the CrossFit dataset, with few-shot ($\leq$ 16) demonstrations per task. Through our experiments, we show that PROMPTTRANS is indeed instrumental in eliciting in-context learning ability in smaller LMs (T5-large), without updating any parameter of the backbone. A particularly interesting finding is that across our extensive experiments PROMPTTRANS consistently outperforms baselines that meta-train the whole backbone LM for in-context learning and even large off-the-shelf LMs (with 16.8$\times$ parameters). Our promising results and analysis throw light on how even smaller LMs can learn in-context and alludes towards a more effective in-context learning paradigm.

1 Introduction

In recent years, we have witnessed a trend of scaling up language models (LMs) in size (Brown et al., 2020; Chowdhery et al., 2022; Shoeybi et al., 2019; Hoffmann et al., 2022). These large LMs demonstrate an emergent ability of in-context learning: they can perform a new task simply by conditioning on few demonstration examples in the input context (Wei et al., 2022b; Brown et al., 2020). For example, to predict the sentiment of "It's so dumb it's brilliant!", we only need to concatenate some demonstration examples to the original input like "I really enjoyed this movie! Positive. The plot was boring. Negative. It's so dumb it's brilliant!", and prompt the large LM with the concatenated prompt to get the expected prediction "Positive".

Such in-context learning ability can be game changing compared to the predominant finetuning paradigm (Devlin et al., 2019) with two key advantages (Dong et al., 2023). First, in-context learning eliminates the need to train the model on each downstream task, which is highly resource consuming for large LMs. Second, such ability makes LM-as-a-service possible (Sun et al., 2022), powering wide range of real world applications.

However, it appears that in-context learning ability is severely compromised as model size reduces: while GPT-3 175B model achieves over 60% accuracy on the synthesized task of removing a symbol from a word, GPT-3 1.3B model can only achieve around 5% accuracy (Brown et al., 2020). Such phenomenon compels the research question (Question A): "Are smaller LMs not capable of in-context learning? Why?"

To answer the above question, we first look into how in-context learning works for large LMs. Intuitively, in-context learning is about learning through analogies drawn from the given demonstration examples and applying that to infer on the query example (Dong et al., 2023). However, Lu et al. (2022) and Zhao et al. (2021) have empirically shown that LMs are sensitive to the order and format of demonstration context. In addition, Min et al. (2022c) show that randomizing the outputs in demonstration examples, which should break input-output mapping in each example, does not hinder in-context learning. These counter-intuitive evidences suggest that the way large LMs use demonstrations might be different from how humans would use them. These findings pose ques-

[1]We will make the source code public upon acceptance

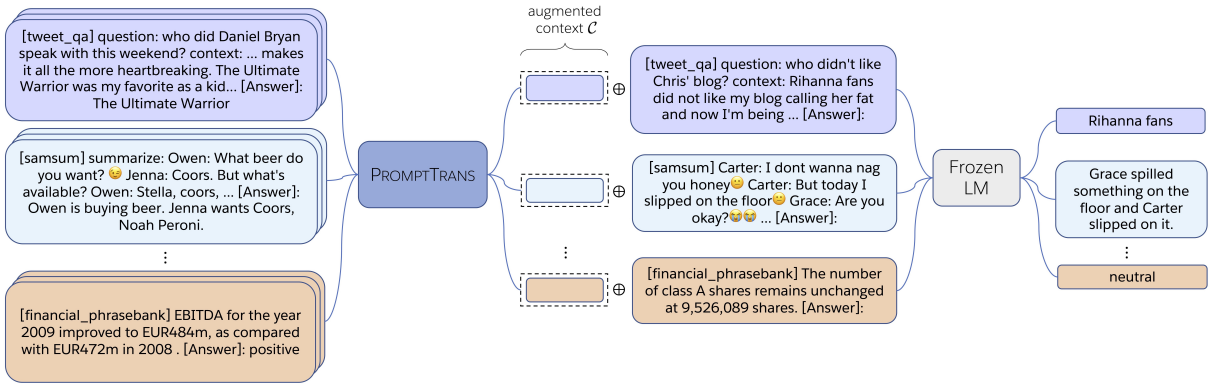

Figure 1: Overview of PROMPTTRANS framework. Step 1 (**left**): PROMPTTRANS receives task demonstrations as input and generates an augmented context $\mathcal{C}$ for each task. Step 2 (**right**): augmented context $\mathcal{C}$ concatenated with the actual query of each task is fed as input to the backbone LM (frozen) to produce the final prediction. PROMPTTRANS is parameter-efficient (26M, 3.4% of T5-large). The overall framework enjoys the same in-batch multi-task inference capability as soft-prompt tuning (Lester et al., 2021) or LM-as-a-service (Sun et al., 2022).

tions on whether concatenating the demonstration examples as the input context, though intuitive to humans, is the best representation for LMs to condition on them. Is there a better way to condition on these examples? (Question B)

To this end, in this work we propose a data driven framework, PROMPTTRANS (Figure 1), which (*i*) translates the demonstration examples to a fixed-length soft prompt – a sequence of *soft tokens* (Qin and Eisner, 2021) (*ii*) learns a global (not generated from demonstrations) soft prompt (Lester et al., 2021). PROMPTTRANS then combines the global prompt, the translated prompts and the original context to create an augmented context which is given as final input for the backbone LM to use.

To facilitate PROMPTTRANS's ability to translate demonstration examples to better meet the needs of the backbone LM without being exposed to test tasks, we meta-train it on a collection of up to 120 training tasks (with no overlap with the test tasks). During each iteration in meta-training, PROMPTTRANS receives a pair of demonstration set and a query input, and learns to adapt its translation mechanism to help the underlying LM make more accurate predictions when conditioned on the augmented context. Through such meta-training, we believe PROMPTTRANS (with only 3.4% parameters of the backbone LM) learns to effectively encode the kind of knowledge that is required by the backbone LM to learn in-context without requiring any parameter update.

We evaluate PROMPTTRANS on a collection of test tasks (up to 40), with few-shot ($\leq$ 16) demonstrations per task. From extensive experiments, we find that it can elicit in-context learning from smaller LMs like T5-large, without updating any parameter of the backbone LM (answer to Question A). With PROMPTTRANS, T5-large can consistently outperform baselines, including (*i*) MetaICL-Finetune (Min et al., 2022b), which finetunes the whole T5-large model during meta-training stage. (*ii*) INF-FT, which finetunes the whole T5-large model on demonstration examples at test time, (*iii*) MetaICL-PT, which tunes only a global soft prompt (Lester et al., 2021) during meta-training stage, and (*iv*) large off-the-shelf LMs such as OPT-13B (Zhang et al., 2022a) with 16.8$\times$ parameters.

Furthermore, we perform test analysis on PROMPTTRANS and find it is (*i*) robust to variation in the number of demonstrations provided during inference (a useful feature for in-context learning application in reality) and (*ii*) more sensitive to input-output mapping of demonstrations than (Min et al., 2022c) observed for Large LMs.

Since PROMPTTRANS only changes the input context without changing any parameters of the backbone LM or altering its forward propagation step like Adapters (Houlsby et al., 2019), its strong performance suggests that smaller LMs are infact inherently capable of learning in-context but the conventional context is not a good representation to elicit its in-context learning ability. As PROMPTTRANS verifies that a better representation of context is beneficial, we hope this can be the stepping stone for future research to study better context representations for in-context learning.

2 Related Work

In-context Learning Brown et al. (2020) show that large LMs can learn an unseen task, by con-

catenating few-shot demonstration examples in-context without any parameter update. Subsequent studies (Zhao et al., 2021; Lu et al., 2022; Rubin et al., 2022; Zhang et al., 2022b) further improve in-context learning by proposing techniques to find the best order and selection of demonstration examples. This work explores a direction orthogonal to example selection or ordering - that of improving the design of the in-context learning framework itself; by investigating into better representations of the input context. Unlike (Min et al., 2022a,b), where the model prediction is selected as the candidate option with the highest likelihood, we consider text-to-text generative prediction as it is more generalizable and does not require candidate options.

Meta-training for In-context Learning While LMs acquire in-context learning ability implicitly (Brown et al., 2020) through training on language modeling loss, Min et al. (2022b); Chen et al. (2022b) propose to explicitly finetune the model to learn in-context learning through a meta-training stage. During meta-training, the model learns to predict the output of a given query, by conditioning on the input context obtained by concatenating some demonstration data with the query input. Min et al. (2022b) show such meta-training can significantly boost in-context learning performance of GPT2-large. While PROMPTTRANS also goes through meta-training stage to learn its parameters, it does not change any parameter of the backbone LM. In this work, we show that PROMPTTRANS can significantly boost in-context learning ability of smaller LMs like T5-large to the extent that they even outperform meta-trained counterparts that finetune the backbone LM.

Meta-training for Instruction Tuning More recently, there has been a trend of explicitly finetuning colossal-sized LMs on large instruction-tuning corpus (Sanh et al., 2022; Wei et al., 2022a; Wang et al., 2022; Ouyang et al., 2022). Some task instructions contain demonstration examples (Wang et al., 2022) while others only include general task descriptions (Sanh et al., 2022; Wei et al., 2022a). To human's intuition, task descriptions reveal a different aspect of the task knowledge compared to demonstrations. While it is possible to extend PROMPTTRANS to include task descriptions, in this pioneer study we stay focused on learning better representation for demonstrations.

Prompt Tuning and Generation Lester et al. (2021) propose soft prompt tuning (PT), which only tunes a sequence of soft tokens appended to the input and keeps the backbone LM frozen. Levine et al. (2022); Zhang et al. (2022c) propose to generate an input-dependent soft prompt instead of learning a global task-level soft prompt. These studies generate prompts based on the actual query input and can be seen as advanced PT methods. Although the translation model in PROMPTTRANS has similar architecture as (Levine et al., 2022), our work is distinct in that we condition not on the actual query input but on demonstration examples. Thus, PROMPTTRANS learns to generate better in-context learning representation for the backbone LM, rather than to solve specific tasks better.

3 Method

3.1 In-context Learning

The general setting of in-context learning on a given task assumes, that for every evaluation instance or query x^q, some (*e.g.,* n for n-shot) annotated data $\mathcal{D}^s = \{d_i^s = (x_i^s, y_i^s)\}_{i=1}^n$ are provided as demonstrations (*a.k.a.* support set) . The task of in-context learning is to predict y^q, the output, conditioned on $\mathcal{D}^s$ and x^q. Formally, $\hat{y}^q = \arg\max_y P_\phi(y|x^q, \mathcal{D}^s)$ where ϕ is the backbone model. $\mathcal{D}^s$ is usually few-shot, with $n \leq 16$. The conventional way to condition on x^q and $\mathcal{D}^s$ is to first concatenate the examples in $\mathcal{D}^s$ to form the input context $\mathcal{C}$ as: $\mathcal{C} \leftarrow d_1^s \oplus \ldots \oplus d_n^s$, then concatenate $\mathcal{C}$ with x^q: $\mathcal{C} \oplus x^q$, as the final input to the model (Min et al., 2022b).

3.2 PROMPTTRANS Framework

Is the input context $\mathcal{C}$ defined in §3.1 the best context representation for the backbone model to make use of the support set? In PROMPTTRANS framework, we seek a better context representation than simply concatenating examples in plain text.

Figure 1 illustrates the overall framework. Given the few-shot demonstrations of a new task, PROMPTTRANS converts them to an augmented context $\mathcal{C}$, which is then concatenated with actual query input of that task and fed as input to the backbone LM for prediction. PROMPTTRANS is a parameter efficient framework, with only 3.4% parameters of the backbone LM (T5-large) and thus can be easily deployed to multiple accelerators (GPUs, TPUs) for multi-task inference. Since the backbone LM is kept frozen, the overall framework enjoys the

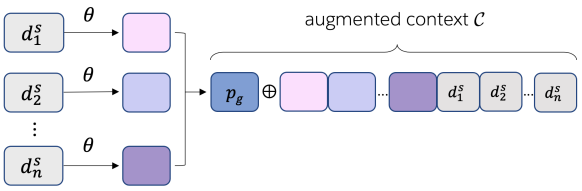

Figure 2: Generating augmented context by PROMPTTRANS. θ denotes the translation model, d_i^s denotes i-th support example and $\boldsymbol{p}_g$ denotes a learnable soft prompt.

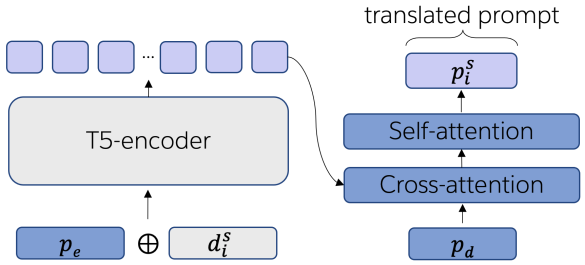

Figure 3: Translation model architecture. $\boldsymbol{p}_e$: soft prompt to encoder, $\boldsymbol{p}_d$: soft prompt to decoder (cross-attention layer + self-attention layer). All tunable parameters are colored in blue.

same in-batch multi-task inference capability as soft prompt tuning (Lester et al., 2021) and LM-as-a-service (Sun et al., 2022).

Translation Pipeline As illustrated in Figure 2, we use a translation model θ to convert the support examples to soft prompts that can be better understood by the backbone LM ϕ and also guide it better to solve a new task. We then augment the input context $\mathcal{C}$ with the translated prompt.

Formally, the translation model first converts each support example $d_i^s = x_i^s \oplus y_i^s$ to a soft prompt p_i^s. Then the translated soft prompts $P^s = \{p_i^s\}_{i=1}^n$ are concatenated with the support examples $\{d_i^s\}_{i=1}^n$. Additionally, a learnable soft prompt $\boldsymbol{p}_g$ is left-appended to it to construct the final input context $\mathcal{C}$: $\mathcal{C} \leftarrow \boldsymbol{p}_g \oplus p_1^s \oplus \ldots \oplus p_n^s \oplus d_1^s \oplus \ldots \oplus d_n^s$. Unlike p_i^s which is dependent on the support data, $\boldsymbol{p}_g$ is a global soft prompt similar to (Lester et al., 2021) and is generated from the support set. $\boldsymbol{p}_g$ gives a general context for in-context learning while p_i^s help the backbone LM better understand each support example.

Translation Model Architecture The translation model essentially converts the support set $\mathcal{D}^s = \{d_i^s\}_{i=1}^n$ into a set of soft tokens $P^s = \{p_i^s\}_{i=1}^n$. One way to model this would be to adopt a seq2seq architecture like the Transformer (Vaswani et al., 2017) to generate P^s autoregressively from $\mathcal{D}^s$. However, that would assume the demonstration samples (and the soft tokens) to be sequentially dependent. In reality, the demonstration samples are independent and the probability of them appearing together is low in natural text (Xie et al., 2022). Therefore, instead of using an auto-regressive model which is also time-consuming (non-parallelizable), we follow (Jaegle et al., 2022; Levine et al., 2022) to construct a non-autoregressive model which independently accepts the demonstration examples and generates the corresponding soft tokens.

Figure 3 illustrates the architecture of the translation model. Given an input d_i^s of variable length, we concatenate it with a learnable soft prompt $\boldsymbol{p}_e$ and pass it into a frozen T5 encoder. The variable length output of the T5 encoder is then passed through a simple 1-step decoder model to get a fixed-length output vector. The decoder takes a learnable soft prompt $\boldsymbol{p}_d \in \mathbb{R}^k$ as input and first attends it on T5 encoded features through a cross attention layer to get a contextualised representation. This is further passed through another self-attention layer to get the final soft token $p_i^s \in \mathbb{R}^k$. Since each p_i^s is only dependent on its corresponding d_i^s, the process of translating $d_i^s \rightarrow p_i^s$ for $i = 1 \ldots n$ can be parallelized.

3.3 Meta Training and Inference

We meta train PROMPTTRANS with a collection of tasks $\mathbb{T}_{\text{train}}$. For every training iteration, a task $\mathcal{T}_{tr}$ is first sampled from $\mathbb{T}_{\text{train}}$. Then the support data $\mathcal{D}_{tr}^s$ and one query $d_{tr}^q = (x_{tr}^q, y_{tr}^q)$ are sampled from $\mathcal{T}_{tr}$. We use PROMPTTRANS to convert $\mathcal{D}_{tr}^s$ to input context $\mathcal{C}$ and feed $\mathcal{C}$ with query input x_{tr}^q to predict the output y_{tr}^q. We compute cross entropy on the final prediction as the training loss:

$$\mathcal{L}^{\text{meta}}(\theta') = \mathbb{E}_{(\mathcal{D}_{tr}^s, d_{tr}^q)\sim\mathcal{T}_{tr}}\big[-\log P_\phi(y_{tr}^q|\,\mathcal{C}\oplus x_{tr}^q)\big] \quad (1)$$

We update the PROMPTTRANS parameters $\theta' = \{\theta, \boldsymbol{p}_g\}$ while keeping the backbone model ϕ fixed.

We evaluate PROMPTTRANS with a collection of unseen tasks $\mathbb{T}_{\text{test}}$. Each target task $\mathcal{T} \in \mathbb{T}_{\text{test}}$ consists of multiple support and query data pairs. For each pair $(\mathcal{D}^s, d^q)$, we provide $\mathcal{D}^s$ as input to the PROMPTTRANS module to get the output $\mathcal{C}$, which is then concatenated to x^q (i.e. $\mathcal{C} \oplus x^q$) and passed to the backbone model to get the final prediction: $\hat{y}^q \leftarrow \arg\max_y P_\phi(y|\,\mathcal{C}\oplus x^q)$

4 Experimental Setup

4.1 Datasets

We use the CROSSFIT (Ye et al., 2021) dataset for meta training and evaluation. Primarily, we experiment with three partitions: *Random*, *cls-to-cls* and *half-to-cls*. As shown in Table 1, each partition contains a collection of meta-training and evaluation tasks. Each task has its own training set $\mathcal{D}^{\text{train}}$ and development set $\mathcal{D}^{\text{dev}}$. There are 80 examples *per class* in $\mathcal{D}^{\text{train}}$ or $\mathcal{D}^{\text{dev}}$ for classification and regression tasks, and 160 examples for other tasks[2]. During meta training, support data $\mathcal{D}^{s}_{tr}$ is sampled from $\mathcal{D}^{\text{train}}_{tr}$ and query data d^{q}_{tr} is sampled from $\mathcal{D}^{\text{dev}}_{tr}$. We construct the evaluation data by sampling 50 pairs of $(\mathcal{D}^{s}, d^{q})$ for each evaluation task $\mathcal{T} \in \mathbb{T}_{\text{test}}$ on *Random* partition and 100 pairs for each task in other two partitions, resulting in 2k in-context learning test instances for each partition.

4.2 Baselines

We use T5-large (Raffel et al., 2020) as our backbone LM across all experiments. We consider the following baselines to compare with:

MetaICL-Finetune finetunes the backbone LM during the meta-training stage, following Min et al. (2022b).

MetaICL-PT does soft prompt tuning (Lester et al., 2021) during meta-training stage. It appends a global soft to input context and tunes only on that soft prompt while keeping the backbone LM frozen.

INF-Finetune optimizes the LM during testing with $\mathcal{D}^{s}$, without involving any meta-training. This falls back to conventional finetuning on annotated data $\mathcal{D}^{s}$ instead of in-context learning. Specifically, given $(\mathcal{D}^{s}, d^{q})$ pair during inference, we finetune the backbone LM for 100 steps on $\mathcal{D}^{s}$ and use the finetuned model to infer on d^{q}. This process is repeated for each test data point.

Off-the-shelf LM uses the off-the-shelf backbone LM for inference on in-context learning data. For this baseline, we report on different LM backbones including T5-{large, xxl} (Raffel et al., 2020) and OPT-{13B,30B,66B} (Zhang et al., 2022a).

To summarise, amongst all models, only MetaICL and PROMPTTRANS perform meta-training. The demonstration context is appended to input for all models other than INF-FT which instead uses it to finetune the LM parameters during inference.

4.3 Evaluation Metrics

There are 7 task-specific evaluation metrics in CROSSFIT: Classification-F1, Accuracy, QA-F1, Exact Match(EM), Rouge-L, Mattew correlation and Pearson correlation. To evaluate a model on a collection of tasks, we use the following metrics:

- Average of task-specific performances (Avg)
- Weighted Geometric Mean of Relative Gain per task group (wGMRG)

For wGMRG, we construct task groups out of tasks having the same evaluation metric. Computing average relative gain per task group allows a more stable evaluation than average relative gain per specific task.

Suppose the relative performance ratio per task group is: $r = (r_1, \ldots, r_n)$, the size ratio for each task group is: $w = (w_1, \ldots, w_n)$. Then wGMRG is computed by $(\prod_{i=1}^{n} r_i^{w_i}) - 1$. For example, say Model1 performs $1.2\times$ better than Model2 in QA-F1 category and achieves only $0.9\times$ of Model2's scores in Rouge-L tasks, then the then supposing there are 2 QA-F1 tasks and 1 Rouge-L task, then the wGMRG is $(1.2^{2/3} \times 0.9^{1/3}) - 1 \approx +9\%$

4.4 Implementation Details

We use a learning rate of $1e$-5 for MetaICL-PT baseline following (Lester et al., 2021) and $5e$-5 for MetaICL-Finetune and INF-Finetune baselines following (Chen et al., 2022a). For PROMPTTRANS, we set $1e$-5 learning rate for tunable soft prompts $\{\boldsymbol{p}_e, \boldsymbol{p}_d, \boldsymbol{p}_g\}$ and $5e$-5 for parameters in 1-layer decoder. All experiments use Adam optimizer (Kingma and Ba, 2015) with batch size 8, weight decay of $1e$-5 and 150 warm-up steps. For each partition, we train 400 epochs for 16-shot support setting and 200 epochs for 8-shot support. For each epoch, we sample a single data point (i.e. support set and a query) from each training task. The numbers of soft tokens for $\{\boldsymbol{p}_e, \boldsymbol{p}_d, \boldsymbol{p}_g\}$ are set to {100,30,100}. The number of soft tokens for MetaICL-PT is 200[3]. To ensure a fair comparison with comparable inference costs, we set the max context length of concatenated support examples to 1024 tokens for MetaICL-{Finetune,PT} and 512 tokens for PROMPTTRANS. The max output length is set to 64 for all models. All experiments are carried

[2] we combine 5 seeds data splits in CROSSFIT

[3] More tokens do not improve performance

Methods	#params		Inf-Time	Random		cls-to-cls		half-to-cls	
				120 Train → 40 Test tasks		45 cls. → 20 cls. tasks		23 cls. + 22 non-cls. → 20 cls.	
	backbone	tunable	(minutes)	8-shot	16-shot	8-shot	16-shot	8-shot	16-shot
MetaICL-Finetune	770M	770M	8.2	33.5(-14.1%)	37.0(- 9.7%)	38.4(-20.2%)	38.3(-19.6%)	39.2(-12.0%)	40.7(-13.8%)
MetaICL-PT	770M	0.13M	5.7	28.2(-27.7%)	32.3(-18.8%)	27.1(-44.6%)	25.6(-47.2%)	29.2(-35.1%)	39.9(-15.2%)
INF-FT	770M	770M	263.8	34.9(- 7.2%)	39.2(- 0.1%)	31.7(-34.8%)	38.4(-18.8%)	31.7(-29.4%)	38.4(-18.4%)
OPT-66B	66B	-	312.5	**39.8**(+ **9.3**%)	**41.0**(+ 7.3%)	46.3(- 3.7%)	47.0(- 0.7%)	**46.3**(+ **4.1**%)	47.0(- 0.2%)
OPT-30B	30B	-	189.6	38.0(+ 4.5%)	39.2(+ 2.8%)	44.3(- 7.8%)	44.3(- 6.4%)	44.3(- 0.3%)	44.3(- 5.9%)
OPT-13B	13B	-	151.4	36.1(- 0.8%)	36.8(- 3.3%)	42.0(-12.8%)	41.5(-12.4%)	42.0(- 5.7%)	41.5(-12.0%)
T5-xxl	11B	-	206.4	1.0(-99.0%)	3.0(-99.8%)	0.4(-99.6%)	0.0(-100.0%)	0.4(-99.6%)	0.0(-100.0%)
T5-large	770M	-	22.1	1.8(-99.9%)	1.8(-99.9%)	0.0(-100.0%)	0.0(-100.0%)	0.0(-100.0%)	0.0(-100.0%)
PROMPTTRANS	770M	26M	8.0	37.7(+ 0.0%)	40.0(+ 0.0%)	**48.2**(+ **0.0**%)	**47.6**(+ **0.0**%)	44.5(+ 0.0%)	**47.1**(+ **0.0**%)

Table 1: Performance on unseen test data for various data partition. Each cell shows *points(percentage)*, where *points* is Avg and *percentage* is wGMRG compared to PROMPTTRANS. #params shows the parameter size of the backbone LM (backbone) and the size of trainable parameters (tunable). Inf-Time denotes the average time required to inference on 2k test data. **Best** and 1st runner-up results are highlighted.

out on eight 40GB A100 GPUs and 400 epochs of training with PROMPTTRANS on *Random* partition takes around 3.3 hours.

5 Experimental Results

5.1 Main Results

Table 1 shows the main results of PROMPTTRANS compared with baselines, under 3 partitions. Overall, we observe similar trend over compared models across all settings. (see Appendix B for more detailed results)

T5-large off-the-shelf cannot learn in-context T5-large and T5-xxl off-the-shelf baselines (row 7-8) shows nearly zero performance on all three partitions, which indicates that T5 is not amenable to in-context learning. However, we observe that PROMPTTRANS, even without changing T5-large parameters, is able to elicit in-context learning and achieve either best results (in *cls-to-cls* / *half-to-cls* partition) or close to best results (in *Random* partition), out of all the baselines. Such empirical results strongly suggest that LMs considered not suitable for in-context learning (like T5-large) are inherently capable of it. The key factor to elicit its in-context ability is to use PROMPTTRANS to construct better context representation.

PROMPTTRANS consistently outperforms MetaICL methods The first row block shows our MetaICL baselines. MetaICL-Finetune demonstrates effective in-context learning ability, inline with Min et al. (2022b). Although MetaICL-Finetune trains the whole T5-large (770M parameters), PROMPTTRANS consistently outperforms it in all settings while only tuning

Methods	PROMPTTRANS			*Random*	
	$\lvert p_g \rvert$	$\lvert p_d \rvert$	$\lvert p_e \rvert$	8-shot	16-shot
PROMPTTRANS	100	20	100	37.7(+ 0.0%)	40.0(+ 0.0%)
MetaICL-Finetune	\	\	\	33.5(-14.1%)	37.0(- 9.7%)
PROMPTTRANS	50	=	=	37.1(- 1.4%)	39.0(- 2.0%)
PROMPTTRANS	0	=	=	29.5(-34.7%)	32.7(-17.5%)
PROMPTTRANS	=	10	=	38.1(+ 0.3%)	40.1(- 0.3%)
PROMPTTRANS	=	30	=	36.5(- 4.4%)	36.9(- 7.8%)
PROMPTTRANS	=	=	50	38.1(- 0.6%)	38.6(- 3.2%)
PROMPTTRANS	=	=	0	35.9(- 4.9%)	37.5(- 5.5%)

Table 2: Ablation on the number of tokens for $\boldsymbol{p}_g$, $\boldsymbol{p}_d$ and $\boldsymbol{p}_e$. "=" indicates same as default setting (first row)

3.4% of T5-large parameters and keeping the backbone LM fixed. This shows that finetuning the LM parameters for explicit meta-training is neither necessary nor optimal. On the other hand, MetaICL-PT keeps the backbone LM frozen like PROMPTTRANS, but its performance is significantly compromised: 7-22 points below PROMPTTRANS across 3 partitions. Such performance gap suggests that translating the demonstration examples to soft prompts is indeed crucial to elicit strong in-context learning of backbone LMs.

PROMPTTRANS is better than INF-FT Finetuning the backbone LM used to be the default paradigm to adapt LMs to downstream tasks. Here we observe that PROMPTTRANS consistently outperforms INF-FT, with up to 16 points advantage. In other words, with PROMPTTRANS, T5-large is better at in-context learning on few-shot data than finetuning itself.

PROMPTTRANS is not far behind large LMs We observe strong in-context learning performance from large LMs like OPT-{66B,30B,13B}, inline

with (Brown et al., 2020; Chowdhery et al., 2022; Wei et al., 2022b). Remarkably, PROMPTTRANS on T5-large is able to outperform OPT-66B model in *cls-to-cls*, surpass OPT-30B in *half-to-cls* and achieve performance comparable to OPT-30B in *Random*. Contrary to Wei et al. (2022b)'s finding that in-context learning ability can only emerge with large LMs (typically $\geq$ 10B), we observe that with better context representation from PROMPTTRANS, small LMs like T5-large can consistently outperform OPT-13B having 16.8$\times$ parameters.

Inference is fast for PROMPTTRANS Finally, we investigate if PROMPTTRANS can perform fast inference, a crucial property for in-context learning. We record the average time to predict on 2k test data across 6 settings (3 partitions $\times$ {8-shot,16-shot}), on eight 40GB GPUs with batch size 1 per GPU. Firstly, we observe that PROMPTTRANS has one of the lowest inference time of 8 minutes, which is roughly 0.24 seconds per test query. On the contrary, INF-FT is 32$\times$ slower than PROMPTTRANS, as it needs to train the LM for 100 steps on each demonstration $\mathcal{D}^s$ data during inference. Large LMs (OPT-{66B,30B,13B}) are also time consuming (19$\times$-39$\times$ slower than PROMPTTRANS), as the model is too large to be loaded into a single GPU thus we have to split the model over multiple GPUs using Accelerate library (Sylvain Gugger, 2022). Besides, T5-large is slower than PROMPTTRANS as it is not amenable to in-context learning and hence typically generate longer outputs.

5.2 Ablation on PROMPTTRANS

Number of prompt tokens To understand the role of each learnable soft prompt in PROMPTTRANS, we conduct an ablation on the size of soft prompts $\boldsymbol{p}_g$, $\boldsymbol{p}_d$ and $\boldsymbol{p}_e$. From Table 2, we can see while reducing the length of $\boldsymbol{p}_g$ to 50 slightly degrades performance, entirely removing it severely hurts the effectiveness of PROMPTTRANS. This shows that $\boldsymbol{p}_g$ is a necessary component and it is important to provide a global context to the backbone LM. Next, we observe that longer length for $\boldsymbol{p}_d$ actually decreases performance while shortening its length keeps the performance comparable. This shows that 10-20 tokens might be a sweet spot for $\boldsymbol{p}_d$. In other words, it is optimal to translate each demonstration example to 10-20 soft tokens. Note that reducing the length of $\boldsymbol{p}_d$ to 0 token will reduce PROMPTTRANS to MetaICL-PT. Finally, the size of $\boldsymbol{p}_e$ affects the representation capacity of the translation model and reducing its length naturally comprises the overall performance.

Methods	*Random*	
	8-shot	16-shot
PROMPTTRANS	37.7(+ 0.0%)	40.0(+ 0.0%)
MetaICL-Finetune	33.5(-14.1%)	37.0(- 9.7%)
PROMPTTRANS-replace-input	37.2(- 5.7%)	38.5(- 7.7%)
PROMPTTRANS-replace-all	20.0(-50.4%)	13.8(-72.9%)

Table 3: Ablation on removing text context.

Remove original context The PROMPTTRANS pipeline in Figure 2 primarily augment original text context with translated soft prompts. As soft prompts show strong results in eliciting in-context learning ability, we wonder if they can replace the text context altogether. In Table 3, we compare PROMPTTRANS with two variations: (*i*) *replace-input* variation will remove only the inputs of demonstration examples and concatenate translated soft prompts and demonstration outputs in an interleaved manner: $\mathcal{C} \leftarrow \boldsymbol{p}_g \oplus p_1^s \oplus y_1^s \ldots p_n^s \oplus y_n^s$. (*ii*) *replace-all* variation will remove all demonstration context and only use translated prompts: $\mathcal{C} \leftarrow \boldsymbol{p}_g \oplus p_1^s \ldots p_n^s$. We find that while *replace-input* variation hurts PROMPTTRANS performance to some extent, it still outperforms MetaICL-Finetune. Whereas, the *replace-all* variation leads to a drastic performance degradation. Such results suggest a different role of the demonstration input and output data during in-context learning. While the inputs can be replaced by translated soft prompts, the demonstration outputs are more critical and should be explicitly retained in context. This observation provides an insight on how in-context learning works for LMs, inline with Min et al. (2022c).

5.3 Further Analysis on PROMPTTRANS

Performance Break-down per Task Group Figure 4 shows the weighted relative gain per task group ($r_i^{w_i} - 1$) of baselines compared to PROMPTTRANS. We observe that 1) MetaICL-{Finetune,PT} models perform worser than PROMPTTRANS in 4 out of 5 task groups. 2) large OPT models fall behind PROMPTTRANS on natural language understanding tasks but are better at generative tasks (Rouge-L, Exact-Match). We conjecture this is because larger LMs trained with causal LM objective on larger-scale data are better at text generation than smaller LMs like T5-large, which was primarily trained with masked span prediction (Raffel et al., 2020).

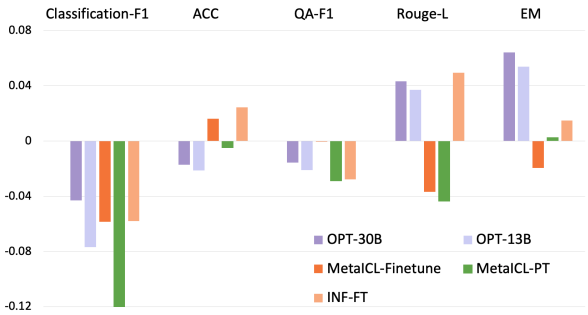

Figure 4: Weighted relative gain per task group w.r.t PROMPTTRANS, under *Random* partition 16-shot.

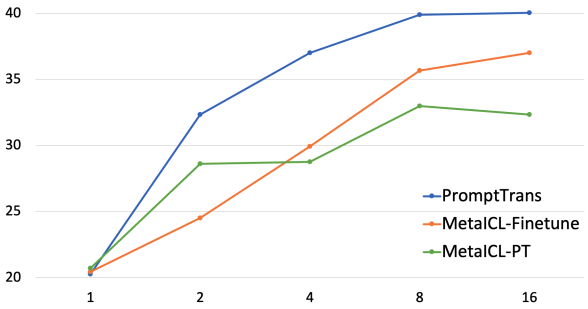

Figure 5: Ablation on different support size at inference. Models are trained on 16-shot under *Random* partition but tested with different number of support examples.

Different Support Size at Inference In realistic scenarios, there might be different number of demonstration examples available, especially during inference on an unseen test task. To understand how PROMPTTRANS adapts to lesser number of demonstrations at inference, after being trained on 16-shot demonstrations, figure 5 shows the performance with regard to different support sizes. We observe that PROMPTTRANS is indeed more robust to variable number of demonstrations, compared to MetaICL-Finetune and MetaICL-PT.

Intervention on Demonstrations As Min et al. (2022c) point out, randomizing the labels of demonstration examples "barely hurts performance". To verify if PROMPTTRANS shows similar idiosyncrasies, we perform an intervention on demonstrations which permutes the outputs of examples among themselves. This will break the input-output mapping while ensuring that the output space stays intact. Table 4 shows the empirical results, where we can observe that while for 8-shot MetaICL-Finetune is not much affected by such intervention, PROMPTTRANS is more sensitive to it and drops 15.3% in performance. In 16-shot case, both models are significantly hurt by such intervention.

Methods	Permute Output	*Random* 8-shot	*Random* 16-shot
PROMPTTRANS	No	37.7(+ 0.0%)	40.0(+ 0.0%)
	Yes	32.9(-15.3%)	35.2(-15.5%)
MetaICL-Finetune	No	33.5(-14.1%)	37.0(- 9.7%)
	Yes	33.1(-16.6%)	32.7(-22.6%)

Table 4: Test Analysis by permuting demonstration outputs

Comparison to Instruction-tuned Models We additionally compare PROMPTTRANS with two instruction-tuned models T0++ (11B) (Sanh et al., 2022) and FLAN-T5-xxl (11B) (Chung et al., 2022). We use a reduced test set of 31 tasks in *Random* partition, after removing the overlapped tasks that are used in T0's training[4]. The results show FLAN-T5-xxl is 3.1 points (or 2.8% in wGMRG) behind PROMPTTRANS and T0++ is 20.7 points (68.7% in wGMRG) below PROMPTTRANS. We believe FLAN-T5-xxl is much better than T0++ because it is trained with instructions containing demonstrations while T0++ is trained on pure task descriptions. Nevertheless, PROMPTTRANS outperforms FLAN-T5-xxl which is trained on much larger corpus (1.8K tasks) with 14× parameters. Given its effectiveness of eliciting in-context learning on demonstrations, it is appealing to extend PROMPTTRANS to elicit LMs to learn instructions(with or without demonstrations) in-context as a future work.

6 Conclusion

In this work, we investigate into better representations of the input context to make in-context learning more effective especially for smaller LMs. Towards this end, we propose PROMPTTRANS, a parameter-efficient framework to translate demonstrations to soft prompts and augment the input context with the translated prompts. Our extensive experiments show that without changing the backbone LM parameters, PROMPTTRANS is able to elicit strong in-context learning ability in smaller LMs like T5-large and outperform even large off-the-shelf LMs and models that finetune the backbone LM. As a pioneer study, we hope this work spurs more research towards learning better context representation for in-context learning.

[4]FLAN-T5-xxl training tasks contain T0 training tasks

Limitations

In this section, we discuss the limitation of PROMPTTRANS and potential future directions to make it more valuable.

On Backbone LM In this work, we use T5-large as the backbone LM for PROMPTTRANS. As we show PROMPTTRANS can improve T5-large from nearly zero in-context learning performance to a level outperforming OPT-13B ($16.8\times$ parameters), it is thus appealing to extend PROMPTTRANS to large LMs which are already good at in-context learning. It will be interesting to see how much PROMPTTRANS can further improve on top of those large LMs. In other words, how much potential of in-context learning ability is unexplored in large LMs due to conventional non-optimal representation of context (by concatenating demonstration examples).

On Scalability of In-context Learning In-context learning has the issue of long input as all demonstrations are concatenated to the context. PROMPTTRANS does not solve this problem as it need to augment the context (although we truncate the text to ensure total context length is comparable §4.4). However, it would be appealing to study how PROMPTTRANS can make the context representation more compact and thus accommodate more demonstration samples. One possible solution is to replace lengthy demonstrations with compact translated prompts. However, as §5.2 shows that simply removing text context hurts performance, more advanced techniques such as adding reconstruction loss for translated prompts or curriculum learning to gradually replace more demonstrations with prompts could be explored. Alternatively, another promising direction can be to combine PROMPTTRANS with grouped context encoding (Hao et al., 2022) i.e. improving representation of each demonstration with PROMPTTRANS while using grouped context encoding to accommodate up to 1k demonstrations.

References

Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared 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 M. Ziegler, Jeffrey Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. 2020. Language models are few-shot learners. In *NeurIPS*.

Hailin Chen, Amrita Saha, Shafiq R. Joty, and Steven C. H. Hoi. 2022a. Learning label modular prompts for text classification in the wild. *CoRR*, abs/2211.17142.

Yanda Chen, Ruiqi Zhong, Sheng Zha, George Karypis, and He He. 2022b. Meta-learning via language model in-context tuning. In *ACL (1)*, pages 719–730. Association for Computational Linguistics.

Aakanksha Chowdhery, Sharan Narang, Jacob Devlin, Maarten Bosma, Gaurav Mishra, Adam Roberts, Paul Barham, Hyung Won Chung, Charles Sutton, Sebastian Gehrmann, Parker Schuh, Kensen Shi, Sasha Tsvyashchenko, Joshua Maynez, Abhishek Rao, Parker Barnes, Yi Tay, Noam Shazeer, Vinodkumar Prabhakaran, Emily Reif, Nan Du, Ben Hutchinson, Reiner Pope, James Bradbury, Jacob Austin, Michael Isard, Guy Gur-Ari, Pengcheng Yin, Toju Duke, Anselm Levskaya, Sanjay Ghemawat, Sunipa Dev, Henryk Michalewski, Xavier Garcia, Vedant Misra, Kevin Robinson, Liam Fedus, Denny Zhou, Daphne Ippolito, David Luan, Hyeontaek Lim, Barret Zoph, Alexander Spiridonov, Ryan Sepassi, David Dohan, Shivani Agrawal, Mark Omernick, Andrew M. Dai, Thanumalayan Sankaranarayana Pillai, Marie Pellat, Aitor Lewkowycz, Erica Moreira, Rewon Child, Oleksandr Polozov, Katherine Lee, Zongwei Zhou, Xuezhi Wang, Brennan Saeta, Mark Diaz, Orhan Firat, Michele Catasta, Jason Wei, Kathy Meier-Hellstern, Douglas Eck, Jeff Dean, Slav Petrov, and Noah Fiedel. 2022. Palm: Scaling language modeling with pathways. *CoRR*, abs/2204.02311.

Hyung Won Chung, Le Hou, Shayne Longpre, Barret Zoph, Yi Tay, William Fedus, Eric Li, Xuezhi Wang, Mostafa Dehghani, Siddhartha Brahma, Albert Webson, Shixiang Shane Gu, Zhuyun Dai, Mirac Suzgun, Xinyun Chen, Aakanksha Chowdhery, Sharan Narang, Gaurav Mishra, Adams Yu, Vincent Y. Zhao, Yanping Huang, Andrew M. Dai, Hongkun Yu, Slav Petrov, Ed H. Chi, Jeff Dean, Jacob Devlin, Adam Roberts, Denny Zhou, Quoc V. Le, and Jason Wei. 2022. Scaling instruction-finetuned language models. *CoRR*, abs/2210.11416.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: pre-training of deep bidirectional transformers for language understanding. In *NAACL-HLT (1)*, pages 4171–4186. Association for Computational Linguistics.

Qingxiu Dong, Lei Li, Damai Dai, Ce Zheng, Zhiyong Wu, Baobao Chang, Xu Sun, Jingjing Xu, Lei Li, and Zhifang Sui. 2023. A survey for in-context learning. *CoRR*, abs/2301.00234.

Yaru Hao, Yutao Sun, Li Dong, Zhixiong Han, Yuxian Gu, and Furu Wei. 2022. Structured prompting: Scaling in-context learning to 1, 000 examples. *CoRR*, abs/2212.06713.

Jordan Hoffmann, Sebastian Borgeaud, Arthur Mensch, Elena Buchatskaya, Trevor Cai, Eliza Rutherford, Diego de Las Casas, Lisa Anne Hendricks, Johannes Welbl, Aidan Clark, Tom Hennigan, Eric Noland, Katie Millican, George van den Driessche, Bogdan Damoc, Aurelia Guy, Simon Osindero, Karen Simonyan, Erich Elsen, Jack W. Rae, Oriol Vinyals, and Laurent Sifre. 2022. Training compute-optimal large language models. *CoRR*, abs/2203.15556.

Neil Houlsby, Andrei Giurgiu, Stanislaw Jastrzebski, Bruna Morrone, Quentin De Laroussilhe, Andrea Gesmundo, Mona Attariyan, and Sylvain Gelly. 2019. Parameter-efficient transfer learning for nlp. In *International Conference on Machine Learning*, pages 2790–2799. PMLR.

Andrew Jaegle, Sebastian Borgeaud, Jean-Baptiste Alayrac, Carl Doersch, Catalin Ionescu, David Ding, Skanda Koppula, Daniel Zoran, Andrew Brock, Evan Shelhamer, Olivier J. Hénaff, Matthew M. Botvinick, Andrew Zisserman, Oriol Vinyals, and João Carreira. 2022. Perceiver IO: A general architecture for structured inputs & outputs. In *ICLR*. OpenReview.net.

Diederik P. Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. In *ICLR (Poster)*.

Brian Lester, Rami Al-Rfou, and Noah Constant. 2021. The power of scale for parameter-efficient prompt tuning. In *EMNLP (1)*, pages 3045–3059. Association for Computational Linguistics.

Yoav Levine, Itay Dalmedigos, Ori Ram, Yoel Zeldes, Daniel Jannai, Dor Muhlgay, Yoni Osin, Opher Lieber, Barak Lenz, Shai Shalev-Shwartz, Amnon Shashua, Kevin Leyton-Brown, and Yoav Shoham. 2022. Standing on the shoulders of giant frozen language models. *CoRR*, abs/2204.10019.

Yao Lu, Max Bartolo, Alastair Moore, Sebastian Riedel, and Pontus Stenetorp. 2022. Fantastically ordered prompts and where to find them: Overcoming few-shot prompt order sensitivity. In *ACL (1)*, pages 8086–8098. Association for Computational Linguistics.

Sewon Min, Mike Lewis, Hannaneh Hajishirzi, and Luke Zettlemoyer. 2022a. Noisy channel language model prompting for few-shot text classification. In *ACL (1)*, pages 5316–5330. Association for Computational Linguistics.

Sewon Min, Mike Lewis, Luke Zettlemoyer, and Hannaneh Hajishirzi. 2022b. Metaicl: Learning to learn in context. In *NAACL-HLT*, pages 2791–2809. Association for Computational Linguistics.

Sewon Min, Xinxi Lyu, Ari Holtzman, Mikel Artetxe, Mike Lewis, Hannaneh Hajishirzi, and Luke Zettlemoyer. 2022c. Rethinking the role of demonstrations: What makes in-context learning work? *CoRR*, abs/2202.12837.

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 F. Christiano, Jan Leike, and Ryan Lowe. 2022. Training language models to follow instructions with human feedback. *CoRR*, abs/2203.02155.

Guanghui Qin and Jason Eisner. 2021. Learning how to ask: Querying lms with mixtures of soft prompts. In *NAACL-HLT*, pages 5203–5212. Association for Computational Linguistics.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. 2020. Exploring the limits of transfer learning with a unified text-to-text transformer. *J. Mach. Learn. Res.*, 21:140:1–140:67.

Ohad Rubin, Jonathan Herzig, and Jonathan Berant. 2022. Learning to retrieve prompts for in-context learning. In *NAACL-HLT*, pages 2655–2671. Association for Computational Linguistics.

Victor Sanh, Albert Webson, Colin Raffel, Stephen Bach, Lintang Sutawika, Zaid Alyafeai, Antoine Chaffin, Arnaud Stiegler, Arun Raja, Manan Dey, M Saiful Bari, Canwen Xu, Urmish Thakker, Shanya Sharma Sharma, Eliza Szczechla, Taewoon Kim, Gunjan Chhablani, Nihal V. 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 Févry, Jason Alan Fries, Ryan Teehan, Teven Le Scao, Stella Biderman, Leo Gao, Thomas Wolf, and Alexander M. Rush. 2022. Multitask prompted training enables zero-shot task generalization. In *ICLR*. OpenReview.net.

Mohammad Shoeybi, Mostofa Patwary, Raul Puri, Patrick LeGresley, Jared Casper, and Bryan Catanzaro. 2019. Megatron-lm: Training multi-billion parameter language models using model parallelism. *CoRR*, abs/1909.08053.

Tianxiang Sun, Yunfan Shao, Hong Qian, Xuanjing Huang, and Xipeng Qiu. 2022. Black-box tuning for language-model-as-a-service. In *ICML*, volume 162 of *Proceedings of Machine Learning Research*, pages 20841–20855. PMLR.

Thomas Wolf Philipp Schmid Zachary Mueller Sourab Mangrulkar Sylvain Gugger, Lysandre Debut. 2022. Accelerate: Training and inference at scale made simple, efficient and adaptable. https://github.com/huggingface/accelerate.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *NIPS*, pages 5998–6008.

Yizhong Wang, Swaroop Mishra, Pegah Alipoormolabashi, Yeganeh Kordi, Amirreza Mirzaei, Anjana Arunkumar, Arjun Ashok, Arut Selvan Dhanasekaran, Atharva Naik, David Stap, et al. 2022.

Super-naturalinstructions:generalization via declarative instructions on 1600+ tasks. In *EMNLP*.

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

Jason Wei, Yi Tay, Rishi Bommasani, Colin Raffel, Barret Zoph, Sebastian Borgeaud, Dani Yogatama, Maarten Bosma, Denny Zhou, Donald Metzler, Ed H. Chi, Tatsunori Hashimoto, Oriol Vinyals, Percy Liang, Jeff Dean, and William Fedus. 2022b. Emergent abilities of large language models. *CoRR*, abs/2206.07682.

Sang Michael Xie, Aditi Raghunathan, Percy Liang, and Tengyu Ma. 2022. An explanation of in-context learning as implicit bayesian inference. In *ICLR*. OpenReview.net.

Qinyuan Ye, Bill Yuchen Lin, and Xiang Ren. 2021. Crossfit: A few-shot learning challenge for cross-task generalization in NLP. In *EMNLP (1)*, pages 7163–7189. Association for Computational Linguistics.

Susan Zhang, Stephen Roller, Naman Goyal, Mikel Artetxe, Moya Chen, Shuohui Chen, Christopher Dewan, Mona T. 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. 2022a. OPT: open pre-trained transformer language models. *CoRR*, abs/2205.01068.

Yiming Zhang, Shi Feng, and Chenhao Tan. 2022b. Active example selection for in-context learning. *CoRR*, abs/2211.04486.

Yue Zhang, Hongliang Fei, Dingcheng Li, and Ping Li. 2022c. Promptgen: Automatically generate prompts using generative models. In *NAACL-HLT (Findings)*, pages 30–37. Association for Computational Linguistics.

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

A Detailed Partition

The training tasks of partitions {*Random*, *cls-to-cls*, *half-to-cls*} are the same as partition {1, 2.1, 2.2} in CrossFit (Ye et al., 2021). The test tasks in our partition are the combinations of *dev* and *test* tasks ($\mathcal{T}_{dev} \cup \mathcal{T}_{test}$) of the corresponding partition in CrossFit.

B Detailed Results

In this section, we show more detailed results on three partitions. Figure 6 and Figure 7 show task-level performance on *Random* partition with 16-shot demonstrations. Figure 8 shows the task-level performance on *cls-to-cls* partition with 16-shot demonstrations. Finally, Figure 9 shows the task-level performance on *half-to-cls* partition with 16-shot demonstrations.

C Potential Risks

PROMPTTRANS elicits in-context learning from backbone language models, which are pretrained with web-crawled corpus (Raffel et al., 2020). Although Raffel et al. (2020) has made efforts to remove bad words in pretrain corpus, it still potentially includes toxic or biased content. Thus we emphasize that PROMPTTRANS should be considered as a research prototype rather than a released product for real world users. Further efforts in reducing the risks is needed to deploy PROMPTTRANS for LM-as-a-service (Sun et al., 2022).

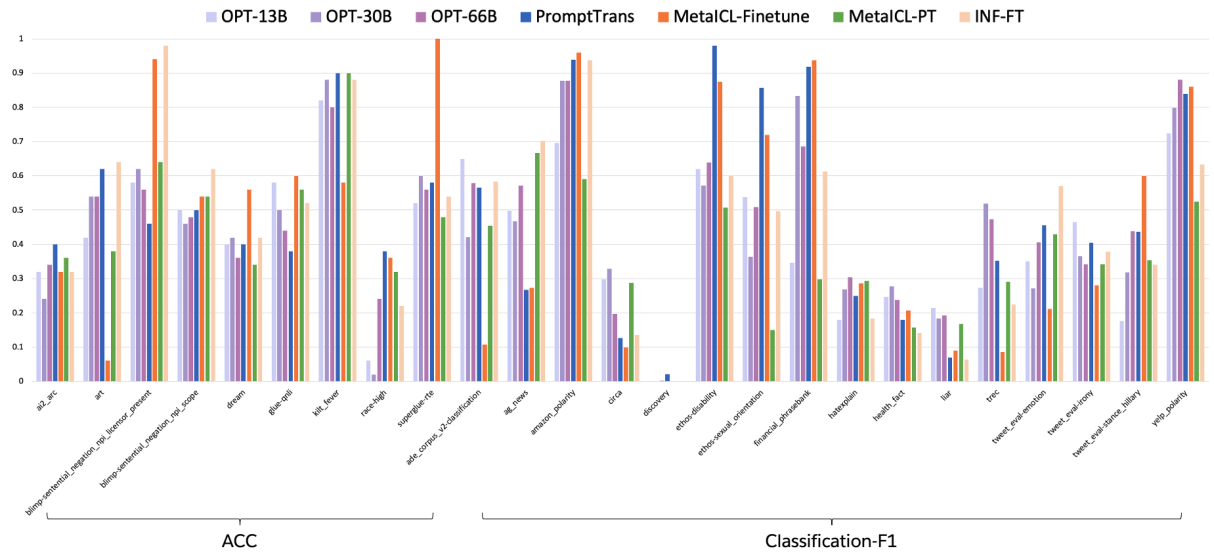

Figure 6: Detdetailed task-specific performance on *Random* 16-shot (a).

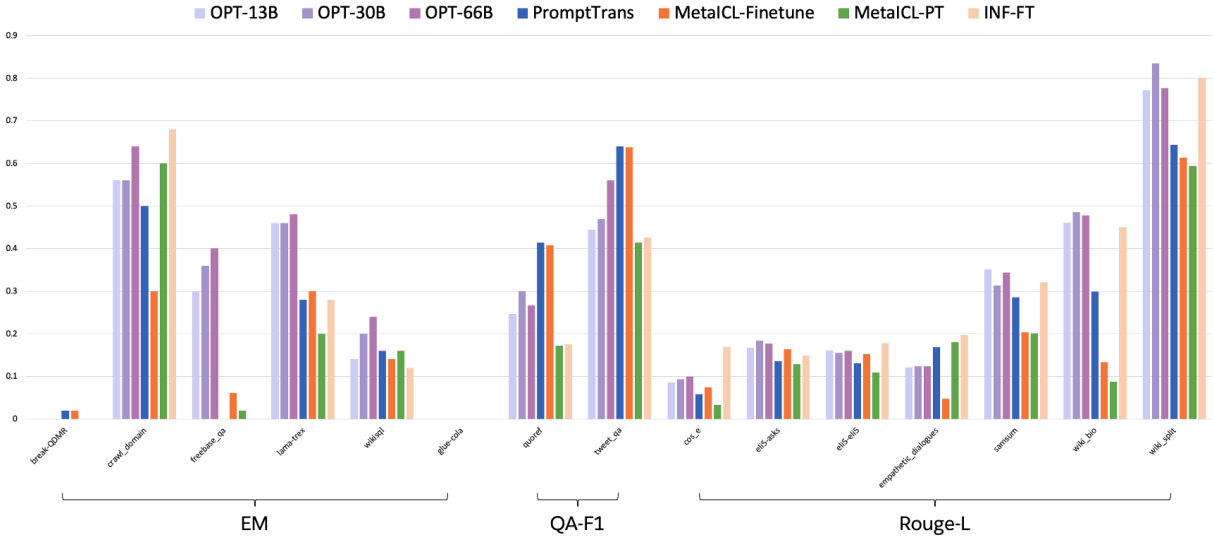

Figure 7: Detailed task-specific performance on *Random* 16-shot (b).

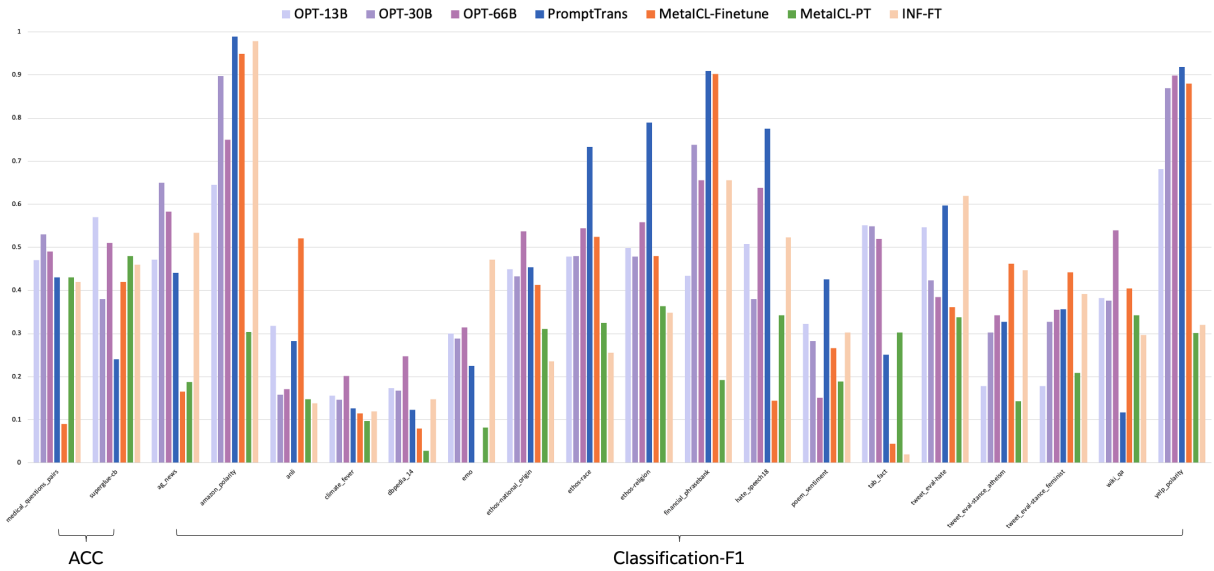

Figure 8: Detailed task-specific performance on *cls-to-cls* 16-shot

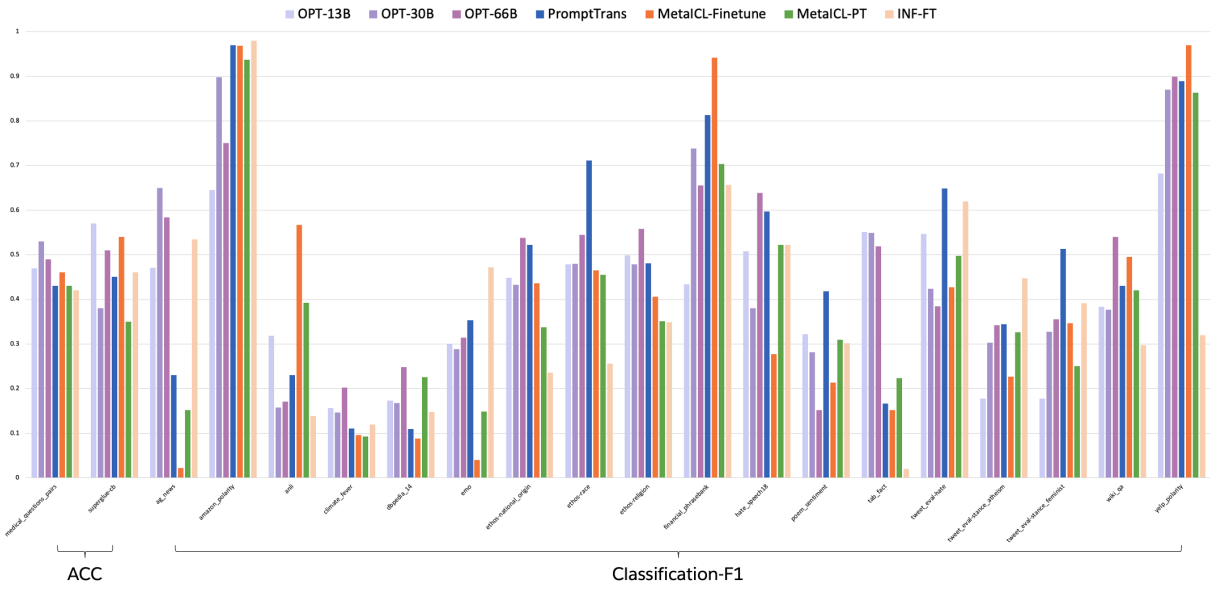

Figure 9: Detailed task-specific performance on *half-to-cls* 16-shot