Improving Abstract Reasoning Ability of Large Language Models through Mixture Program-based Data Synthesis

Yile Wang, Hui Huang✉
College of Computer Science and Software Engineering, Shenzhen University
wangyile@szu.edu.cn, hhzhiyan@gmail.com

Abstract

Abstract reasoning is a challenging task that involves identifying patterns from limited input-output grids and applying them to new grids. With the development of large language models (LLMs), recent studies attempt to transfer the problems to textual format and tackle abstract reasoning tasks using models such as GPT-4. However, the overall accuracy is still low, which also results in the poor quality of abstract reasoning data directly synthesized by GPT-4, making it unsuitable as effective fine-tuning data. In this paper, we propose mixture program-based data synthesis strategies, including low-level code-based synthesis, high-level DSL-based synthesis, and shuffle-based synthesis. Through these strategies, we construct diverse and valid abstract reasoning instruction data to help improving the general abstract reasoning ability of LLMs for multiple datasets. Experimental results show that, by supervised fine-tuning Qwen-2.5-7B on our synthesized instruction data, the resulting model shows improved abstract reasoning ability and outperforms various strong baseline LLMs, including closed-source model GPT-4 and open-source models such as LLaMA-3 and Qwen-2.5. We release the logs by GPT and our model at https://github.com/szu-tera/ARC.

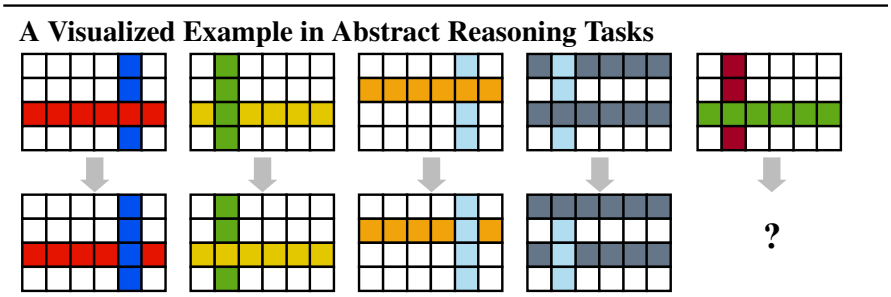
Keywords: Large Language Models , Abstract Reasoning , Data Synthesis

1 Introduction

Abstract reasoning tasks require uncovering underlying operations from a limited number of grid-based input-output pairs and predicting the output accordingly for a new test input, as shown in Table 1. This task is of great value in assessing current artificial intelligence (Chollet, 2019) and has gained the attention from researchers (Lee et al., 2024; Li et al., 2024; Chollet et al., 2024).

Recent studies find that large language models (LLMs) inherently possess some analogy reasoning (Webb et al., 2023) and pattern recognition (Mirchandani et al., 2023) capabilities, enabling them to solve few abstract reasoning problems. However, there still exists a gap compared with human (Mitchell et al., 2023; LeGris et al., 2024). In contrast to commonsense (Talmor et al., 2019) and mathematical reasoning (Cobbe et al., 2021) which LLMs excel at, the main challenges for abstract reasoning lies in three aspects:

1) Abstract reasoning tasks have a strict form, with some problems requiring complex prior knowledge and being quite challenging (Chollet, 2019). This results in a scarcity of manually designed data, making it difficult for sufficient model training. 2) Unlike the problems in natural language expressions, abstract reasoning problems are composed of transferred sequence of numbers. This less common data also poses challenges for LLMs to tackle such tasks (Wang et al., 2025). 3) Although there are some post-training chain-of-thought (Wei et al., 2022b) prompting techniques to elicit the reasoning ability of LLMs, they are primarily useful for math and symbolic problems (Sprague et al., 2024), and we also find that these methods are not particularly helpful for abstract reasoning tasks.



Data	Cost ↓	Validity ↑	#Samples ↑	Generalization ↑
By Human	High	High	Small	Low
By GPT	High	Low	Large	High
By Ours	**Low**	**High**	**Large**	**High**

Table 1: Top: An example in abstract reasoning tasks with input-output grids following underlying operations. Bottom: Comparing human-labeled, GPT-generated, and our program-based synthesized data.

In this work, we aim to tackle the above challenges from the perspective of data synthesis. Considering the characteristics of abstract reasoning task, we propose program-based data synthesis to construct more training data to further improve the abstract reasoning ability of LLMs. In particular, we introduce three strategies, including low-level code-based synthesis, high-level DSL (domain specific language)-based synthesis, and shuffle-based synthesis, which allow us to generate diverse and valid data at a lower cost for instruction tuning open-source LLMs. We find that the mixture of low-level and high-level data helps the generalization of abstract reasoning on diverse datasets.

We conduct supervised fine-tuning (Ouyang et al., 2022b) on Qwen-2.5-7B (Yang et al., 2024) using our synthetic instruction data and find that the resulting model show largely improved performance across four datasets, and it outperforms various closed-source and open-source LLMs, including GPT-3.5, GPT-4, GPT-4o, LLaMA-3, and Qwen-2.5, demonstrating the effectiveness of our program-based data synthesis. To the best of our knowledge, we are the first to evaluate general abstract reasoning ability on four complete datasets including ARC (Chollet, 2019), Mini-ARC (Kim et al., 2022), Concept-ARC (Moskvichev et al., 2023), and 1D-ARC (Xu et al., 2024), while most works focus on one of them or selected problems.

2 Method

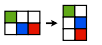
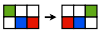
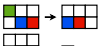
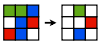
2.1 Task Formulation

Formally, a sample from the abstract reasoning task is defined as follows. Given n input grids $\{\mathcal{I}_i\}_{i=1}^n$ and n output grids $\{\mathcal{O}_i\}_{i=1}^n$ that satisfy $\mathcal{O}_i = \mathcal{T}(\mathcal{I}_i)$ for $i = 1, 2, ..., n$, where $\mathcal{T}$ represents an implicit operation that transforms each input grid $\mathcal{I}_i$ to its corresponding output grid $\mathcal{O}_i$, n is usually 3∼4. The task is to predict the output grid $\mathcal{O}_t$ for a given test input grid $\mathcal{I}_t$ following the transformation $\mathcal{T}$. Specifically, in abstract reasoning corpus (ARC) dataset (Chollet, 2019), $\{\mathcal{I}_i\}_{i=1}^n$, $\{\mathcal{O}_i\}_{i=1}^n$, $\mathcal{I}_t$, and $\mathcal{O}_t$ can be replaced by two-dimensional textual arrays, wherein the elements are integers 0 to 9 for indicating different colors.

The original training set of ARC dataset contains only 400 samples, and each of them differs in the transformation $\mathcal{T}$, which makes neural models difficult to learn from limited data and generalize to new samples for testing abstract reasoning ability. To compensate for the scarcity of abstract reasoning data and alleviate the difficulty of manually designing such complicated paired grids, we aim to automatically synthesize more diverse and valid data that contributes to enhancing abstract reasoning ability of LLMs. Specifically, we propose three program-based data synthesis strategies in following subsections.

2.2 Low-level Code-based Synthesis

Considering that the inputs and outputs are all two-dimensional arrays, we select some common array operations as our low-level transformation $\mathcal{T}$. The ten low-level operations we used are shown in Table 2,

Transformation $\mathcal{T}_{\text{code}}$	**Example of Input-Output Grids**	**Visualization**
TRANSPOSE	3,0,0\n0,1,2 → 3,0\n0,1\n0,2	
FLIP (horizontal, vertical)	3,0,0\n0,1,2 → 0,0,3\n2,1,0	
SHIFT (left, right, up, down)	3,0,0\n0,1,2 → 0,0,0\n1,2,0	
STRIP (around)	0,0,0\n0,3,0\n0,0,0 → 3	
EXPAND (around)	2 → 0,0,0\n0,2,0\n0,0,0	
DELETE (horizontal, vertical)	3,0\n0,1\n0,2 → 3,0\n0,2	
INSERT (horizontal, vertical)	3,0\n0,1 → 3,0,0\n0,0,1	
DUPLICATE (horizontal, vertical)	3\n1 → 3,3\n1,1	
CUT (left, right, up, down)	1,0\n0,3 → 0\n3	
MASK (diagonally)	3,3,1\n0,1,2\n2,1,0 → 0,3,0\n0,0,2\n0,1,0	

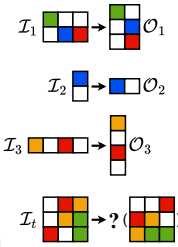
A Synthesized Textual Sample for LLMs with TRANSPOSE Operation		**Visualization**
(*Instruction*)		
You are a smart chatbot. Your goal is to give the output for the last input.		
input:\n3,0,0\n0,1,2\n ($\mathcal{I}_1$)	output:\n3,0\n0,1\n0,2\n ($\mathcal{O}_1$)	
input:\n1\n0\n ($\mathcal{I}_2$)	output:\n1,0\n ($\mathcal{O}_2$)	
input:\n7,0,2,0\n ($\mathcal{I}_3$)	output:\n7\n0\n2\n0\n ($\mathcal{O}_3$)	
input:\n0,2,7\n0,7,3\n2,0,3\n ($\mathcal{I}_t$)	output:\n	
(*Ground Truth*)		
0,0,2\n2,7,0\n7,3,3\n ($\mathcal{O}_t$)		

Table 2: Top: Ten low-level basic transformations, examples of two-dimensional input-output grids, and visualizations. Bottom: A synthesized sample which simulates abstract reasoning task with underlying "transpose" operation. The grids in green are sampled, and the grids in blue are automatically generated through transformation code $\mathcal{T}_{\text{code}}$.

including "TRANSPOSE", "FLIP", "SHIFT", "STRIP", "EXPAND", "DELETE", "INSERT", "DUPLICATE", "CUT", and "MASK". All the basic operations can all be implemented via corresponding simple code $\mathcal{T}_{\text{code}}$ (e.g., few lines of python function). Based on code implementations, we can first generate uniformly distributed inputs $\{\mathcal{I}_i\}_{i=1}^n$ and $\mathcal{I}_t$ with two-dimensional arrays composed of integers from 0 to 9, and then obtain the corresponding transformed outputs $\{\mathcal{O}_i\}_{i=1}^n = \{\mathcal{T}_{\text{code}}(\mathcal{I}_i)\}_{i=1}^n$ and $\mathcal{O}_t = \mathcal{T}_{\text{code}}(\mathcal{O}_t)$ through $\mathcal{T}_{\text{code}}$ automatically. These input-output grids can be further formalized as instruction tuning data with instruction and ground truth response. Note that we avoid designing complex task-specific foundational low-level operations. We hope that these meta operations can enhance the model's more generalized abstract reasoning abilities.

2.3 High-level DSL-based Synthesis

Besides basic operations, the ARC dataset also requires higher-level prior knowledge for abstract reasoning, such as object priors, goal-directedness priors, numbers and counting priors, geometry and topology priors, which imposes higher challenges on constructing both input grids ($\{\mathcal{I}_i\}_{i=1}^n, \mathcal{I}_t$) and underlying operations ($\mathcal{T}$). However, it is difficult to accomplish through codes with simple array operations. Therefore, we consider performing data synthesis based on higher-level domain specific language (DSL) for the task.

We apply the released DSL data by Hodel (2024) for the 400 training task from ARC dataset. By calling pre-defined primitives, the DSL generator simulates each task, including the input grids and the corresponding operation. By using DSL generator, we can synthesize samples that better match the distribution of the training set from ARC dataset. In Figure 1, we show an example of using DSL generator to synthesize samples following task d037b0a7 from limited training set of ARC. We hope that the combination of high-level and low-level data can help LLMs solve both challenging and simplified abstract reasoning problems, thereby facilitating stronger generalization ability.

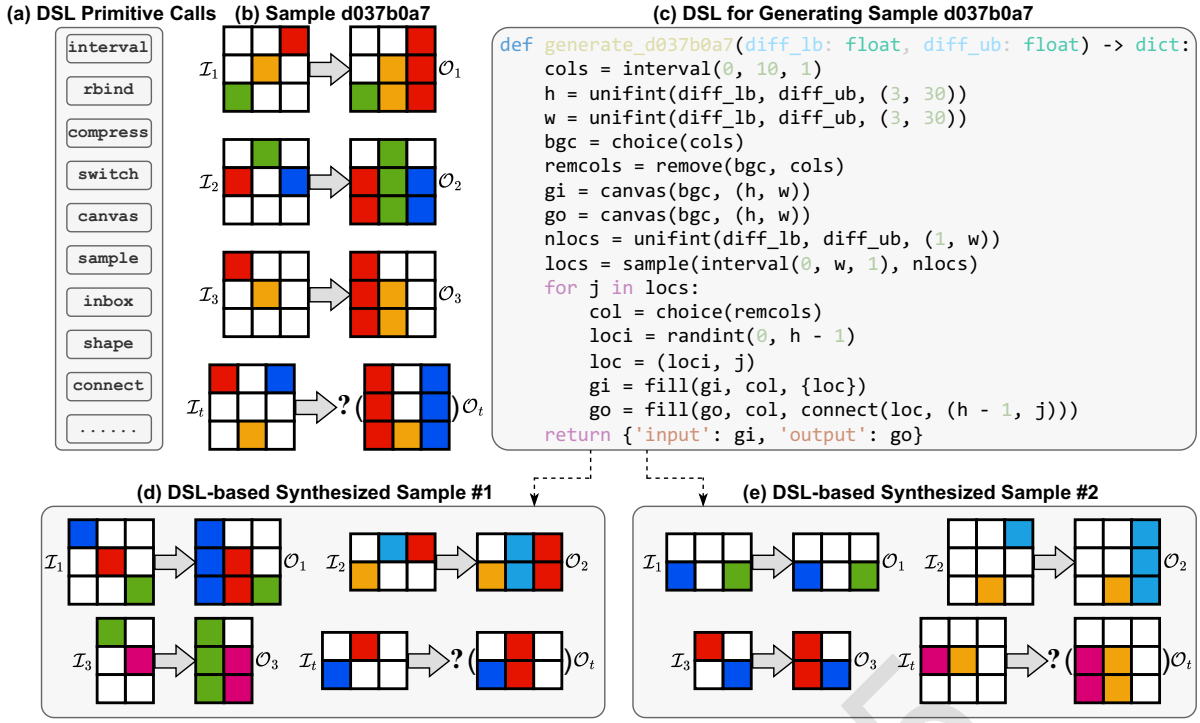


Figure 1: An example illustrating how to synthesize two samples (d, e) based on a specific sample (b) using DSL (domain specific language) generator (c), where the DSL generator involves using serval DSL primitive calls (a).

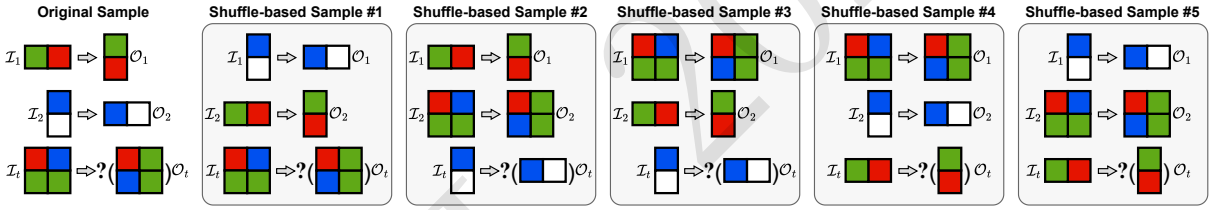


Figure 2: An example illustrating the original sample and five shuffle-based augmented samples.

2.4 Shuffle-based Synthesis

The last strategy is rather intuitive. Given a synthetic sample with $\{\mathcal{I}_i\}_{i=1}^n$, $\{\mathcal{O}_i\}_{i=1}^n$, $\mathcal{I}_t$, and $\mathcal{O}_t$, considering that the n input-output pairs $\{\mathcal{I}_i, \mathcal{O}_i\}_{i=1}^n$ and the test pair $\{\mathcal{I}_t, \mathcal{O}_t\}$ follow the same operation $\mathcal{T}$ and they are order-agnostic, we can shuffle the order of each pair to construct a total of $\frac{n(n+1)}{2}$ valid samples that require abstract reasoning ability with the same operation $\mathcal{T}$. As an example shown in Figure 2, for a task with 2 input-output pairs and 1 test pair, we can expand it to six samples with the same "TRANSPOSE" operation. In practice, we randomly select 3∼5 shuffle-based augmented samples out of all combinations.

2.5 Overall Comparison with Previous Work on Abstract Reasoning

In Table 3, we compare with previous work on abstract reasoning from different perspectives. In particular, we apply our model across all four abstract reasoning tasks and require no external prompting or test-time costs (e.g., training or calling large language models).

3 Experiments

3.1 Settings

Supervised Fine-tuning. Based on our three data synthesis strategies, we perform supervised fine-tuning (SFT) on open-source models using the collected 300k instruction tuning data, including 50k low-level

Related Work	Evaluation Datasets				External Requirement	
	ARC	Concept-ARC	1D-ARC	Mini-ARC	Prompting	Test-time Costs
Concept-ARC (Moskvichev et al., 2023)	-	full	-	-	no	no
Object-based Reasoning (Xu et al., 2024)	-	-	subsets	-	yes	no
Symbol2Language (Wang et al., 2025)	-	-	subsets	-	yes	no
ReARC (Hodel, 2024)	full	-	-	-	no	no
Hypothesis Search (Qiu et al., 2024)	-	-	-	full	no	yes
Test-Time Training (Akyürek et al., 2024)	full	-	-	-	no	yes
Ours	**full**	**full**	**full**	**full**	**no**	**no**

Table 3: Overall comparison with previous work on ARC-like problems.

Method	#Problems Solved	Method	#Problems Solved
(*Reported by Mirchandani et al. (2023)*)		(*Our results, pass@1*)	
gpt-4-0613	77	gpt-3.5-turbo-0125	42
text-davinci-003 (Ouyang et al., 2022a)	85	gpt-4-0613	113
text-davinci-002 (Ouyang et al., 2022a)	64	gpt-4o-mini	42
PaLM (Chowdhery et al., 2023)	42	LLaMA-3-8B	10
Ainooson et al. (2023)	130	LLaMA-3-8B w/ CoT	9
Kaggle 1st Place (2020)†	164	Qwen-2.5-7B	38
Xu et al. (2023)‡	57	Qwen-2.5-7B w/ CoT	31
Ferré (2021)	32	**Ours**	**185**
Human Performance*	500∼600		

Table 4: The problems solved (out of 800) on ARC dataset. Reported results on left and our results on right. † among 3 candidates. ‡ out of selected subsets. *: 64.2%∼76.2% accuracy in LeGris et al. (2024).

code-based data, 50k high-level DSL-based data, and 200k shuffle-based data. All instruction tuning data is synthetic and does not include any evaluation samples. We follow LLaMA-Factory (Zheng et al., 2024) and apply LoRA (Hu et al., 2022) fine-tuning on LLaMA-3-8B (Dubey et al., 2024) and Qwen-2.5-7B (Yang et al., 2024) model using 4 NVIDIA 6000 Ada GPUs, with learning rate 1e-4, batch size 32, and epochs 5 (when the performance becomes saturated). We choose the checkpoint based on Qwen-2.5-7B as our final model considering its better performance.

Datasets. We evaluate on the original ARC dataset (Chollet, 2019) as well as three modified versions of ARC, including Mini-ARC (Kim et al., 2022), Concept-ARC (Moskvichev et al., 2023), and 1D-ARC (Xu et al., 2024).

Baselines. We compare with widely-used LLMs, including closed-source GPT-3.5-turbo (OpenAI, 2022), GPT-4 (OpenAI, 2023), and GPT-4o-mini (OpenAI, 2024) models, as well as open-source models LLaMA-3-8B (Dubey et al., 2024) and Qwen-2.5-7B (Yang et al., 2024). We use the same instruction without any prompt engineering except for zero-shot chain-of-thought (CoT) baseline with "Let's think step by step" prompts (Kojima et al., 2022) to elicit reasoning ability. In addition, we also compare to few existing reported results such as self-refine (Madaan et al., 2023) and hypothesis search (Qiu et al., 2024) on each dataset mentioned above. The decoding temperature is set as 0 and we evaluate using pass@1 accuracy based on single response.

3.2 Main Results

ARC Dataset. The results on ARC dataset are shown in Table 4. The study by Mirchandani et al. (2023) mainly show that LLMs can extract general patterns from limited observations. For instance, text-davinci-003 solves 85 problems, surpassing or approaching some specific symbolic methods by Xu et al. (2023) and Ainooson et al. (2023).

Among the results we obtained, **gpt-4-0613 performs the best** (solves 113 problems), surpassing gpt-3.5-turbo-0125 and showing a stronger textual reasoning capability. Additionally, it outperformed

Concept	GPT-4 (τ=0)†	GPT-4 (τ=0.5)†	gpt-3.5-turbo-0125	gpt-4-0613	gpt-4o-mini	LLaMA-3-8B (w/ CoT)	Qwen-2.5-7B (w/ CoT)	Ours	Human‡
Above and Below	0.23	0.37	0.10	0.30	0.10	0.00 (0.03)	0.03 (0.00)	0.27	0.90
Center	0.33	0.33	0.03	0.33	0.17	0.03 (0.07)	0.13 (0.10)	0.17	0.94
Clean Up	0.20	0.27	0.17	0.40	0.13	0.00 (0.00)	0.03 (0.03)	0.33	0.97
Complete Shape	0.23	0.23	0.10	0.23	0.10	0.03 (0.00)	0.07 (0.10)	0.23	0.85
Copy	0.23	0.27	0.07	0.23	0.13	0.07 (0.07)	0.03 (0.07)	0.20	0.94
Count	0.13	0.17	0.07	0.07	0.03	0.03 (0.03)	0.10 (0.07)	0.13	0.88
Extend To Boundary	0.07	0.10	0.03	0.10	0.03	0.00 (0.00)	0.03 (0.07)	0.20	0.93
Extract Objects	0.03	0.07	0.00	0.03	0.00	0.00 (0.00)	0.00 (0.00)	0.17	0.86
Filled and Not Filled	0.17	0.27	0.07	0.23	0.07	0.00 (0.07)	0.17 (0.17)	0.40	0.96
Horizontal and Vertical	0.27	0.33	0.03	0.17	0.10	0.07 (0.03)	0.07 (0.07)	0.23	0.91
Inside and Outside	0.10	0.16	0.03	0.13	0.03	0.03 (0.00)	0.07 (0.07)	0.27	0.91
Move To Boundary	0.20	0.20	0.00	0.17	0.03	0.00 (0.00)	0.00 (0.00)	0.17	0.91
Order	0.27	0.27	0.27	0.27	0.17	0.07 (0.03)	0.17 (0.13)	0.20	0.83
Same and Different	0.17	0.27	0.03	0.17	0.17	0.10 (0.07)	0.03 (0.30)	0.30	0.88
Top and Bottom 2D	0.23	0.37	0.10	0.40	0.13	0.00 (0.00)	0.13 (0.03)	0.43	0.95
Top and Bottom 3D	0.20	0.27	0.07	0.13	0.13	0.03 (0.00)	0.03 (0.03)	0.40	0.93
All Concepts	0.19	<u>0.24</u>	0.07	0.22	0.10	0.03 (0.03)	0.09 (0.08)	**0.26**	0.91

Table 5: Accuracy on Concept-ARC with 16 concepts (30 problems each). † Reported by Moskvichev et al. (2023). ‡: Reported by Mitchell et al. (2023).

Task	GPT-3.5†	GPT-4†	gpt-3.5-turbo-0125	gpt-4-0613	gpt-4o-mini	LLaMA-3-8B (w/ CoT)	Qwen-2.5-7B (w/ CoT)	Ours
1D Move 1p	0.20	0.66	0.40	0.70	0.46	0.36 (0.26)	0.28 (0.34)	0.68
1D Move 2p	0.06	0.26	0.12	0.40	0.16	0.06 (0.06)	0.14 (0.10)	0.26
1D Move 3p	0.14	0.24	0.10	0.24	0.08	0.14 (0.06)	0.16 (0.10)	0.12
1D Move Dynamic	0.12	0.22	0.16	0.22	0.08	0.08 (0.12)	0.10 (0.06)	0.08
1D Move 2p Towards	0.06	0.34	0.26	0.48	0.24	0.06 (0.10)	0.12 (0.12)	0.42
1D Fill	0.12	0.66	0.18	0.80	0.30	0.22 (0.22)	0.42 (0.48)	0.90
1D Padded Fill	0.06	0.26	0.06	0.68	0.14	0.08 (0.04)	0.16 (0.14)	0.72
1D Hollow	0.04	0.56	0.20	0.64	0.12	0.14 (0.18)	0.26 (0.30)	0.74
1D Flip	0.22	0.70	0.48	0.68	0.48	0.44 (0.22)	0.42 (0.32)	0.80
1D Mirror	0.08	0.20	0.08	0.10	0.02	0.04 (0.02)	0.06 (0.06)	0.26
1D Denoise	0.22	0.36	0.14	0.64	0.14	0.10 (0.02)	0.16 (0.22)	0.98
1D Denoise Multicolor	0.26	0.60	0.52	0.90	0.38	0.28 (0.08)	0.24 (0.28)	0.92
1D Pattern Copy	0.22	0.36	0.22	0.62	0.36	0.06 (0.02)	0.46 (0.38)	0.80
1D Pattern Copy Multicolor	0.32	0.38	0.24	0.54	0.32	0.24 (0.08)	0.42 (0.36)	0.80
1D Recolor by Odd Even	0.26	0.32	0.24	0.28	0.24	0.18 (0.10)	0.28 (0.22)	0.24
1D Recolor by Size	0.04	0.28	0.06	0.28	0.04	0.04 (0.04)	0.10 (0.10)	0.24
1D Recolor by Size Comparison	0.12	0.20	0.10	0.22	0.12	0.08 (0.02)	0.10 (0.16)	0.20
1D Scaling	0.28	0.88	0.38	0.88	0.30	0.52 (0.32)	0.44 (0.48)	0.78
All Tasks	0.16	0.42	0.22	<u>0.52</u>	0.22	0.17 (0.11)	0.24 (0.23)	**0.55**

Table 6: Accuracy on 1D-ARC with 18 sub-tasks (50 problems each). † Reported by Xu et al. (2024).

gpt-4o-mini, indicating that multimodal ability does not significantly aid in abstract reasoning, which aligns with the findings of Mitchell et al. (2023). **The performance of 7/8B open-source models is very poor**, only solve less than 40 problems, showing the incapability of abstract reasoning. Furthermore, we find that using **CoT reasoning does not improves the results**, which aligns with recent studies indicating that CoT is not a broadly effective method, where its effectiveness mainly lies in math and symbolic problems (Sprague et al., 2024). As comparison, after supervised fine-tuning Qwen-2.5-7B model on program-based synthesized instruction data, **our 7B size model outperforms various closed-source and open-source LLMs**, solving 185 problems without using any prompting techniques. Overall, the results demonstrates the effectiveness of our program-based data synthesis, which can improve the abstract reasoning ability of LLMs by large margin.

Concept-ARC. The results on Concept-ARC dataset are shown in Table 5. Similarly, **our model achieves the best performance** (26% overall accuracy) compared with GPT-4 (24% accuracy) and open-source models (3%∼9% accuracy). We find that **our model performs well in recognizing concepts like "top&bottom" and "filled&not filled"**, with accuracy exceeding 40%. In contrast, the model **performs poorly in counting** related problems, with only 13% accuracy, which also reflects the current limitations of LLMs in some basic arithmetic capabilities (Yehudai et al., 2024).

1D-ARC. The results on 1D-ARC dataset are shown in Table 6. Different from standard ARC dataset, all input-output pairs of 1D-ARC are composed of grids with only a single row. We find that **our model still show a large improvement compared to baselines** including LLaMA-3 and Qwen-2.5, even surpassing gpt-4-0613 model. Moreover, **our model achieve 98% accuracy for sub-problems with "denoise" operations**, while we do not explicit construct data with similar operations. These results indicate our

Proceedings of the 24th China National Conference on Computational Linguistics, pages 904-921, Jinan, China, August 11-14, 2025.

Method	Accuracy
gpt-3.5-turbo-0125	0.107
gpt-4-0613	**0.241**
gpt-4o-mini	0.100
LLaMA-3-8B	0.047
LLaMA-3-8B w/ CoT	0.047
Qwen-2.5-7B	0.094
Qwen-2.5-7B w/ CoT	0.101
gpt-4-0613 w/ Self-refine (Madaan et al., 2023)†	0.151
gpt-4-0613 w/ Hypothesis Search (Qiu et al., 2024)†	0.187
Ours	0.201

Table 7: Accuracy on 150 problems in Mini-ARC. † Reported by Qiu et al. (2024).

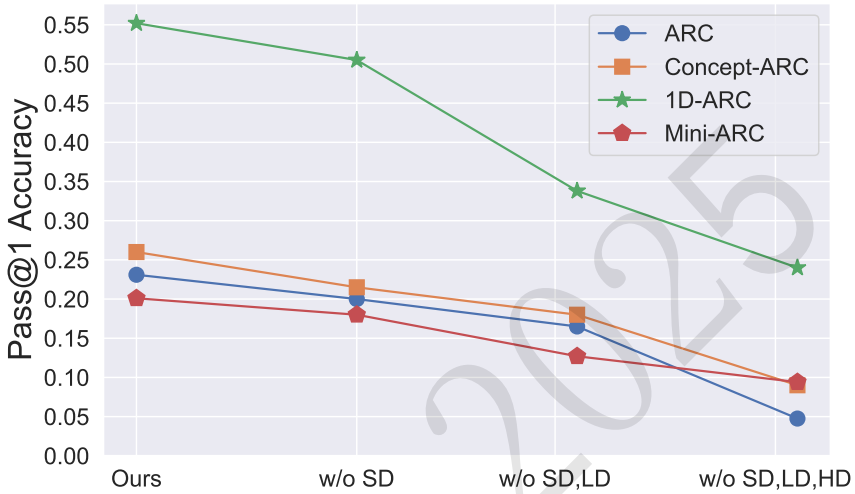


Figure 3: Ablation study on different types of synthesized data. SD: shuffle-based data. LD: low-level code-based data. HD: high-level DSL-based data.

program-based synthesized 2D data indeed helps enhancing abstract reasoning ability, **not simply by memorizing more samples, but by having a certain level of generalization** in solving related problems.

Mini-ARC. The results on Mini-ARC dataset are shown in Table 7. Overall, gpt-4-0613 gives the best results (24.1% accuracy), however, Qiu et al. (2024) indicate that explicit post-training methods such as self-refine (Madaan et al., 2023) and hypothesis search (Qiu et al., 2024) lead to decreased results (15.1% and 18.7% accuracy). As for the results from direct response, **our model achieves the second-best performance** (20.1% accuracy) without relying on post-training techniques, still consistently outperforming gpt-3.5-turbo-0125, gpt-4o-mini, and open-source models.

3.3 Ablation Study

We further analyze the impact of our different types of synthetic data. Specifically, we trained the model by gradually removing the part of shuffle-based data, low-level code-based data, and high-level DSL-based data, while keeping the total training steps consistent. Then we evaluate different checkpoints separately across all datasets.

The results are shown in Figure 3. We find that the performance on various datasets experiences a slight decrease without shuffle-based data. The code-based data has the largest impact on 1D-ARC and consistently aids on other datasets. As for the DSL-based data, since it is originally designed based on the samples from ARC, it closely resembles ARC in terms of data distribution and lead to larger improvement. However, if we only use the DSL data as adopted by Hodel (Hodel, 2024), the performance on Mini-ARC

HD: LD	ARC	Concept-ARC	1D-ARC	Mini-ARC	Avg.
50%: 50%	0.23	**0.26**	0.55	**0.20**	**0.31**
70%: 30%	**0.24**	**0.26**	0.34	0.12	0.24
30%: 70%	0.13	0.16	**0.58**	**0.22**	0.27

Table 8: Results with different ratio of mixture data.

Prompt and Generated Pairs by GPT-4	**Re-organized GPT-Generated Data for Abstract Reasoning**
(*Instruction & Limited Pairs from ARC Dataset*) Generate more input and output following the same operation. last input. input: $\mathcal{I}_1$ output: $\mathcal{O}_1$, input: $\mathcal{I}_2$ output: $\mathcal{O}_2$, input: $\mathcal{I}_3$ output: $\mathcal{O}_3$	(*Instruction & GPT-Generated Input-Output Grids*) You are a smart chatbot. Your goal is to give the output for the last input. input: $\mathcal{I}_1^{\text{gen}}$ output: $\mathcal{O}_1^{\text{gen}}$, input: $\mathcal{I}_2^{\text{gen}}$ output: $\mathcal{O}_2^{\text{gen}}$, input: $\mathcal{I}_3^{\text{gen}}$ output:
(*Response by GPT-4*) input: $\mathcal{I}_1^{\text{gen}}$ output: $\mathcal{O}_1^{\text{gen}}$, input: $\mathcal{I}_2^{\text{gen}}$ output: $\mathcal{O}_2^{\text{gen}}$, input: $\mathcal{I}_3^{\text{gen}}$ output: $\mathcal{O}_3^{\text{gen}}$, input: $\mathcal{I}_4^{\text{gen}}$ output: $\mathcal{O}_4^{\text{gen}}$, ...	(*GPT-Generated Ground Truth*) $\mathcal{O}_3^{\text{gen}}$

Table 9: Illustration of synthesizing ARC data using GPT-4. We re-organize the generated grids for constructing more data and fine-tuning open-source LLMs.

and 1D-ARC quickly saturates and will not exceed or come close to the level of gpt-4-0613.

For different ratio of high- and low-level data, results in Table 8 show that A balanced using of both types of data yields the best overall results, where high-level data only show limited improvements for 1D-ARC and Mini-ARC, and similarly, low-level data offers limited improvements for ARC and Concept-ARC. Overall, combining different types of synthetic data contribute to enhancing the general abstract reasoning ability of LLMs across different datasets, **showing the importance of mixture of low-level and high-level synthesized data**.

4 Analysis

Comparison with Synthetic Data by GPT-4. Researchers have been investigating the utilization of LLMs (e.g., GPT-4) to produce synthetic datasets directly. As for the abstract reasoning task, we also considering using LLMs to synthesize more diverse input-output grids that follow to the same operations based on given input-output grids, and then constructing synthesized instruction data based on the generated results, as illustrated in Table 9.

Similarly, we utilize the collected synthetic GPT-generated data for supervised fine-tuning and compare it with our program-based synthesized data. The GPT-generated data is used in two ways: 1) We directly use them for fine-tuning Qwen-7B with the same steps as comparison. 2) Based on our model trained on program-based synthesized data, we use them for continued training with 3000 steps.

The results are shown in Figure 4. By using GPT-generated data, the performance improvement of Qwen-2.5 is very marginal. Moreover, we find that regardless of whether the model was trained directly on Qwen-2.5 or continued training based on our model, there was a certain degree of performance gaps, with accuracy on 1D-ARC even drops by 23%. This suggests that the **data quality synthesized automatically by GPT is poor** for the challenging non-natural language based reasoning task with abstract sequence of numbers. This also indicates the validity and effectiveness of our program-based data synthesis strategies. We also show some failure cases of the GPT-generated invalid data in Appendix B.

Length of Input-Output Grids. Intuitively, longer input-output grids may indicate more complicated underlying operations, making it more challenging for LLMs. We conduct analyses on problems of varying lengths and the results are shown in Figure 5. We find that the performance indeed drops as the length increases on ARC dataset. However, for problems in concept-ARC, the impact of increasing length is not significant. In contrast, for problems on 1D-ARC and Mini-ARC, the model performs better

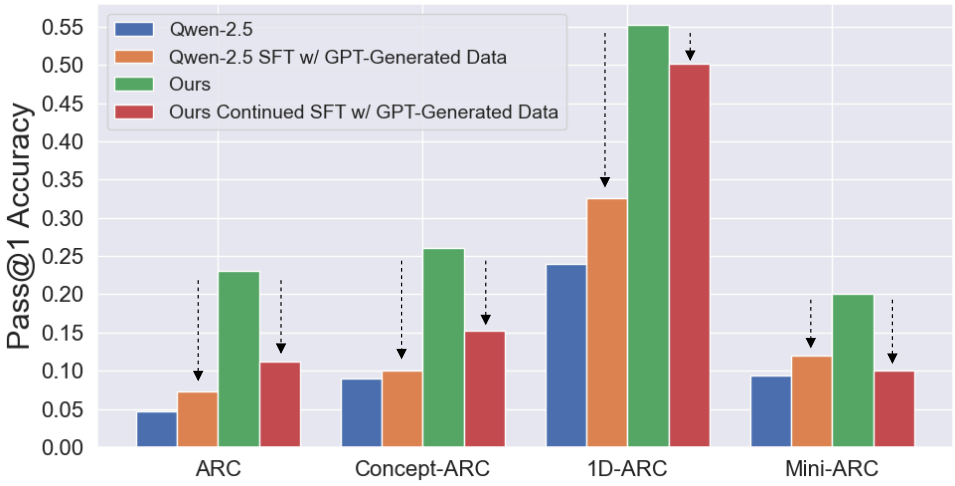


Figure 4: Results with GPT-generated synthetic data.

for longer problems. This suggests that the difficulty of abstract reasoning tasks is not fully related to the length of the problems. It is more crucial to solve abstract reasoning tasks **from transformation understanding and generation perspectives rather than dealing with long texts** for existing LLMs.

Error Analysis. Analyzing errors in ARC dataset directly is quite challenging because each sample presents unique and highly challenging operation. Therefore, most errors stem from the lack of understanding for these transformations. To conduct a more intuitive and fine-grained analysis, we select problems with "Move-1p", "Move-2p", and "Move-3p" operations from 1D-ARC. These operations are generally considered straightforward for humans which moves the object in grids forward for few pixels. However, the model's accuracy on these 150 problems is relatively low (12%∼68%), as shown in the first three rows of Table 6.

The results are shown in Figure 6. Qwen-2.5 only solves 19.3% problems, with the majority of errors attributed to incorrect steps, inconsistent object, and incorrect repetition. Specifically, the incorrect repetition results accounted for 25.3%, where the response keeps the same with the test inputs, indicating a failure to recognizing the "move" operation from given grids. In our model, the accuracy increases to 35.3%, though incorrect steps and inconsistent object errors remains high, which **indicates some inherent LLMs' issues of counting and planning ability due to the nature of auto-regressive language model** (Bubeck et al., 2023). We note that incorrect repetition issue decreases to 7.3%, suggesting an improvement in the model's ability to recognize underlying transformation and take operations for test inputs accordingly.

Case Study. In appendix A, we show responses by gpt-4-0613 and our model for different datasets in Tables 10∼13, respectively.

5 Related Work

Evaluating Abstract Reasoning Ability of LLMs. Recent studies test the abstract reasoning ability of LLMs (LeGris et al., 2024; Lee et al., 2024) and compare the results with human performance. Mirchandani et al. (2023) shows that GPT-3 can recognize patterns and solve problems on ARC dataset. Mitchell et al. (2023) reveals that GPT-4 and GPT-4V only achieve 65% and 25% accuracy for 48 select sub-tasks, lower than the 95% accuracy by humans. Xu et al. (2024) and Moskvichev et al. (2023) evaluate GPT-4 and LLaMA-2 models on proposed 1D-ARC and Concept-ARC datasets, respectively. Overall, there is room for further enhancement in performance and generalization across different types of problems.

Improving Abstract Reasoning Ability of LLMs. Xu et al. (2024) proposes "objective-based" representation to describe the input-output grids. Wang et al. (2025) uses "symbol-to-language" prompting to integrate textual representations. Huang et al. (2023) and Qiu et al. (2024) propose generating codes for underlying operations during reasoning. Akyürek et al. (2024) investigate test-time training for abstract

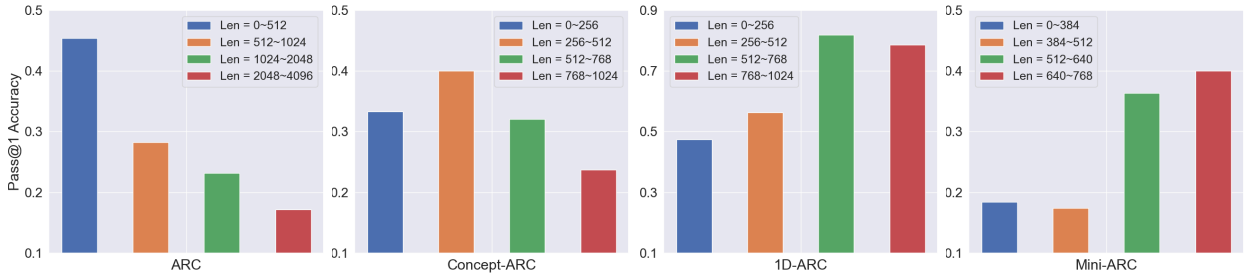


Figure 5: The fine-grained results for problems with different length of input-output grids after tokenization operationfor different datasets.

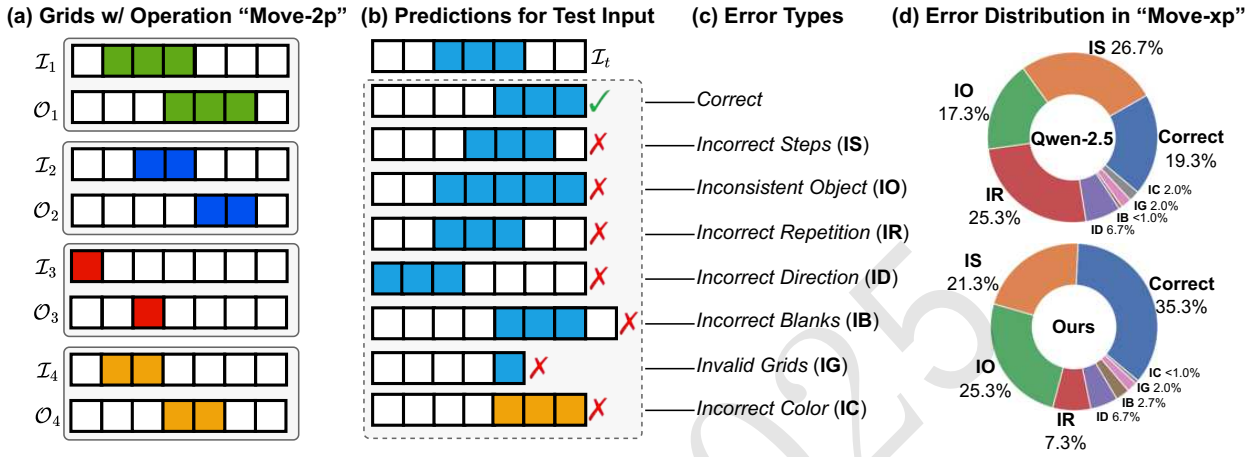


Figure 6: Error analysis for samples with "move 1/2/3 pixels" operations. (a) A "move 2 pixels" example. (b) Test input and possible model predictions. (c) Different error types. (d) Error distribution for Qwen-2.5 and our model.

reasoning. These efforts improves the performance of LLMs to some extent. However, they either rely on prompt design or increase the inference cost, and are limited in a single dataset or few selected subsets. Further, our data synthesis method can combine with these post-training approaches in a straightforward manner, i.e., leveraging these methods on models trained with the mixture synthesized data.

Data Synthesis in the Era of LLMs. LLMs-driven data synthesis and augmentation have gained much attention (Long et al., 2024; Liu et al., 2024; Wang et al., 2024). Data synthesis can help improving various abilities of LLMs, including instruction-following (Wei et al., 2022a), reasoning (Wei et al., 2022b), coding (Roziere et al., 2023), and human-preference aligning (Ouyang et al., 2022b), etc. Given sufficient training data resources, LLMs can synthesize high-quality data of these common tasks. However, for abstract reasoning, the complexity of tasks result in the scarcity of training data and poor performance of current models. Therefore, we still need more reliable data synthesis methods beyond using LLMs.

6 Conclusion and Future Work

We propose different program-based data synthesis strategies to construct data for abstract reasoning task, including low-level code-based synthesis, high-level DSL-based synthesis, and shuffle-based synthesis. These strategies allows us to generate a large amount of valid data at a low cost, serving as useful instruction tuning data for LLMs. Experimental results show that it consistently improves the abstract reasoning ability of 7B size open-source LLMs across four datasets, surpassing various widely-used LLMs such as GPT-3.5, GPT-4, GPT-4o, LLaMA-3, and Qwen-2.5 without using any post-training techniques. For future work, we plan to explore integrating test-time scaling methods to further enhance the performance of smaller open-source models on the ARC dataset.

Proceedings of the 24th China National Conference on Computational Linguistics, pages 904-921, Jinan, China, August 11-14, 2025.

Acknowledgements

This work is supported in parts by NSFC (62306161, U21B2023), DEGP Innovation Team (2022KC XTD025), Shenzhen Science and Technology Program (KJZD20240903100022028, KQTD20210811 090044003, RCJC20200714114435012), and Scientific Development Funds from Shenzhen University.

References

James Ainooson, Deepayan Sanyal, Joel P Michelson, Yuan Yang, and Maithilee Kunda. 2023. A neurodiversity-inspired solver for the abstraction\& reasoning corpus (arc) using visual imagery and program synthesis. *arXiv preprint arXiv:2302.09425*.

Ekin Akyürek, Mehul Damani, Linlu Qiu, Han Guo, Yoon Kim, and Jacob Andreas. 2024. The surprising effectiveness of test-time training for abstract reasoning. *arXiv preprint arXiv:2411.07279*.

Sébastien Bubeck, Varun Chandrasekaran, Ronen Eldan, Johannes Gehrke, Eric Horvitz, Ece Kamar, Peter Lee, Yin Tat Lee, Yuanzhi Li, Scott Lundberg, et al. 2023. Sparks of artificial general intelligence: Early experiments with GPT-4. *arXiv preprint arXiv:2303.12712*.

Francois Chollet, Mike Knoop, Gregory Kamradt, and Bryan Landers. 2024. ARC prize 2024: Technical report. *arXiv preprint arXiv:2412.04604*.

François Chollet. 2019. On the measure of intelligence. *arXiv preprint arXiv:1911.01547*.

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. 2023. Palm: Scaling language modeling with pathways. *Journal of Machine Learning Research*, 24(240):1–113.

Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, et al. 2021. Training verifiers to solve math word problems. *arXiv preprint arXiv:2110.14168*.

Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Amy Yang, Angela Fan, et al. 2024. The llama 3 herd of models. *arXiv preprint arXiv:2407.21783*.

Sébastien Ferré. 2021. First steps of an approach to the arc challenge based on descriptive grid models and the minimum description length principle.

Michael Hodel. 2024. Addressing the abstraction and reasoning corpus via procedural example generation. *arXiv preprint arXiv:2404.07353*.

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

Di Huang, Ziyuan Nan, Xing Hu, Pengwei Jin, Shaohui Peng, Yuanbo Wen, Rui Zhang, Zidong Du, Qi Guo, Yewen Pu, and Yunji Chen. 2023. Anpl: Towards natural programming with interactive decomposition. In A. Oh, T. Naumann, A. Globerson, K. Saenko, M. Hardt, and S. Levine, editors, *Advances in Neural Information Processing Systems*, volume 36, pages 69404–69440. Curran Associates, Inc.

Kaggle 1st Place. 2020. Abstraction and rasoning challenge 1st place solution.

Subin Kim, Prin Phunyaphibarn, Donghyun Ahn, and Sundong Kim. 2022. Playgrounds for abstraction and reasoning. In *NeurIPS 2022 Workshop on Neuro Causal and Symbolic AI (nCSI)*.

Takeshi Kojima, Shixiang (Shane) Gu, Machel Reid, Yutaka Matsuo, and Yusuke Iwasawa. 2022. Large language models are zero-shot reasoners. In S. Koyejo, S. Mohamed, A. Agarwal, D. Belgrave, K. Cho, and A. Oh, editors, *Advances in Neural Information Processing Systems*, volume 35, pages 22199–22213. Curran Associates, Inc.

Seungpil Lee, Woochang Sim, Donghyeon Shin, Wongyu Seo, Jiwon Park, Seokki Lee, Sanha Hwang, Sejin Kim, and Sundong Kim. 2024. Reasoning abilities of large language models: In-depth analysis on the abstraction and reasoning corpus. *arXiv preprint arXiv:2403.11793*.

Solim LeGris, Wai Keen Vong, Brenden M Lake, and Todd M Gureckis. 2024. H-ARC: A robust estimate of human performance on the abstraction and reasoning corpus benchmark. *arXiv preprint arXiv:2409.01374*.

Wen-Ding Li, Keya Hu, Carter Larsen, Yuqing Wu, Simon Alford, Caleb Woo, Spencer M Dunn, Hao Tang, Michelangelo Naim, Dat Nguyen, et al. 2024. Combining induction and transduction for abstract reasoning. *arXiv preprint arXiv:2411.02272*.

Ruibo Liu, Jerry Wei, Fangyu Liu, Chenglei Si, Yanzhe Zhang, Jinmeng Rao, Steven Zheng, Daiyi Peng, Diyi Yang, Denny Zhou, and Andrew M. Dai. 2024. Best practices and lessons learned on synthetic data. In *First Conference on Language Modeling*.

Lin Long, Rui Wang, Ruixuan Xiao, Junbo Zhao, Xiao Ding, Gang Chen, and Haobo Wang. 2024. On LLMs-driven synthetic data generation, curation, and evaluation: A survey. In Lun-Wei Ku, Andre Martins, and Vivek Srikumar, editors, *Findings of the Association for Computational Linguistics: ACL 2024*, pages 11065–11082, Bangkok, Thailand, August. Association for Computational Linguistics.

Aman Madaan, Niket Tandon, Prakhar Gupta, Skyler Hallinan, Luyu Gao, Sarah Wiegreffe, Uri Alon, Nouha Dziri, Shrimai Prabhumoye, Yiming Yang, Shashank Gupta, Bodhisattwa Prasad Majumder, Katherine Hermann, Sean Welleck, Amir Yazdanbakhsh, and Peter Clark. 2023. Self-refine: Iterative refinement with self-feedback. In *Thirty-seventh Conference on Neural Information Processing Systems*.

Suvir Mirchandani, Fei Xia, Pete Florence, brian ichter, Danny Driess, Montserrat Gonzalez Arenas, Kanishka Rao, Dorsa Sadigh, and Andy Zeng. 2023. Large language models as general pattern machines. In *7th Annual Conference on Robot Learning*.

Melanie Mitchell, Alessandro B. Palmarini, and Arsenii Kirillovich Moskvichev. 2023. Comparing humans, GPT-4, and GPT-4v on abstraction and reasoning tasks. In *AAAI 2024 Workshop on "Are Large Language Models Simply Causal Parrots?"*.

Arsenii Kirillovich Moskvichev, Victor Vikram Odouard, and Melanie Mitchell. 2023. The conceptARC benchmark: Evaluating understanding and generalization in the ARC domain. *Transactions on Machine Learning Research*.

OpenAI. 2022. ChatGPT.

OpenAI. 2023. GPT-4 technical report. *arXiv preprint arXiv:2303.08774*.

OpenAI. 2024. GPT-4o system card.

Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Gray, John Schulman, Jacob Hilton, Fraser Kelton, Luke Miller, Maddie Simens, Amanda Askell, Peter Welinder, Paul Christiano, Jan Leike, and Ryan Lowe. 2022a. Training language models to follow instructions with human feedback. In Alice H. Oh, Alekh Agarwal, Danielle Belgrave, and Kyunghyun Cho, editors, *Advances in Neural Information Processing Systems*.

Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll 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. 2022b. Training language models to follow instructions with human feedback. In S. Koyejo, S. Mohamed, A. Agarwal, D. Belgrave, K. Cho, and A. Oh, editors, *Advances in Neural Information Processing Systems*, volume 35, pages 27730–27744. Curran Associates, Inc.

Linlu Qiu, Liwei Jiang, Ximing Lu, Melanie Sclar, Valentina Pyatkin, Chandra Bhagavatula, Bailin Wang, Yoon Kim, Yejin Choi, Nouha Dziri, and Xiang Ren. 2024. Phenomenal yet puzzling: Testing inductive reasoning capabilities of language models with hypothesis refinement. In *The Twelfth International Conference on Learning Representations*.

Baptiste Roziere, Jonas Gehring, Fabian Gloeckle, Sten Sootla, Itai Gat, Xiaoqing Ellen Tan, Yossi Adi, Jingyu Liu, Romain Sauvestre, Tal Remez, et al. 2023. Code llama: Open foundation models for code. *arXiv preprint arXiv:2308.12950*.

Zayne Sprague, Fangcong Yin, Juan Diego Rodriguez, Dongwei Jiang, Manya Wadhwa, Prasann Singhal, Xinyu Zhao, Xi Ye, Kyle Mahowald, and Greg Durrett. 2024. To cot or not to cot? chain-of-thought helps mainly on math and symbolic reasoning. *arXiv preprint arXiv:2409.12183*.

Alon Talmor, Jonathan Herzig, Nicholas Lourie, and Jonathan Berant. 2019. CommonsenseQA: A question answering challenge targeting commonsense knowledge. In Jill Burstein, Christy Doran, and Thamar Solorio, editors, *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4149–4158, Minneapolis, Minnesota, June. Association for Computational Linguistics.

Ke Wang, Jiahui Zhu, Minjie Ren, Zeming Liu, Shiwei Li, Zongye Zhang, Chenkai Zhang, Xiaoyu Wu, Qiqi Zhan, Qingjie Liu, et al. 2024. A survey on data synthesis and augmentation for large language models. *arXiv preprint arXiv:2410.12896*.

Yile Wang, Sijie Cheng, Zixin Sun, Peng Li, and Yang Liu. 2025. Leveraging language-based representations for better solving symbol-related problems with large language models. In Owen Rambow, Leo Wanner, Marianna Apidianaki, Hend Al-Khalifa, Barbara Di Eugenio, and Steven Schockaert, editors, *Proceedings of the 31st International Conference on Computational Linguistics*, pages 5544–5557, Abu Dhabi, UAE, January. Association for Computational Linguistics.

Taylor Webb, Keith J Holyoak, and Hongjing Lu. 2023. Emergent analogical reasoning in large language models. *Nature Human Behaviour*, 7(9):1526–1541.

Jason Wei, Maarten Bosma, Vincent 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 *International Conference on Learning Representations*.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, brian ichter, Fei Xia, Ed Chi, Quoc V Le, and Denny Zhou. 2022b. Chain-of-thought prompting elicits reasoning in large language models. In S. Koyejo, S. Mohamed, A. Agarwal, D. Belgrave, K. Cho, and A. Oh, editors, *Advances in Neural Information Processing Systems*, volume 35, pages 24824–24837. Curran Associates, Inc.

Yudong Xu, Elias B. Khalil, and Scott Sanner. 2023. Graphs, constraints, and search for the abstraction and reasoning corpus. *Proceedings of the AAAI Conference on Artificial Intelligence*, 37(4):4115–4122, Jun.

Yudong Xu, Wenhao Li, Pashootan Vaezipoor, Scott Sanner, and Elias Boutros Khalil. 2024. LLMs and the abstraction and reasoning corpus: Successes, failures, and the importance of object-based representations. *Transactions on Machine Learning Research*.

An Yang, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chengyuan Li, Dayiheng Liu, Fei Huang, Haoran Wei, et al. 2024. Qwen2. 5 technical report. *arXiv preprint arXiv:2412.15115*.

Gilad Yehudai, Haim Kaplan, Asma Ghandeharioun, Mor Geva, and Amir Globerson. 2024. When can transformers count to n? *arXiv preprint arXiv:2407.15160*.

Yaowei Zheng, Richong Zhang, Junhao Zhang, YeYanhan YeYanhan, and Zheyan Luo. 2024. LlamaFactory: Unified efficient fine-tuning of 100+ language models. In Yixin Cao, Yang Feng, and Deyi Xiong, editors, *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 3: System Demonstrations)*, pages 400–410, Bangkok, Thailand, August. Association for Computational Linguistics.

A Case Study

We show example of responses by gpt-4-0613 and our model across different datasets in Table 10, Table 11, Table 12, and Table 13, respectively. In these cases, the response by gpt-4-0613 is inaccurate, while the answer by our model is accurate.

B Cases of GPT-Generated Samples

We also show some failure cases of GPT-generated samples in Table 14.

Prompt

You are a smart chatbot. Your goal is to give the output for the last input.
input:8,5,0\n8,5,3\n0,3,2
output:8,5,0,0,8,8\n8,5,3,3,5,5\n0,3,2,2,3,0\n
0,3,2,2,3,0\n5,5,3,3,5,8\n8,8,0,0,5,8
input:3,8,2\n3,2,2\n8,5,2
output:3,8,2,8,3,3\n3,2,2,5,2,8\n8,5,2,2,2,2\n
2,2,2,2,5,8\n8,2,5,2,2,3\n3,3,8,2,8,3
input:0,3,0\n6,6,6\n0,3,0
output:0,3,0,0,6,0\n6,6,6,3,6,3\n0,3,0,0,6,0\n
0,6,0,0,3,0\n3,6,3,6,6,6\n0,6,0,0,3,0
input:2,5,0\n2,5,1\n3,1,1
output:

Response by gpt-4-0613

2,5,0,5,2,2\n2,5,1,1,5,3\n3,1,1,1,1,2\n
2,1,1,1,5,2\n5,1,5,1,5,3\n2,1,2,5,2,2 ✗

Response by ours

2,5,0,3,2,2\n2,5,1,1,5,5\n3,1,1,1,1,0\n
0,1,1,1,1,3\n5,5,1,1,5,2\n2,2,3,0,5,2 ✓

Visualization

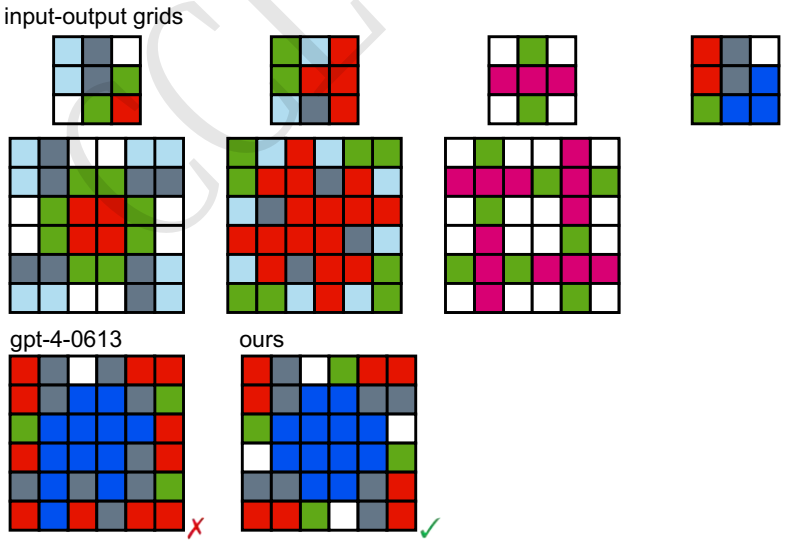


Table 10: A example showing the responses by gpt-4-0613 and our model on ARC dataset.

Prompt

You are a smart chatbot. Your goal is to give the output for the last input.

input:0,0,0,0,1,0\n0,0,0,0,1,0\n0,0,0,0,1,0\n
2,2,2,2,2,2\n0,0,0,0,1,0\n0,0,0,0,1,0
output:0,0,0,0,1,0\n0,0,0,0,1,0\n0,0,0,0,1,0\n
2,2,2,1,2\n0,0,0,0,1,0\n0,0,0,0,1,0
input:0,3,0,0,0,0\n0,3,0,0,0,0\n0,3,0,0,0,0\n
4,3,4,4,4,4\n0,3,0,0,0,0\n0,3,0,0,0,0
output:0,3,0,0,0,0\n0,3,0,0,0,0\n0,3,0,0,0,0\n
4,4,4,4,4,4\n0,3,0,0,0,0\n0,3,0,0,0,0
input:0,0,0,0,8,0\n0,0,0,0,8,0\n7,7,7,7,7,7\n
0,0,0,0,8,0\n0,0,0,0,8,0\n0,0,0,0,8,0
output:0,0,0,0,8,0\n0,0,0,0,8,0\n7,7,7,7,8,7\n
0,0,0,0,8,0\n0,0,0,0,8,0\n0,0,0,0,8,0
input:0,8,0,0,0,0\n5,8,5,5,5,5\n0,8,0,0,0,0\n
5,5,5,5,5,5\n0,8,0,0,0,0\n0,8,0,0,0,0
output:0,8,0,0,0,0\n5,5,5,5,5,5\n0,8,0,0,0,0\n
5,8,5,5,5,5\n0,8,0,0,0,0\n0,8,0,0,0,0
input:0,9,0,0,0,0\n0,9,0,0,0,0\n0,9,0,0,0,0\n
3,3,3,3,3,3\n0,9,0,0,0,0\n0,9,0,0,0,0
output:

Response by gpt-4-0613

0,9,0,0,0,0\n0,9,0,0,0,0\n0,9,0,0,0,0\n
3,3,3,3,9,3\n0,9,0,0,0,0\n0,9,0,0,0,0 ✗

Response by ours

0,9,0,0,0,0\n0,9,0,0,0,0\n0,9,0,0,0,0\n
3,9,3,3,3,3\n0,9,0,0,0,0\n0,9,0,0,0,0 ✓

Visualization

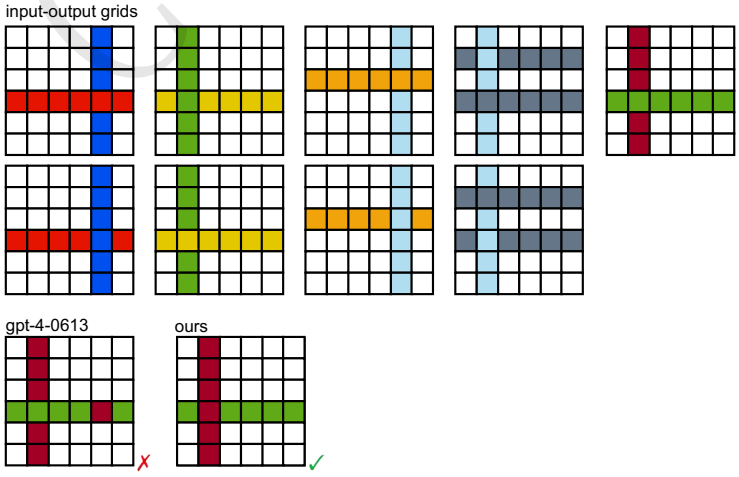


Table 11: A example showing the responses by gpt-4-0613 and our model on Concept-ARC dataset.

Prompt

You are a smart chatbot. Your goal is to give the output for the last input.
input:0,6,6,6,6,6,6,6,6,6,6,6,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
output:0,0,0,0,6,6,6,6,6,6,6,6,6,6,6,6,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
input:0,0,0,0,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,0,0,0,0,0,0,0,0,0,0,0,0
output:0,0,0,0,0,0,0,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,0,0,0,0,0,0,0,0,0,0
input:0,0,0,0,5,5,5,5,5,5,5,5,5,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
output:0,0,0,0,0,0,0,5,5,5,5,5,5,5,5,5,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
input:0,0,0,0,0,0,0,0,0,7,7,7,7,7,7,7,7,7,7,7,7,7,7,7,7,7,0,0,0,0,0,0,0
output:

Response by gpt-4-0613

0,0,0,0,0,0,0,0,0,0,0,7,7,7,7,7,7,7,7,7,7,7,7,7,7,7,0,0,0,0,0 ✗

Response by ours

0,0,0,0,0,0,0,0,0,0,0,0,7,7,7,7,7,7,7,7,7,7,7,7,7,7,7,0,0,0 ✓

Visualization

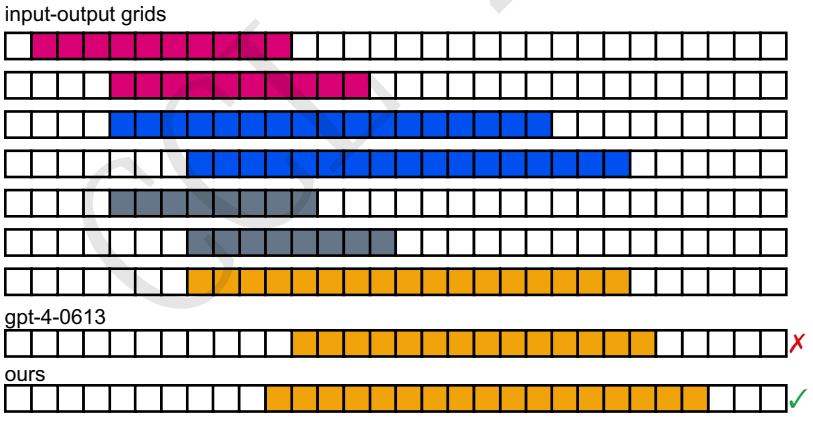


Table 12: A example showing the responses by gpt-4-0613 and our model on 1D-ARC dataset.

Prompt

You are a smart chatbot. Your goal is to give the output for the last input.
input:5,0,0,0,0\n5,0,5,0,0\n5,0,5,5,0\n5,5,5,5,0\n5,5,5,5,5
output:1,0,0,0,0\n1,0,5,0,0\n1,0,5,5,0\n1,5,5,5,0\n1,5,5,5,2
input:0,0,5,0,0\n5,0,5,0,0\n5,0,5,5,0\n5,5,5,5,0\n5,5,5,5,5
output:0,0,1,0,0\n5,0,1,0,0\n5,0,1,5,0\n5,5,1,5,0\n5,5,1,5,2
input:5,0,0,0,0\n5,5,0,0,0\n5,5,5,0,0\n5,5,5,5,0\n5,5,5,5,5
output:1,0,0,0,0\n1,5,0,0,0\n1,5,5,0,0\n1,5,5,5,0\n1,5,5,5,2
input:0,5,0,0,0\n0,5,0,0,5\n0,5,5,0,5\n5,5,5,0,5\n5,5,5,5,5
output:0,1,0,0,0\n0,1,0,0,5\n0,1,5,0,5\n5,1,5,0,5\n5,1,5,2,5
input:0,0,0,5,0\n0,0,0,5,5\n0,0,5,5,5\n5,0,5,5,5\n5,5,5,5,5
output:

Response by gpt-4-0613

0,0,0,1,0\n0,0,0,1,5\n0,0,5,1,5\n5,0,5,1,5\n5,5,5,1,2 ✗

Response by ours

0,0,0,1,0\n0,0,0,1,5\n0,0,5,1,5\n5,0,5,1,5\n5,2,5,1,5 ✓

Visualization

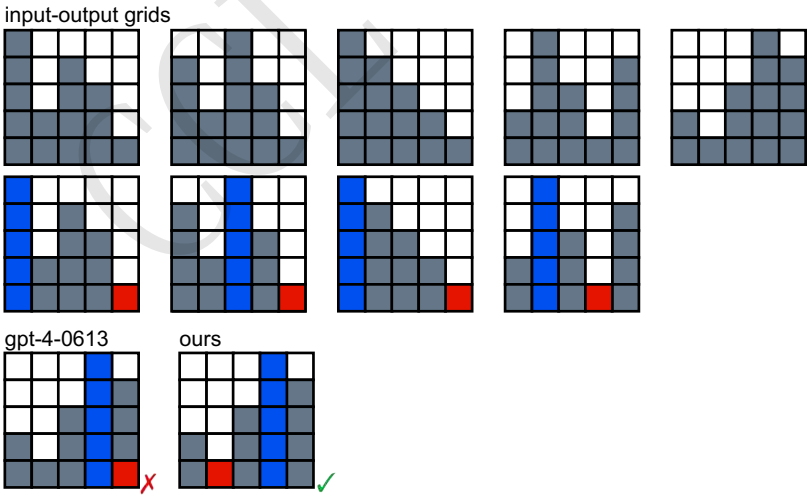


Table 13: A example showing the responses by gpt-4-0613 and our model on Mini-ARC dataset.

Prompt for GPT-4
Generate more input and output following the same operation. input: 1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,0,0,0,0,0 output: 0,0,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,0,0,0 input: 3,3,3,3,3,3,3,3,3,3,3,3,3,3,3,0,0,0,0,0,0 output: 0,0,3,3,3,3,3,3,3,3,3,3,3,3,3,3,3,0,0,0,0 input: 0,8,8,8,8,8,8,8,8,8,8,8,8,0,0,0,0,0,0,0,0 output: 0,0,0,8,8,8,8,8,8,8,8,8,8,8,8,0,0,0,0,0,0
Generated Data #1 (incorrect steps)
input: 0,0,0,0,4,4,4,4,4,4,4,4,4,4,4,4,4,0,0,0,0 output: 0,0,0,0,0,0,0,4,4,4,4,4,4,4,4,4,4,4,4,4,0
Generated Data #2 (inconsistent object)
input: 0,0,0,0,0,0,7,7,7,7,7,7,7,7,7,0,0,0,0,0,0 output: 0,0,0,0,0,0,0,0,7,7,7,7,7,7,7,7,7,0,0,0,0

Table 14: A example showing failure cases of GPT-generated samples.