Specializing Smaller Language Models towards Multi-Step Reasoning

Yao Fu[♠] Hao Peng[♣] Litu Ou[♠] Ashish Sabharwal[♣] Tushar Khot[♣]
[♠]University of Edinburgh [♣]Allen Institute for AI
{yao.fu, s1970716}@ed.ac.uk {haop, ashishs, tushark}@allenai.org

Abstract

The surprising ability of Large Language Models (LLMs) to perform well on complex reasoning with only few-shot chain-of-thought prompts is believed to emerge only in very large-scale models (100+ billion parameters). We show that such abilities can, in fact, be distilled down from GPT-3.5 ($\geq$ 175B) to T5 variants ($\leq$ 11B). We propose *model specialization*, to specialize the model's ability towards a target task. The hypothesis is that large models (commonly viewed as larger than 100B) have strong modeling power, but are spread on a large spectrum of tasks. Small models (commonly viewed as smaller than 10B) have limited model capacity, but if we concentrate their capacity on a specific target task, the model can achieve a decent improved performance. We use multi-step math reasoning as our testbed because it is a very typical emergent ability. We show two important aspects of model abilities: (1). there exists a very complex balance/ tradeoff between language models' multi-dimensional abilities; (2). by paying the price of decreased generic ability, we can clearly lift up the scaling curve of models smaller than 10B towards a specialized multi-step math reasoning ability. We further give comprehensive discussions about important design choices for better generalization, including the tuning data format, the start model checkpoint, and a new model selection method. We hope our practice and discoveries can serve as an important attempt towards specialized smaller models in the new research paradigm set by LLMs.

1. Introduction

Recently, the field of NLP is significantly impressed by large language models' strong abilities (Brown et al., 2020; Chowdhery et al., 2022). Wei et al. (2022a) discuss the emergent abilities of large language models – abilities that seems to only exist in large models (more than 100B parameters), but not in small models. A very typical example (also the first discovered emergent ability) is to perform multi-step reasoning on math word problems by chain-of-thought (CoT) prompting (Wei et al., 2022b) where the authors let the model generate a step-by-step reasoning chain to help get the final answer. The existence of such abilities has a very deep, profound influence on the community: on the positive side, such abilities open countless opportunities for new research directions; on the negative side, very few organizations have the compute to even fine-tune 100B-scale models, making the accessibility of such abilities extremely hard. It would be ideal if smaller models can also obtain emergent abilities like math CoT reasoning, so they can be accessed by a larger range of researchers and practitioners. However, preliminary results of Wei et al. (2022a) show that if the model scale is small (empirically less than 100B parameters), CoT exhibits flat, sometimes even near zero scaling curve (Wei et al., 2022b). Later smaller models' scaling curve is partially improved in Chung et al. (2022), but still worse than large models. These results so far are rather pessimistic since they suggest increasing CoT performance for smaller models can be challenging. At the current stage, the community is eager to know to what extent such abilities can be further improved in smaller models.

This paper addresses the problem of CoT reasoning for smaller models by *model specialization*. Our hypothesis is that large models ($\geq$ 100B) have strong modeling power but are spread over a large spectrum of tasks. Small models ($\leq$ 10B) have limited model capacity, but if we concentrate their capacity on a target task, the model may still have a decent improved performance. There exists promising preliminary work on smaller models' chain-of-thought abilities such as UL2 (Tay et al., 2022) and FlanT5 (Chung et al., 2022), but they focus on generic abilities and consequently, the model's (limited) power is not concentrated. In our experiments, we show that by paying the price of decreased abilities in generic tasks (specifically we lose a large portion of accuracy on the BigBench Hard suite Suzgun et al., 2022), we can lift the scaling curve of CoT reasoning on small FlanT5 models (250M, 760M, and 3B) by a large margin (an average +10 accuracy gain) on a suite of 4 math reasoning tasks (1 in-distribution and 3 out-of-distribution). This means that we can indeed move the model's power from generic abilities to concentrate on the target math CoT.

Our approach is to fine-tune an instruction-tuned model (FlanT5) by distilling chain-of-thought reasoning paths of the GSM8K data from a large teacher model (GPT-3.5 code-davinci-002 Chen et al., 2021), then do a model selection on the average performance of three held-out math reasoning data to ensure the model's out-of-distribution generalization. Although distillation per se is a well-studied area, there are multiple caveats in our process, as we will demonstrate: (1). the teacher model code-davinci-002 and our student model FlanT5 use different tokenizers, we address the tokenizer alignment problem by dynamic programming. (2). Distillation induces different performance on an instruction-tuned checkpoint (in our case, FlanT5) and the raw pretrained checkpoint (T5), where specialized FlanT5 performs better but specialized T5 achieves more accuracy gain. (3). at the late training stage, the model's in-distribution and out-of-distribution (OOD) performance fluctuates differently, so if one wants better OOD generalization, the model selection should be performed on held-out math datasets, rather than the validation portion of the tuning data. (4). multiple trade-offs happen during the distillation/ specialization process: as we start distillation, on BigBench Hard test suite (the measure of generic ability), the model immediately loses all its CoT prompting abilities, and gradually loses a large portion (but not all) of answer-only prompting abilities. The data format we use for tuning is also closely related to model ability: in-context examples enable both in-context and zero-shot performance, but zero-shot examples lose the model's in-context ability for increased zero-shot ability.

These findings deepen our understanding of language model chain-of-thought reasoning behavior in multiple aspects: (1). the previous hypothesis is that CoT has near-flat scaling curves on small scale, we show that we can lift up the scaling curve by concentrating the model's capacity on a target ability. This indicates that chain-of-thought might not be an emergent ability because, after specialization, smaller models' scaling curves become log-linear, just like large models (Kaplan et al., 2020; Hoffmann et al., 2022). (2). previous observation of LLM behaviors indicates complex tradeoffs and balances of model ability across multiple dimensions, we give a detailed description of how we move the model's power from generic abilities to a target ability, clearly showing what can be gained at what cost. (3). classical model selection theory selects the model on the validation portion of the same dataset, we select the model based on the performance of different math reasoning datasets, to prevent overfitting on one single dataset. We hope our practice and discoveries can serve as an example attempt towards strong specialized smaller models.

2. Background

Large Language Models' Abilities Large language models have significantly changed the research paradigm in NLP by showing strong abilities on multiple dimensions (Brown et al., 2020; Hoffmann et al., 2022; Chowdhery et al., 2022; Wei et al., 2022a). Currently, the new recipe for training LLMs is to first train a base model (e.g., GPT-3, PaLM, OPT), then elicit the abilities of the base model by instruction tuning (e.g., GPT-3 $\rightarrow$ InstructGPT Ouyang et al., 2022; PaLM $\rightarrow$ FlanPaLM Chung et al., 2022, OPT $\rightarrow$ OPT-IML Iyer et al., 2022, also see Fig. 1A step 1 and 2). For the base model, initially, Wei et al. (2022b) shows that the chain-of-thought performance curve is near-zero if the model size is smaller than 100B. Later Chung et al. (2022) updated this hypothesis by showing CoT can be unlocked if CoT data is included as one particular type of instruction, but their model's performance is not as good because their model's ability is spread over multiple dimensions. This work shows that CoT performance can be significantly lifted if we concentrate model's power toward a target ability (Fig. 1A, step 3).

Specialized Language Models Although modern language models show strong generic abilities on multiple directions, recent analysis (Fu et al., 2022) shows models do have different focuses (e.g., code-davinci-002 for code and text-davinci-003 for text). Ability tradeoff happens at all scale: for large models, such a tradeoff does not have to be all or nothing: code-davinci-002, although specialized for code, can still solve a lot of text problems; for small models, due to limited model capacity, they have to trade all generic abilities for one special ability. One example is GitHub Copilot, which supposedly is a 12B small model (Thakkar, 2022). The actual practice of specialization is simply finetuning: to specialize a model towards a target ability, one simply tunes the model using the related data, which is the practice of concurrent work about smaller models' CoT ability (Magister et al., 2022; Shridhar et al., 2022; Ho et al., 2022). The problem here is how to generalize beyond the tuning data, as small models may simply overfit the tuning distribution but struggle to generalize when the distribution shifts (Liu et al., 2022; Si et al., 2022). So far the community's hypothesis of OOD generation involves two important aspects: (1). model scale (Chowdhery et al., 2022); (2). instruction tuning (Chung et al., 2022), which we will also study. These factors mark the differences between our work and the concurrent distillation work: we show how the model trades generic abilities for the target ability, and how model scale and instruction tuning help the model gain better in-distribution and OOD performance.

Distillation and Data Augmentation Our approach of using data generated from code-davinci-002 to tune smaller FlanT5 can be viewed as either distillation (Tan et al., 2019)

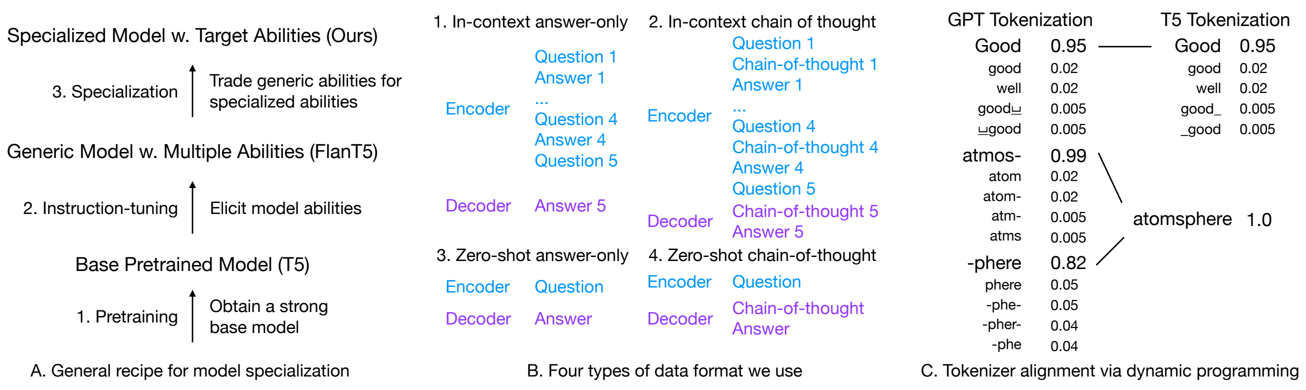

Figure 1. **A.** Model specialization process. Pretraining gives a strong base model (Raffel et al., 2020; Chowdhery et al., 2022), instruction tuning elicits the model ability (Chung et al., 2022), then specialization (this work's focus) moves model abilities to a target direction. In this work, we trade the model's generic abilities (as measured by BigBench Hard) for the model's multi-step math reasoning abilities. **B.** Four data formats we consider for tuning the model. We will show tuning with in-context chain-of-thought examples is particularly important for the model's CoT ability. **C.** Aligning GPT tokenization to T5 tokenization by dynamic programming. If a T5 token has a one-to-one alignment to a GPT token, we reuse the GPT's top 5 probability as the target distribution. If there the mapping is one-to-many/ many-to-one, we treat the T5 token's distribution as one-hot.

or data augmentation (Li et al., 2022). Here we note that we merely use the generated data as the tool for model specialization, and the specialization data can also be from other sources like human annotation. Our focus is to study the ability tradeoff during specialization, but not directly contribute to the distillation or data augmentation literature.

Most closely related works There are two threads of most related works: (1). FlanT5 (Chung et al., 2022) and UL2 (Tay et al., 2022) which is the first work discussing smaller models' CoT ability, but they focus on generic CoT while we trade generic ability for math CoT. (2). language model self-improvement (Huang et al., 2022) which also use CoT data augmentation, but they only consider large models and do not show the tradeoff between model abilities. Here we focus on small models and clearly show the price for ability improvements.

3. Specializing Multi-Step Reasoning

Our objective is to study what it takes to improve smaller models' chain-of-thought math reasoning. We use GSM8K (Cobbe et al., 2021) as our seed dataset because it is one of the datasets with most diverse math reasoning problems, but test the model's performance of three additional math datasets (MultiArith, ASDiv, and SVAMP Wei et al., 2022b) to show the model generalizes to OOD data. We further use BigBench Hard to test to model's generic reasoning ability, demonstrating the tradeoff between generic and target abilities. We use T5 (raw pretrained checkpoint) and FlanT5 (instruction tuned checkpoint) as our base model, and use code-davinci-002 to generate distillation/ specialization data.

Distillation from Code-Davinci-002 Given a training question corpora, we use code-davinci-002 to generate 40 new CoT solutions then take the ones that lead to the correct answers as our training data. One solution consists of an answer and a chain of thought explaining the intermediate steps towards the answer. In addition to the standard fine-tuning setting where one uses the question as the input and use the [CoT, answer] pair as the output (Fig. 1 B4), we further consider three additional data formats: (1). in-context answer-only (Fig. 1 B1), where we do not use the CoT data (hence the name "answer-only") and prepend 4 in-context examples before the question (hence the name "in-context"). The reason we prepend the in-context example is that previous work shows tuning with in-context examples improves the model's in-context learning ability (Min et al., 2022). (2). in-context chain-of-thought (Fig. 1 B2), where we add CoT to both the in-context example and the output. (3). zero-shot answer-only, where we directly input the question and output the answer. Using answer-only data is because previous work shows they improve performance. In our experiments, we will show that in-context data induces zero-shot ability but zero-shot data sacrifice in-context learning ability. We note that there also exist techniques like adding a calculator (Cobbe et al., 2021) or self-consistency decoding (Wang et al., 2022) that can further improve the performance. These techniques are orthogonal to the distillation we use and can definitely be integrated to our work for better performance. Since our focus is the balance of the models' special and generic abilities, we leave the integration of these orthogonal techniques to future work.

In terms of training objectives, in the distillation literature, there are typically two types of distillation approaches: (1).

sample matching, where one trains the student model on the data generated by the teacher. In our case, sample matching means we directly optimize the student's likelihood on the data generated by code-davinci-002. (2). distribution matching, where one minimizes the KL divergence between the student's output distribution (in our case, the per-step autoregressive distribution) and the teacher's. Usually, distribution matching is shown to achieve faster convergence and better performance than sample matching, so we use distribution matching as our training objective. Distribution matching has an additional challenge in storing the distribution parameter: at each step, we need to store the whole distribution defined on the vocabulary $\mathcal{V}$, so the size of the dataset is $|\mathcal{V}|$ times larger than sample matching. Yet the OpenAI API only grants access to the 5 most probable tokens at each decoding step, but not the probability distribution over the entire vocabulary. Although the per-step distribution only covers the top 5 tokens, most of the time their probability sum is close to 1, being a good enough approximation of the full vocabulary distribution. We set to zero the probabilities of tokens not in the top 5.

Aligning tokenizers by dynamic programming One problem when matching the two distributions is the misalignment between the GPT tokenizer and the T5 tokenizer. We solve this problem by dynamic programming. Specifically, given two sequences to tokens $[\mathbf{s}_{1:L}, \mathbf{t}_{1:N}]$, our objective is to find an alignment that minimizes the total cost of editing one sequence to the other. Our dynamic program is a slight tweak of the textbook dynamic programming algorithms used in bioinformatics for sequence alignment (such as the Needleman–Wunsch algorithm (Needleman & Wunsch, 1970)) and in signal processing (such as dynamic time wrapping (Senin, 2008)). The recursion function is:

$$f(i,j) = \min\{f(i-1,j) + c(\mathbf{s}_i, \mathbf{t}_j), \quad (1)$$
$$f(i,j-1) + c(\mathbf{s}_i, \mathbf{t}_j), \quad (2)$$
$$f(i-1,j-1) + c(\mathbf{s}_i, \mathbf{t}_j)\} \quad (3)$$

where $f(i,j)$ denotes the total cost aligning $\mathbf{s}_{1:i}$ and $\mathbf{t}_{1:j}$ and $c(\mathbf{s}_i, \mathbf{t}_j)$ is the predefined string edit distance between token $\mathbf{s}_i$ and $\mathbf{t}_j$. our algorithm does not enforce one-on-one matching between tokens in the two sequences, and one token in $\mathbf{s}$ might align with multiple in $\mathbf{t}$ and vice versa Fig. 1C gives an example alignment. If there exists a one-to-one mapping between a GPT token and a T5 token, we use the GPT distribution as the T5 distribution. If the mapping is not one-to-one, e.g., two T5 tokens map to one GPT token, or two GPT tokens map to one T5 token (Fig. 1 C lower part), we do not use the corresponding GPT distribution and set the T5 distribution to be one-hot. We further note that aligning sequences generated by different tokenizers is a generic problem of contemporary NLP, yet we are not aware of any existing libraries approaching it. We plan to release the implementation of our dynamic program and hope it can be useful for future research.

4. Experiments

The objective of the experiments is to see to what extent we can lift up the scaling curve of smaller models' math CoT performance and what is the price of it. We conduct model specialization on two model families: the raw pretrained checkpoints, and their instruction-tuned checkpoints (recall that the instruction-tuned checkpoints are generally more capable than the raw pretrained checkpoints, Fig 1A). Specifically, we consider the raw pretrained T5 Base (250M)/ Large (760M)/ XL (3B)/ XXL (11B), and the instruction-tuned FlanT5s. In Sec. 4.1, we validate our main hypothesis that large models can perform well on a wide range of tasks while smaller model's ability can be moved from generic abilities to a specialized target ability. Specifically, we show model specialization can indeed improve CoT math performance for FlanT5-Base/ Large/ XL/ XXL, while paying the price of generic abilities, i.e., losing all CoT abilities on BigBench Hard and a large portion of answer-only (AO) abilities. In Sec. 4.2, we study the scaling behavior of smaller models and show how specialization lifts up the scaling curve for both T5 and FlanT5. This modifies the previous belief that smaller models exhibit a flat scaling curve (Wei et al., 2022b); we show that their scaling curve becomes log-linear after specialization, but not flat. In Sec 4.3, we show the dynamics and the generalization behavior of specialization: the model's target performance increases gradually but generic abilities decrease gradually during tuning, and there exists tradeoffs between in-distribution v.s. OOD performance and in-context v.s. zero-shot performance.

4.1. Overall Performance Tradeoff

We test the models' math reasoning ability and generic ability and show their tradeoffs. For the math reasoning ability, we use the code-davinci-002 augmented GSM8K dataset (Cobbe et al., 2021) as our tuning dataset. The GSM8K has 7K training questions, for each question we ask the large model to generate 40 different solutions, taking the correct ones from the generation, we have 130K tuning data points in total. We test the model's out-of-distribution performance on MultiArith, ASDiv, and SVAMP (collectively denoted as M-A-S) datasets (Wei et al., 2022b). None of the datasets has official train-dev-test splits, so we randomly sample 500 instances as the validation set, and use the remaining instances (800 for GSM8K, 400 for MultiArith, 18K for ASDiv, 500 for SVAMP) as the test set. The difference between M-A-S and GSM8K is that they are all primary school level arithmetic reasoning problems, but the entities involved in the datasets are different. For example, GSM8K may consider arithmetic reasoning on foods (e.g, 5 apples + 8 bananas = 13 fruits) and MultiArith may con-

Table 1. Overall test set performance. We specialize Flan-T5's ability from the generic tasks (BigBench Hard) to math reasoning tasks. After paying the cost of BigBench Hard performance (the model loses all the CoT prompting ability and a large portion of the Answer-only (AO) prompting ability), we see the specialized T5 models have improved in-distribution (GSM8K) performance (where our 3B and 11B models outperform concurrent works) as well as out-of-distribution (MultiArith, ASDiv and SVAMP) performance, showing that we can move the model's ability from generic tasks (BBH) to a specific target task (math reasoning). Magister22: Magister et al. (2022); Shridhar22: Shridhar et al. (2022); Ho22: Ho et al. (2022).

		CoT Reasoning on Maths Word Problems								**BigBench-Hard**			
		GSM8K		**MultiArith**		**ASDiv**		**SVAMP**		**AO**		**CoT**	
Models	**#Params.**	**Acc.**	**Δ**	**Acc.**	**Δ**	**Acc.**	**Δ**	**Acc.**	**Δ**	**Acc.**	**Δ**	**Acc.**	**Δ**
code-davinci-002	≥175B	63.1	-	95.8	-	80.4	-	76.4	-	56.6	-	73.9	-
LaMDA	137B	14.8	-	45.0	-	46.6	-	37.5	-	-	-	-	-
PaLM	60B	29.9	-	75.0	-	61.9	-	46.7	-	37.4	-	43.0	-
UL2	20B	4.4	-	-	-	16.9	-	12.5	-	-	-	-	-
Concurrent Works with Knowledge Distillation													
Magister22, T5	11B	21.9	-	-	-	42.1	-	-	-	?	-	?	-
Shridhar22, GPT	6B	21.0	-	-	-	-	-	-	-	?	-	?	-
Ho22, GPT	6B	6.8	-	33.3	-	-	-	-	-	?	-	?	-
Our Specialized Models Compared with Baselines													
FlanT5-XXL	11B	16.1	-	51.7	-	36.5	-	39.7	-	47.4	-	41.8	-
+ Specialized	11B	27.1	+11.0	63.0	+11.3	37.6	+1.1	35.6	-4.1	19.6	-27.8	0.0	-41.8
FlanT5-XL	3B	13.5	-	24.0	-	20.7	-	17.7	-	39.9	-	35.8	-
+ Specialized	3B	22.4	+8.9	42.3	+18.3	28.4	+7.7	23.8	+6.1	3.2	-36.7	0.0	-35.8
FlanT5-Large	760M	6.9	-	13.0	-	10.1	-	6.8	-	30.3	-	30.9	-
+ Specialized	760M	20.2	+13.3	38.5	+25.5	23.8	+13.7	20.4	+13.6	6.5	-23.8	0.3	-30.6
FlanT5-Base	250M	3.0	-	7.0	-	4.2	-	3.8	-	24.2	-	25.9	-
+ Specialized	250M	13.4	+10.4	29.7	+22.7	20.9	+16.7	14.2	+10.4	3.1	-21.1	0.1	-25.8

sider animals (e.g., 2 dogs + 3 cats = 5 animals). This type of out-of-distribution generalization is usually referred to as lexical-level compositional generalization (i.e., both are addition, but the lexicons are different, see Liu et al., 2022). For the generic ability, we use BigBench Hard (BBH, Suzgun et al., 2022) test suite, a list of 26 challenging dataset testing the model's reasoning abilities from multiple dimensions (e.g., date understanding, causal judgement, referential game, .etc). Because of its difficulty and wide-coverage, BBH makes an ideal benchmark testing models' generic ability.

For the baseline models, we consider generic large models and concurrent smaller distilled models, specifically: (1). generic large models, ranked according to scale: code-davinci-002 (our teacher model, presumably larger or equal to 175B); LaMDA 137B (Thoppilan et al., 2022) and PaLM 60B (Chowdhery et al., 2022), both are strong generic models for chain-of-thought reasoning; UL2 (Tay et al., 2022), a 20B model with good CoT ability. We will show that specialized FlanT5 11B outperforms UL2 20B and becomes close to PaLM 60B and LaMDA 137B on the target math reasoning task. (2). concurrent works with knowledge distillation from Magister et al. (2022); Shridhar et al. (2022); Ho et al. (2022). We will show that our specialized FlanT5 clearly outperform all of them on the distillation data (with the cost of BBH performance), mostly because we use an instruction-tuned checkpoint (FlanT5) as the base model rather than the raw pretrained checkpoint (T5).

Trading generic abilities for math CoT reasoning The overall results are in Table 1. After tuning on the seed GSM8K augmented data, all FlanT5 models have improved math reasoning performance with approximately +10 average accuracy gain. We note that our smaller 3B model outperforms the current 11B and 6B distillation models on the GSM8K test set. Despite multiple confounders including the size and the formats of tuning data, we believe our 3B model gets a better performance mostly because the base model is an instruction-tuned FlanT5, rather than the raw pretrained T5. Later we will show that instruction-tuned checkpoint consistently outperforms pretrained checkpoint after specialization (Sec. 4.2), showing the importance of the choice of the base model. Also, although not performing

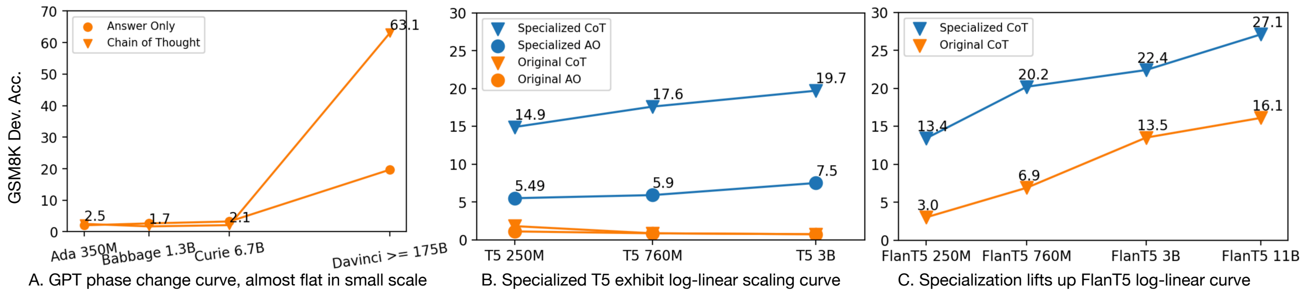

Figure 2. X-axis means log of model scale, y-axis means validation accuracy on GSM8K. **A**: Previously, the community believe that small models has flat curve for both AO and CoT prompting and only when models become large enough the performance will have a "phase change" and suddenly increase. **B**: we show that after training on CoT, the model exhibits log-linear curves where both AO and CoT increase with model scale. **C**: for instruction-tuned models (FlanT5) that already exhibit CoT, specialization lifts up the scaling curve, and the two curves are again, log-linear shaped. All the log-linear curves indicate that chain-of-thought may not be an emergent ability which is marked by the flat-then-phase-change curve. Here we show the curve in small scale is not flat but actually log-linear, and continuously increasing model scale leads to continuously increased accuracy (no sudden phase change).

Table 2. GSM8K validation performance. Instruction-tuned models generally performs better than the raw pretrained checkpoints.

Before	Acc	After	Acc
FlanT5 3B	**13.5**	Specialized	**23.8**
T5 3B	0.73	Specialized	20.6
FlanT5 760M	**6.9**	Specialized	**21.8**
T5 760M	0.85	Specialized	16.2
FlanT5 250M	**3.0**	Specialized	**15.2**
T5 250M	1.8	Specialized	14.2

well as the teacher model code-davinci-002, our specialized 11B model performance improves to be on par with LaMDA 137B and slightly below PaLM 60B, showing it is indeed possible to make smaller models expert for the particular math reasoning task. The price is also very clear: all specialized models suffer from performance drop on BigBench, specifically, they lose all the CoT prompting abilities on BBH, and a large portion of AO prompting performance. This observation validates our hypothesis: large models can perform well on a wide range of tasks (here PaLM 60B perform well on both math reasoning and BBH), versus smaller model's ability can be moved from generic tasks (BBH) to a specialized target ability (math reasoning), such that their performance on the target task can still match models that are larger than them, e.g., the average performance on the four math datasets LaMDA 137B 35.9 v.s. specialized FlanT5 11B 40.8.

4.2. Scaling Behavior of Smaller Models' CoT Ability

Now we look the scaling behavoir to smaller models. We compare the scaling curve of: (1). GPT family small variants (Ada, Babbage, Curie and code-davinci-002); (2). raw pretrained T5 of different scales and their specialized versions; (3). the instruction-tuned FlanT5 of different scales and their specialized versions; The results are shown in Fig. 2 where x-axis denotes the model scale in terms of the number of parameters and y-axis denotes the validation accuracy on the GSM8K dataset.

Smaller models have log-linear, but not flat scaling curve Initially, in the original CoT paper Wei et al. (2022b) and the subsequent emergent abilities paper (Wei et al., 2022a), CoT prompting is believed to be an emergent property that only large models exhibit. Smaller model's CoT performance (like smaller GPT variants) was believed to be a flat scaling curve: model performance does not improve with model scale, as is shown in Fig. 2A left part. Later this belief is updated by the FlanT5 paper (Chung et al., 2022), as they show that although the pretrained checkpoint does not have CoT ability, if the model has gone through instruction tuning, smaller models can still exhibit CoT on generic tasks. Our work shows that directly trained on CoT data can also lift up the flat scaling curve of the raw T5 checkpoints (Fig. 2B) to be log-linear. In Fig. 2C, we consider specialization for the instruction-tuned FlanT5, and show that specialization significantly lifts up the scaling curve of FlanT5, and both curves are also log-linear. All the log-linear curves we observed in Fig. 2 means that the chain-of-thought behavoir of smaller models are not flat, but actually log-linear. This further indicates that chain-of-thought may not be an emergent ability which is marked by the flat-then-phase-change curve, but they have the log-linear curve just like large models (Kaplan et al., 2020; Hoffmann et al., 2022).

Instruction-tuned checkpoints perform better than raw pretrained checkpoints Furthermore, comparing

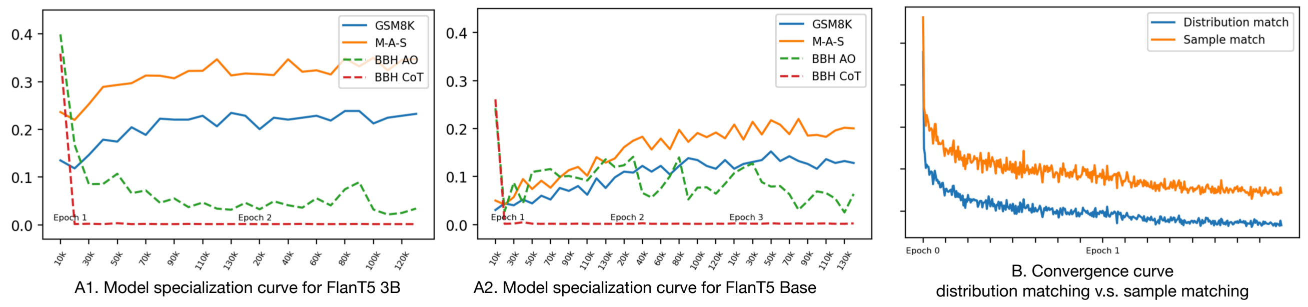

Figure 3. **A1 and A2**: model specialization curve of FlanT5. At the beginning of specialization (e.g., A1 step 10K), the model immediately loses all BBH CoT ability, and a large portion of BBH AO ability. As tuning goes on (e.g., A1 epoch 1), the model's in-distribution performance (GSM8K) and out-of-distribution performance (MultiArith-ASDiv-SVAMP, M-A-S) gradually increases. At the later stage of tuning (e.g., A1 epoch 2), the model's math performance fluctuates and better in-distribution performance does not indicate better out-of-distribution performance. Smaller models need to see the data more times than larger models (A2 has 3 epochs and A1 has 2). **B**: differences between two distillation approaches. Distribution matching gives faster and lower loss convergence than sampling matching.

Fig. 2B and Fig. 2C, we see that specialized FlanT5 generally performs better than T5 (though T5 has a larger performance gain). The exact validation performance is shown in Table 2. We also believe that, despite there exist multiple confounders, a major reason that our performance in Table 1 (FlanT5 11B GSM8K accuracy 27.1) is better than concurrent distillation methods (Magister22 T5 11B, acc. 21.9) is mostly because we use the FlanT5 as our base model versus they use the raw pretrained T5. The intuitive explanation is because instruction-tuning elicits the model's full ability while raw pretrained models' ability are not fully released (conceptually see Fig. 1A, also see Fu et al., 2022; Chung et al., 2022). So for better performance, we recommend using instruction-tuned models in practice.

4.3. Specialization Process and Generalization Behaviors

Now we consider the specialization process. Intuitively, during finetuning, the model's ability does not suddenly become the target ability, but will go through a process of moving the models' ability from generic directions to the target. We save one checkpoint every 10K instances/ updates, then evaluate the checkpoints on (1). in-distribution math performance (GSM8K); (2). out-of-distribution math performance (MultiArith, ASDiv, and SVAMP); (3). generic answer-only prompting performance (BBH-AO); (4). generic chain-of-thought prompting performance (BBH-CoT). We plot the model's performance across the fine-tuning process in Fig. 3.

The dynamics of model specilization. At the beginning of specialization (Figure A1 at step 10K and Figure A2 at step 20K), the model immediately loses all BBH CoT ability (accuracy becomes 0), and a large portion of BBH AO ability (accuracy drops from about 0.3 to about 0.1). As tuning goes on (A1 epoch 1, A2 epoch 1 and 2), the model's in-distribution performance (GSM8K) and out-of-distribution performance (MultiArith-ASDiv-SVAMP, M-A-S) gradually increases, meaning that the model can generalize to three OOD datasets by tuning on GSM8K chain-of-thought data. At the later stage of tuning (Figure A1 at epoch 2, and Figure A2 at epoch 3), the model's math performance fluctuates and better in-distribution performance does not indicate better out-of-distribution performance. The models' BBH-AO performance drops a large portion and the BBH-CoT performance just die completely. Comparing A1 and A2, we also see that smaller models are more data-hungry than larger models (Kaplan et al., 2020): FlanT5 3B's math performance plateaus at about 90K data points, versus FlanT5 Base's performance continues increase until epoch 3 (each epoch has 130K datapoints).

Table 3. Model selection method induces tradeoffs between in-distribution and out-of-distribution performance.

Model	Selection	In-dist	Out-of-dist
FlanT5 3B	GSM8K Dev	23.8	33.2
	M-A-S Dev	21.2 -2.6	35.0 +1.8
FlanT5 Large	GSM8K Dev	21.8	28.7
	M-A-S Dev	19.2 -2.6	30.5 +1.8
FlanT5 Base	GSM8K Dev	15.2	21.7
	M-A-S Dev	13.2 -2.0	22.0 +0.3

In-distribution and out-of-distribution tradeoffs Because in Fig. 3 A, both in-distribution and out-of-distribution fluctuates, choosing the best in-distribution checkpoint does not necessarily lead to the best out-of-distribution checkpoint. This observation is shown in Table 3 where if we select the best model based on the

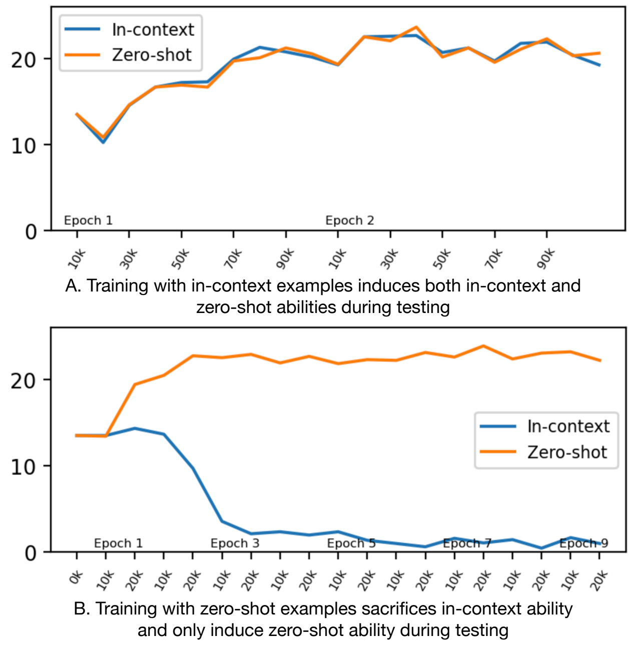

Figure 4. X-axis means tuning datapoints, y-axis means validation accuray on GSM8K. Both figures use FlanT5 3B as the base model. **A**: training with in-context examples automatically give the model zero-shot ability. **B**: training with zero-shot examples sacrifices in-context ability.

GSM8K validation set, it does cannot achieve the best validation performance on the M-A-S OOD setting. Yet choosing the best model based on the M-A-S validation performance leads to a smaller performance drop in GSM8K. Given this observation, in practice, we would recommend choosing the validation checkpoints according to the specific goal: if the goal is in-distribution generalization, use GSM8K, if the goal is OOD generalization, users may want to use their own validation set (in our case, the M-A-S datasets).

4.4. Further Design Choices Analysis

In this section, we study two more design choices we have discussed before: (1). using distribution matching v.s. sample matching for distillation (recall distillation matching minimizes the KL divergence between FlanT5's per-step autoregressive distribution and GPT's autoregressive distribution, versus sample matching maximizes the likelihood of the reasoning paths generated by GPT); (2). the influence of data formats, and how in-context/ zero-shot training data induces different behaviors of the specialized model.

Distribution matching gives faster convergence than sample matching. Fig. 3 B shows the training loss of distribution matching v.s. sample matching. We show that the model converges faster under distribution matching, and the corresponding loss is lower. In terms of validation performance, these two approaches do not differ substantially. Yet since distribution matching has a faster convergence, in practice they may still be considered first especially when the model becomes large and tuning becomes expensive.

In-context data preserves zero-shot ability; Zero-shot data loses in-context ability This is actually a very interesting observation. Specifically, in Fig. 4 A, we tune the model with only in-context data (Format B1 and B2 in Fig 1), then test the models in-context learning and zero-shot generalization performance during validation. In Fig. 4 B, we tune the model with only zero-shot data (no in-context examples prepended, format B3 and B4 in Fig 1), the test if the model can still do in-context learning. As is shown in Fig. 4 A, when tuning with in-context data, the model can do both in-context and zero-shot generalization during validation, even the model is not trained with zero-shot data. In comparison, in Fig. 4 B, when tuning with zero-shot data, the model's zero-shot performance increases, but gradually losses its in-context learning ability. This result aligns with the empirical observation on other large models, for example, text-davinci-002 has better zero-shot performance than code-davinci-002, but worse in-context learning performance (Fu et al., 2022). This means that the model's ability tradeoff not only happens on math v.s. generic ability, but also happens on zero-shot v.s. in-context learning ability. In practice, we would recommend mix the different data formats during tuning (this is why we mix the formats) to maintain a balance between in-context and zero-shot abilities, or adjusting the ratio of different formats according to the specific use case.

5. Conclusion

In this work, we study the problem of specializing smaller language models toward multi-step reasoning using chain-of-thought prompting. We show that it is indeed possible to concentrate the small models' ability from generic directions to the target math reasoning task. After specialization, we show that the model exhibits a log-linear scaling curve where model performance increases smoothly as model scale increases, this is a correction of the previous hypothesis which believes small models have a flat scaling curve that does not increase with model scale. We show the importance of using the instruction-tuned checkpoints as the base model because their generalization performance is better than the raw pretrained checkpoints. Mutiple tradeoff happens during model specialization, including the loss of BBH performance, the balance between in-distribution and out-of-distribution generalization, and the balance of in-context learning and zero-shot generalization ability. We hope our practice and discoveries can serve as an important attempt towards specialized smaller models in the new research paradigm set by LLMs

References

Brown, T., Mann, B., Ryder, N., Subbiah, M., Kaplan, J. D., Dhariwal, P., Neelakantan, A., Shyam, P., Sastry, G., Askell, A., et al. Language models are few-shot learners. *Advances in neural information processing systems*, 33: 1877–1901, 2020.

Chen, M., Tworek, J., Jun, H., Yuan, Q., Pinto, H. P. d. O., Kaplan, J., Edwards, H., Burda, Y., Joseph, N., Brockman, G., et al. Evaluating large language models trained on code. *arXiv preprint arXiv:2107.03374*, 2021.

Chowdhery, A., Narang, S., Devlin, J., Bosma, M., Mishra, G., Roberts, A., Barham, P., Chung, H. W., Sutton, C., Gehrmann, S., et al. Palm: Scaling language modeling with pathways. *arXiv preprint arXiv:2204.02311*, 2022.

Chung, H. W., Hou, L., Longpre, S., Zoph, B., Tay, Y., Fedus, W., Li, E., Wang, X., Dehghani, M., Brahma, S., et al. Scaling instruction-finetuned language models. *arXiv preprint arXiv:2210.11416*, 2022.

Cobbe, K., Kosaraju, V., Bavarian, M., Chen, M., Jun, H., Kaiser, L., Plappert, M., Tworek, J., Hilton, J., Nakano, R., et al. Training verifiers to solve math word problems. *arXiv preprint arXiv:2110.14168*, 2021.

Fu, Y., Peng, H., and Khot, T. How does GPT obtain its ability? tracing emergent abilities of language models to their sources. *Yao Fu's Notion*, Dec 2022. URL `https://yaofu.notion.site/b9a57ac0fcf74f30a1ab9e3e36fa1dc1`.

Ho, N., Schmid, L., and Yun, S.-Y. Large language models are reasoning teachers. *arXiv preprint arXiv:2212.10071*, 2022.

Hoffmann, J., Borgeaud, S., Mensch, A., Buchatskaya, E., Cai, T., Rutherford, E., Casas, D. d. L., Hendricks, L. A., Welbl, J., Clark, A., et al. Training compute-optimal large language models. *arXiv preprint arXiv:2203.15556*, 2022.

Huang, J., Gu, S. S., Hou, L., Wu, Y., Wang, X., Yu, H., and Han, J. Large language models can self-improve. *arXiv preprint arXiv:2210.11610*, 2022.

Iyer, S., Lin, X. V., Pasunuru, R., Mihaylov, T., Simig, D., Yu, P., Shuster, K., Wang, T., Liu, Q., Koura, P. S., et al. Opt-iml: Scaling language model instruction meta learning through the lens of generalization. *arXiv preprint arXiv:2212.12017*, 2022.

Kaplan, J., McCandlish, S., Henighan, T., Brown, T. B., Chess, B., Child, R., Gray, S., Radford, A., Wu, J., and Amodei, D. Scaling laws for neural language models. *arXiv preprint arXiv:2001.08361*, 2020.

Langley, P. Crafting papers on machine learning. In Langley, P. (ed.), *Proceedings of the 17th International Conference on Machine Learning (ICML 2000)*, pp. 1207–1216, Stanford, CA, 2000. Morgan Kaufmann.

Li, S., Chen, J., Shen, Y., Chen, Z., Zhang, X., Li, Z., Wang, H., Qian, J., Peng, B., Mao, Y., et al. Explanations from large language models make small reasoners better. *arXiv preprint arXiv:2210.06726*, 2022.

Liu, L., Lewis, P., Riedel, S., and Stenetorp, P. Challenges in generalization in open domain question answering. In *Findings of the Association for Computational Linguistics: NAACL 2022*, pp. 2014–2029, Seattle, United States, July 2022. Association for Computational Linguistics. doi: 10.18653/v1/2022.findings-naacl.155. URL `https://aclanthology.org/2022.findings-naacl.155`.

Magister, L. C., Mallinson, J., Adamek, J., Malmi, E., and Severyn, A. Teaching small language models to reason. *arXiv preprint arXiv:2212.08410*, 2022.

Min, S., Lewis, M., Zettlemoyer, L., and Hajishirzi, H. MetaICL: Learning to learn in context. In *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pp. 2791–2809, Seattle, United States, July 2022. Association for Computational Linguistics. doi: 10.18653/v1/2022.naacl-main.201. URL `https://aclanthology.org/2022.naacl-main.201`.

Needleman, S. B. and Wunsch, C. D. A general method applicable to the search for similarities in the amino acid sequence of two proteins. *Journal of molecular biology*, 48(3):443–53, 1970.

Ouyang, L., Wu, J., Jiang, X., Almeida, D., Wainwright, C. L., Mishkin, P., Zhang, C., Agarwal, S., Slama, K., Ray, A., et al. Training language models to follow instructions with human feedback. *arXiv preprint arXiv:2203.02155*, 2022.

Raffel, C., Shazeer, N., Roberts, A., Lee, K., Narang, S., Matena, M., Zhou, Y., Li, W., Liu, P. J., et al. Exploring the limits of transfer learning with a unified text-to-text transformer. *J. Mach. Learn. Res.*, 21(140):1–67, 2020.

Senin, P. Dynamic time warping algorithm review. 2008.

Shridhar, K., Stolfo, A., and Sachan, M. Distilling multi-step reasoning capabilities of large language models into smaller models via semantic decompositions. *arXiv preprint arXiv:2212.00193*, 2022.

Si, C., Gan, Z., Yang, Z., Wang, S., Wang, J., Boyd-Graber, J., and Wang, L. Prompting gpt-3 to be reliable. *arXiv preprint arXiv:2210.09150*, 2022.

Suzgun, M., Scales, N., Schärli, N., Gehrmann, S., Tay, Y., Chung, H. W., Chowdhery, A., Le, Q. V., Chi, E. H., Zhou, D., et al. Challenging big-bench tasks and whether chain-of-thought can solve them. *arXiv preprint arXiv:2210.09261*, 2022.

Tan, X., Ren, Y., He, D., Qin, T., and Liu, T.-Y. Multilingual neural machine translation with knowledge distillation. In *International Conference on Learning Representations*, 2019. URL https://openreview.net/forum?id=S1gUsoR9YX.

Tay, Y., Dehghani, M., Tran, V. Q., Garcia, X., Bahri, D., Schuster, T., Zheng, H. S., Houlsby, N., and Metzler, D. Unifying language learning paradigms. *arXiv preprint arXiv:2205.05131*, 2022.

Thakkar, P. Copilot explorer. *thakkarparth007.github.io*, 2022. URL https://thakkarparth007.github.io/copilot-explorer/posts/copilot-internals.html.

Thoppilan, R., De Freitas, D., Hall, J., Shazeer, N., Kulshreshtha, A., Cheng, H.-T., Jin, A., Bos, T., Baker, L., Du, Y., et al. Lamda: Language models for dialog applications. *arXiv preprint arXiv:2201.08239*, 2022.

Wang, X., Wei, J., Schuurmans, D., Le, Q., Chi, E., and Zhou, D. Self-consistency improves chain of thought reasoning in language models. *arXiv preprint arXiv:2203.11171*, 2022.

Wei, J., Tay, Y., Bommasani, R., Raffel, C., Zoph, B., Borgeaud, S., Yogatama, D., Bosma, M., Zhou, D., Metzler, D., Chi, E. H., Hashimoto, T., Vinyals, O., Liang, P., Dean, J., and Fedus, W. Emergent abilities of large language models. *Transactions on Machine Learning Research*, 2022a. URL https://openreview.net/forum?id=yzkSU5zdwD. Survey Certification.

Wei, J., Wang, X., Schuurmans, D., Bosma, M., Chi, E., Le, Q., and Zhou, D. Chain of thought prompting elicits reasoning in large language models. *arXiv preprint arXiv:2201.11903*, 2022b.