Understanding LLMs' Fluid Intelligence Deficiency: An Analysis of the ARC Task

Junjie Wu[1]   **Mo Yu**[2*]   **Lemao Liu**[2]   **Dit-Yan Yeung**[1*]   **Jie Zhou**[2]

[1]Hong Kong University of Science and Technology
[2]WeChat AI, Tencent

junjie.wu@connect.ust.hk   moyumyu@global.tencent.com
{redmondliu, withtomzhou}@tencent.com   dyyeung@ust.hk

Abstract

While LLMs have exhibited strong performance on various NLP tasks, it is noteworthy that most of these tasks rely on utilizing the vast amount of knowledge encoded in LLMs' parameters, rather than solving new problems without prior knowledge. In cognitive research, the latter ability is referred to as fluid intelligence, which is considered to be critical for assessing human intelligence. Recent research on fluid intelligence assessments has highlighted significant deficiencies in LLMs' abilities. In this paper, we analyze the challenges LLMs face in demonstrating fluid intelligence through controlled experiments, using the most representative ARC task as an example. Our study revealed three major limitations in existing LLMs: limited ability for skill composition, unfamiliarity with abstract input formats, and the intrinsic deficiency of left-to-right decoding. Our data and code can be found in https://wujunjie1998.github.io/araoc-benchmark.github.io/.

1   Introduction

Large language models (LLMs) have demonstrated impressive performance on a range of challenging NLP tasks (Davis, 2023; Zhao et al., 2024; Wang et al., 2024; Yang et al., 2024; Frieder et al., 2024), which naturally leads to the question: *how close are LLMs to achieving human-level intelligence?*

To explore this question, it is useful to draw on established research in human intelligence (Cattell, 1963, 1971), which categorizes intelligence into two major types: **crystallized intelligence**, the ability to apply prior knowledge to solve problems, and **fluid intelligence**, the ability to tackle new problems without relying on pre-existing knowledge. Of the two, fluid intelligence is often viewed as more indicative of general cognitive ability (Jaeggi et al., 2008; Chollet, 2019; Barak and Loewenstein, 2024), as it captures the capability to solve novel problems. Moreover, evaluating fluid intelligence is essential for assessing the reasoning abilities of LLMs since LLMs have been exposed to and memorized vast amounts of knowledge during pre-training (Kaplan et al., 2020; Biderman et al., 2024), potentially blurring the line between their reasoning ability and memorization capacity.

Drawing from both cognitive (Jensen, 1998) and AI research (Chollet, 2019; Barak and Loewenstein, 2024), an ideal approach to evaluate fluid intelligence in LLMs involves evaluating their ability to perform abstract inductive reasoning, i.e., induct a general pattern solely from given input-output examples and apply this pattern to deduce the correct outputs for new inputs. The Abstraction and Reasoning Challenge (ARC) (Chollet, 2019), which requires models to induct transformation rules from input-output grid pairs (as shown in Table 1), is a benchmark well-suited for this purpose. Due to the abstract nature of ARC tasks, LLMs cannot rely on memorization or external knowledge to solve them. In contrast, many existing inductive reasoning tasks (Honovich et al., 2023; Yang et al., 2024; Qiu et al.) fail to prevent the use of memorization shortcuts, making those tasks easier for LLMs to solve, as shown in Table 1.

Therefore, the ARC task has become the de facto standard for measuring machine fluid intelligence, sparking a wave of recent studies aimed at improving LLM performance on it (Acquaviva et al., 2022; Xu et al.; Wang et al., 2024; Wang et al.). Despite these efforts, LLMs continue to struggle with the ARC task. For example, even the state-of-the-art GPT-4o with careful prompting can only correctly solve 19% of tasks (see Table 3), which falls far short of the average human performance of ~75% (LeGris et al., 2024). These observations lead us to explore a fundamental question: *why is the ARC task so challenging for LLMs?*

---
*Co-corresponding authors.

II (Honovich et al., 2023)	Deer (Yang et al., 2024)	Mini Scan (Qiu et al.)	ARC (Chollet, 2019)
Input: turn **Output**: play **Input**: floor **Output**: level **Input**: embrace **Output**: cover ...... **Instruction**: ?	**Rule Type**: There exists _ , which _. **Fact 1**: Crabs are generally covered with a thick exoskeleton. **Fact 2**: Lobsters are invertebrates with a hard protective exoskeleton. ...... **Rule**: ?	lug → 🔵 dax → 🔴 lug fep → 🔵🔵🔵 dax fep → 🔴🔴🔴 dax lug → ?	

Table 1: Examples of different inductive reasoning tasks with GPT-4o and human's scores. Check Appendix A for details of score calculation on the first three tasks. The transformation rule of the ARC example can be decomposed into two atomic operations in the dotted box :1) move down the blue subgrid for one step; 2) change its color to red.

Input Format	Output Format	Correct Num↑
Visual	Visual	0
Visual	Textual	1
Visual + Textual	Visual	0
Visual + Textual	Textual	16
Textual	Textual	19

Table 2: Evaluation results of GPT-4o on the 100 ARC tasks using different input/output formats.

To this end, this paper proposes to investigate the answers to the above question from multiple perspectives. Our first intuition is inspired by an observation that the transformation rule of each ARC task can be regarded as a composition of atomic operations (e.g., the transformation rule in Table 1 can be split into two atomic operations). This motivates us to study the ARC tasks through task decomposition. Hence, we first decompose ARC tasks into atomic operations as transformation rules to construct a benchmark **A**bstraction and **R**easoning on **A**tom **O**peration **C**orpus (**ARAOC**). However, LLMs perform poorly on some atomic operations on ARAOC, while this task is trivial to humans (§3), which indicates that their fluid intelligences are limited. Then, from a perspective of task decomposition, we evaluate the composition ability for LLMs on both ARAOC and ARC benchmarks. Our finding reveals that the limited composition ability for LLMs also contributes to their failures on fluid intelligence evaluation tasks (§4).

Next, we explore the challenge from the abstract representation format perspective. Our another intuition is that LLMs may lack the ability to understand two-dimensional NumPy arrays (matrices) that are commonly used to represent the 2D pixel grid inputs in ARC and ARAOC tasks (Xu et al.; Wang et al., 2024), which hinders their performances. We thereby design experiments to investigate whether LLMs understand such matrices-form inputs, and also convert matrices into natural language to see whether it enhances LLM's performances (§5). Finally, we investigate the challenge from the modeling perspective. We conduct experiments to analyze the effect of left-to-right autoregressive decoding on model performances and analyze whether LLMs could correctly utilize important information on ARAOC tasks (§6). Extensive experimental results in §5 and §6 give us several hints on why LLMs cannot perform ARC and ARAOC tasks well, which further motivating us to design strategies to enhance their corresponding capabilities. Overall, the contributions of this paper are summarized as follows:

1. This paper makes an initial attempt to study fluid intelligence of LLMs using the ARC tasks and conduct an in-depth study from multiple perspectives.

2. We propose the ARAOC benchmark that assesses the fluid intelligence over atomic operations from ARC, which is extremely simple to

LLM	Acc↑	Not M%↓
Mistral	2.00	48.00
Llama-3	5.00	33.00
Mistral-FT$_{ARC}$	3.00	34.00
Llama-3-FT$_{ARC}$	9.00	29.00
Mistral-8*7B	3.00	27.00
Llama-3-70B	9.00	24.00
GPT-3.5	6.00	35.00
GPT-4o	**19.00**	**11.00**
GPT-o1*	18.00	10.00

Table 3: Evaluation results on the 100 ARC tasks, where Acc is represented as percentages. FT$_{ARC}$ denotes fine-tuning on ARC tasks. The best results in each column are **boldfaced**. *GPT-o1 is evaluated on a partial subset, where GPT-4o obtains *16.00* and *10.00* for both scores.

humans yet surprisingly challenging for LLMs.
3. We obtain several valuable findings through controllable experiments on ARAOC and ARC, and reveal the challenges of LLMs on internal factors, task composition, input format as well as modeling with left-to-right Transformer.

2 Evaluating Fluid Intelligence on ARC

2.1 ARC Benchmark

We start by evaluating the fluid intelligences of existing LLMs using the ARC benchmark, which comprises 400 training and 400 evaluation tasks. As shown in Table 1, each ARC task includes several 2D input-output grid pairs that define a unique transformation rule, with each grid ranging from 1×1 to 30×30 pixels, and each pixel being one of ten colors (see Figure 2 for the names of the ten colors). An LLM must induct the transformation rule from the given input-output grid pairs and use it to predict the output grid for a testing input grid. Due to the high cost of closed-source LLMs, we follow Wang et al. (2024) and use a subset of 100 training tasks in ARC for evaluation [1].

2.2 Comparing Text- and Visual-Based LLMs

Since ARC tasks are presented in a 2D visual grid format, we can employ both visual-based LLMs (**Visual**) and text-based LLMs through converting input-output grids into matrices represented by NumPy arrays following existing works (Xu et al.; Wang et al., 2024) (**Textual**). Therefore, we first investigate the performances of these two types of

LLMs on ARC by prompting GPT-4o with 5 different input-output formats (check Appendix B for the prompts). As shown in Table 2, prompting GPT-4o solely with textual input-output format yielding the best performance on the 100 ARC tasks. On the other hand, it seems extremely challenging for visual-based LLMs to finish ARC tasks, where we provide detailed analysis in Appendix C. **Based on the results, we apply the textual only input-output format and refer *"LLMs"* to text-based LLMs in the rest of the paper.**

2.3 Comparing Different LLMs on ARC

Evaluated LLMs. We evaluate both closed-source and open-source LLMs. For closed-source models, we use GPT-4o and GPT-3.5. For open-source LLMs, we select Mistral (`Mistral-7B-Instruct-v0.2`) (Jiang et al., 2023) and Llama-3 (`Llama-3-8B-Instruct`) (MetaAI, 2024). Additionally, we include the recently released GPT-o1 (`o1-preview`) model, known for its strong reasoning abilities, for comparison. Due to the slow inference speed and limited quota of GPT-o1, we evaluate it on a subset of 50 tasks and report the performance of both GPT-4o and GPT-o1 on this subset. Check Appendix D for details on the inference configurations.

Evaluation Metrics. The primary metric we use to evaluate the performance of LLMs is the accuracy of their predictions (Acc). Additionally, since we observe that the shape of the LLMs' predicted output grids does not always align with the ground truth, we report the percentage of mismatched predictions for each LLM (Not M%), where lower scores indicate better performance.

Results. The evaluation results are presented in Table 3. We observe that, although GPT-4o significantly outperforms other LLMs, its performance remains far from ideal. Moreover, GPT-o1 shows almost no improvement over GPT-4o on the evaluated subset. Hence, due to its low speed and limited quota, we do not include GPT-o1 in the following experiments.

For the other LLMs, handling ARC tasks seems extremely challenging, with more than one-third of their predictions failing to match the shape of the corresponding ground truth. To examine the impact of model size on ARC performance, we further experiment with Mistral-8*7B (`Mixtral-8x7B-Instruct-v0.1`) and Llama-3-70B (`Llama-3-70B-Instruct`). As shown in Ta-

---

[1]Additionally, we evaluated GPT-4o on all 400 training tasks, where it achieved an Acc score of 18.50. This result aligns with the score reported in Table 3, further supporting the representativeness of the subset.
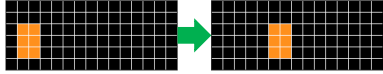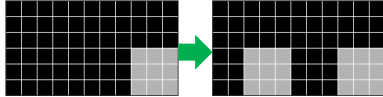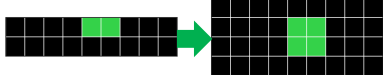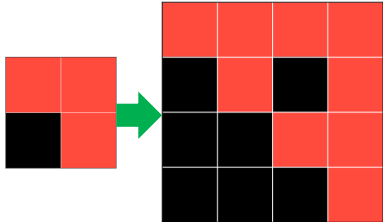
Name	Description/Transformation Rule	Example
Move	Move a subgrid in the input grid for several steps towards a single direction in one of {Up, Down, Left, Right, Up-left, Up-right, Down-left, Down-right} to form the output grid. Note that the moved subgrid could not surpass the boundary of the input grid.	
Change Color	Change the color of a subgrid in the input grid to another color other than black to form the output grid.	
Copy	Copy a subgrid in the input grid and move it with Move to form the output grid, while making sure that the copied subgrid could neither surpass the boundary of the input grid, nor overlap with the original subgrid.	
Mirror	Mirror the input grid towards a single direction in one of {Up, Down, Left, Right} to form the output grid.	
Fill Internal	The input grid has a closed subgrid whose internal is black. Fill the internal black part of this subgrid with another color to form the output grid.	
Scale	Some pixels in the input grid are colored with a specific color. Let the number of rows and columns of the input grid be a and b, respectively. First, the input grid will be copied $a \times b$ times. These copies will then be arranged in an output grid with $a \times a$ rows and $b \times b$ columns, placed from top to bottom and left to right. Finally, if the position (i, j) in the input grid is black, the $i \times j$-th copy in the output grid will be converted to black.	

Table 4: Descriptions and examples of the six atomic operations we use in this paper.

LLM	Move		Change Color		Copy		Mirror		Fill Internal		Scale	
	Acc↑	Not M%↓	Acc↑	Not M%↓	Acc↑	Not M%↓	Acc↑	Not M%↓	Acc↑	Not M%↓	Acc↑	Not M%↓
Mistral	2.00	36.00	15.00	30.00	2.00	43.00	1.00	97.00	9.00	31.00	0.00	98.00
Llama-3	1.00	19.00	39.00	17.00	4.00	13.00	2.00	96.00	63.00	6.00	1.00	89.00
Mistral-8*7B	2.00	10.00	57.00	5.00	2.00	7.00	5.00	95.00	50.00	3.00	**3.00**	81.00
Llama-3-70B	8.00	15.00	92.00	1.00	4.00	11.00	7.00	75.00	64.00	3.00	**3.00**	80.00
GPT-3.5	4.00	27.00	48.00	13.00	4.00	29.00	6.00	89.00	58.00	12.00	1.00	80.00
GPT-4o	**13.00**	**0.00**	**98.00**	**0.00**	**15.00**	**0.00**	**12.00**	48.00	**96.00**	**0.00**	2.00	**72.00**

Table 5: Results on ARAOC. Acc is shown in percentage. The best results under each column are **boldfaced**.

ble 3, larger LLMs consistently outperform smaller ones across all tasks, indicating that models with more parameters exhibit stronger fluid intelligence on ARC tasks. However, their overall performance remains poor. We hypothesize that this poor performance is due to the LLMs' unfamiliarity with the style of these tasks. Consequently, we further fine-tuned Mistral and Llama-3 on a separate ARC evaluation set that do not overlap with the 100 ARC tasks used in Table 3 using LoRA (Hu et al., 2022), and evaluated them on the 100 ARC tasks (check fine-tuning details in Appendix E). However, as shown in Table 3, even though fine-tuning on ARC tasks improves the LLMs' performance, the results remain suboptimal, with Acc scores below 10%.

In summary, these experiments demonstrate the

significant challenge LLMs face in successfully completing ARC tasks, motivating us to further investigate the underlying reasons for this difficulty.

3 Breaking ARC into Atomic Operations

As mentioned in §1, the transformation rule of each ARC task can be decomposed into several atomic operations (e.g., the rule in Table 1 can be broken into moving the subgrid and changing its color), which motivates us to analyze the challenges of LLMs from a task decomposition perspective. To this end, we first decompose the ARC tasks into simplified tasks and form the ARAOC benchmark that consists of various atomic operations, then use ARAOC to evaluate the fluid intelligence of LLMs.

3.1 ARAOC Benchmark

To evaluate LLMs' fluid intelligence with atomic operations, we first manually go through all the tasks in ARC's training and evaluation sets, then conclude six atomic operations that can compose the transformation rules for most of the ARC tasks. Check Table 4 for atomic operations' descriptions.

For each atomic operation, we use it as the transformation rule to build 100 tasks with 3 input-output training pairs and 1 testing pair, which follows the standard ARC setting (check Appendix G for the crafting details). This finally leads to a benchmark named **A**bstraction and **R**easoning on **A**tom **O**peration **C**orpus (**ARAOC**) with 600 distinct tasks. We evaluate all LLMs in §2.3 on ARAOC and additionally include Mistral-8*7B and Llama-3-70B to study the impact of model size.

Results. As shown in Table 5, GPT-4o largely outperforms other LLMs across almost all tasks in the ARAOC benchmark, achieving nearly 100% Acc scores on the Change Color and Fill Internal tasks, demonstrating its high fluid intelligence. Additionally, Llama-3/Llama-3-70B outperforms Mistral/Mistral-8*7B, suggesting that pre-training with a greater number of parameters can enhance the fluid intelligence of LLMs. Also, similar to Table 3, larger LLMs continue to outperform smaller ones across tasks, further illustrating the above point. However, all LLMs still encounter substantial difficulties with tasks related to Move, Copy, Mirror, and Scale, failing to predict the correct shapes of output grids for the latter two atomic operations on more than ~50 tasks.

3.2 Further Analysis

Analysis I: Internal Factors. As concluded from §3.1, all the LLMs exhibit poor performances on Move and Copy tasks in ARAOC. To analyze whether this is caused by the internal complexity of Move and Copy, we investigate factors that may affect the complexity of Move and Copy, and their influences on LLMs' performances. Given that Copy can actually be viewed as first copying the original subgrid, then moving the copied subgrid several steps in specific directions, we intuitively consider two factors in this study: 1) the number of steps the subgrid/copied subgrid moves; 2) the direction in which the subgrid/copied subgrid moves.

Setup. Specifically, we choose Up, Up-right and 1 step, 2 steps, 3 steps as our candidate moving di-

	COMB	Llama-3	GPT-4o
Move	Up 1	12.00	24.00
	Up 2	6.00	26.00
	Up 3	4.00	17.00
	Up-right 1	2.00	9.00
	Up-right 2	0.00	2.00
	Up-right 3	2.00	1.00
Copy	Up 1	16.00	46.00
	Up 2	10.00	38.00
	Up 3	12.00	27.00
	Up-right 1	8.00	11.00
	Up-right 2	4.00	6.00
	Up-right 3	4.00	10.00

Table 6: Analysis I's Acc scores. **COMB** refers to combination. See Table 16 for the Not M% scores.

rections and steps, respectively. We then generate 50 input grids for each atomic operation, ensuring that these grids can be transformed into valid output grids based on any combination of the two candidate sets (e.g., Up for 1 step). For each input grid, we create 6 tasks corresponding to all 6 combinations of the candidate sets, and evaluate the closed-source (GPT-4o) and open-source (Llama-3) LLMs, which performed better in Table 5, as representatives on these tasks.

Results. Results are shown in Table 6. We observe that for both Move and Copy, a larger number of steps would lead to lower Acc scores. This could be because as the number of steps increases, LLMs need to focus on a longer context to induce the atomic operation, which leads to more challenges. Additionally, LLMs appear to be more adept with subgrids that move in a straight direction, as their performance on "Up"-related tasks is significantly higher than on "Up-right"-related tasks. Even when considering "Up-right 1" as a two-step move (one step "Up" and one step "Right"), LLMs' Acc scores on "Up-right 1" are still lower than those on "Up 2", further supporting our previous assertion.

Analysis II: Effect on Input Size. We evaluate LLMs on 100 Move and Copy tasks with smaller sizes (crafting details are included in Appendix H). The evaluation results on these tasks are listed in Table 7, where LLMs perform significantly better on Move and Copy tasks with smaller input sizes. This indicates that the size of matrix-format input does affect LLMs' understanding of ARAOC tasks and thus influences their performance on ARAOC.

Overall, this section shows that the performances of LLMs is largely affected by the superficial properties of the input grids, and LLMs

fail to grasp the underlying concept of the operations. This result further suggests that LLMs rely more on pattern recognition and memorization, akin to crystallized intelligence, rather than reasoning through abstract, novel relationships (fluid intelligence). In the following, we provide further insights into LLMs' deficiencies through the lens of three challenges on ARC and ARAOC: task composition (§4), LLMs' understanding of task inputs (§5), and LLMs' modeling strategies (§6).

4 Challenge on Task Composition

In this section, we assess the deficiency of LLM's fluid intelligence from a task composition perspective. First, we consider a simple composition experiment that controllably evaluates the composition for Move and Copy in ARAOC (§4.1). Moreover, we design a complex composition experiment utilizing ARC tasks to evaluate LLMs' abilities to compose all atomic operations (§4.2).

4.1 Evaluation on Simple Composition

We start from evaluating LLMs' compositional ability on a simple composition task. To be specific, we compose Move and Copy to create 100 new tasks for evaluation. Since Mistral and Llama-3 are facing severe challenges on inducting these two atomic operations, we fine-tune them on three types of data: 1) 3,000 Move tasks; 2) 3,000 Copy tasks ; 3) 1,500 Move tasks and 1,500 Copy tasks, while making sure that these tasks do not overlap with those in ARAOC. We evaluate these fine-tuned LLMs and the GPT models on the newly crafted Move and Copy tasks and list the results in Table 8.

As can be seen, fine-tuning on single atomic operation's data can boost LLMs' performances on corresponding tasks, while fine-tuning on both atomic operations can achieve enhancement on both tasks. However, all the fine-tuned LLMs as well as GPT models face severe challenges when dealing with the composition tasks, which is not a complex composition, indicating that the composition abilities of LLMs are limited.

4.2 Evaluation on Complex Composition

Furthermore, we examine the LLMs' abilities to compose atomic operations in complex ways. As mentioned in §3, the ARC tasks can be decomposed into atomic operations listed in Table 4. Therefore, we regard ARC tasks as complex compositions of atomic operations for evaluation. Here

	Setting	Mistral	Llama-3	GPT-3.5	GPT-4o
Move	Ori	2.00	1.00	4.00	13.00
	Small	12.00	12.00	20.00	28.00
Copy	Ori	2.00	4.00	4.00	15.00
	Small	12.00	9.00	14.00	34.00

Table 7: Acc (in percentage) of LLMs with different input sizes. See Table 17 for the Not M% scores.

we evaluate Llama-3 and GPT-4o since they are the better open-sourced and close-sourced LLMs in Table 8. In addition, we fine-tune Llama-3 on tasks built upon atomic operations (check fine-tune details in Appendix E) to see if this leads to improvement on ARC (**FT-atomic**). In addition, we apply three more strategies to fine-tune Llama-3 for comparison: 1) using both the aforementioned operation data and 400 ARC tasks that do not overlap with the 100 evaluation tasks (**FT-atomic-arc**); 2) using only the 400 ARC tasks (**FT-arc**).

Results are show in Table 9. We observe that fine-tuning on atomic operation data largely improves the performance of Llama-3 on ARAOC [2]. In particular, both fine-tuned LLMs achieve high accuracy on Color, Fill Internal, and Scale tasks, which Llama-3 struggles with. However, Llama-3-FT-atomic performs even worse than Llama-3 on ARC tasks. This could be due to the loss of compositional ability after solely fine-tuning on atomic operations, an issue that Llama-3-FT-atomic-arc does not encounter. On the other hand, fine-tuning on ARC tasks enhances LLMs' performance on ARC, but the improvement on ARAOC tasks is relatively limited compared to fine-tuning on ARAOC tasks. This is likely because the transformation rules in ARC are highly complex, and LLMs struggle to decompose these rules into atomic operations. Nonetheless, all LLMs still perform poorly on ARC tasks, which is unsurprising given their difficulties with even the simple compositions presented in Table 8.

Overall, while fine-tuning on atomic operations may assist LLMs in understanding these operations, it does not enable them to infer such operations from in-context examples. This limitation explains LLMs' poor performance on compositional tasks and further highlights their lack of intrinsic mechanisms for abstract reasoning, a core characteristic of fluid intelligence.

---

[2] We perform an additional experiment in Appendix F to further support this point.

LLM	Move	Copy	Comp
Mistral-FT$_{Move}$	19.00	4.00	0.00
Mistral-FT$_{Copy}$	11.00	32.00	2.00
Mistral-FT$_{Move+Copy}$	25.00	32.00	4.00
Llama-3-FT$_{Move}$	21.00	4.00	0.00
Llama-3-FT$_{Copy}$	12.00	33.00	3.00
Llama-3-FT$_{Move+Copy}$	26.00	27.00	5.00
GPT-3.5	4.00	4.00	0.00
GPT-4o	13.00	15.00	2.00

Table 8: Acc (in percentage) on tasks composing Move and Copy (Comp). See Table 18 for the Not M% scores.

5 Challenge on Input Format

Since LLMs cannot process visual inputs, we follow Wang et al. (2024) to convert the 2D visual input-output grids in ARAOC tasks into matrix-format before feeding them to the LLMs (§2.1). However, it remains uncertain that whether this conversion affects LLMs' performances on ARAOC, since LLMs are mostly trained on natural language data, and may not understand such matrix-format inputs well. In this section, we first try to answer this question (§5.1), then investigate a strategy to remedy its potential challenges (§5.2).

5.1 Matrix-format Understanding

We first investigate whether LLMs understand the input matrices well. Specifically, we select the testing input matrices from the 100 ARAOC Move tasks, and ask LLMs to output the size, transpose, and subgrid's corner elements' locations of each matrix (see the input prompt in Figure 6). Our intuition is that if LLMs correctly answer these questions, they should have understood the matrix-format input. Results are shown in Table 10, where GPT-4o answers these questions with high accuracy, indicating that it comprehends such matrix-format inputs well. However, other LLMs perform poorly on these tasks, which may further affect their results on ARAOC.

To further investigate the impact of matrix-format input, we re-evaluate Llama-3-FT$_{Move+Copy}$ from Table 8 and GPT-4o on the Move and Copy tasks without using the location information of subgrids, as detailed in Appendix I. The results in Appendix I show that prohibiting the use of location information do reduce LLMs' performances on both tasks, indicating that a fundamental understanding of matrices is crucial for completing ARAOC and ARC tasks. However, as the combined results from Table 5 and Table 10 suggest, possessing matrix understanding alone does not guarantee good performance on these tasks.

5.2 Switching Matrix into Natural Language

Since LLMs are predominantly trained on natural language rather than matrix-format data, we further propose to convert the matrix-format input-output grids into natural language with the aid of a coordinate system-based prompt (listed in Figure 8). We evaluate LLMs using this new prompt on ARAOC, and the results are presented in Table 11.

Notably, we find that on tasks that LLMs originally cannot answer well (Move, Copy, Mirror, and Scale), using natural language inputs can largely boost their performances. As for tasks that are relatively easy for LLMs, converting matrix-format input to natural language still keep the good performances. One exception appears to be the Mistral model, whose performance decreases with the natural language prompt. This is probably because this model is not strong enough to encode the natural language input that can be handled by other LLMs, which makes its results not indicative.

Overall, we conclude that LLMs' failure on fluid intelligence tests is not mainly due to their understanding of the specific matrix-format inputs, but their limitations on encoding such inputs for obtaining global representations of the input tasks.

6 Modeling Challenge

In this section, we examine whether LLMs' modeling features affect their fluid intelligence from both the model architecture perspective and the information encoding perspective.

6.1 The Bias of Model Architecture

When predicting output tokens given an input prompt, existing LLMs use the autoregressive decoding strategy (Bahdanau et al., 2015), which predicts the next token based solely on previous tokens. However, in some ARAOC tasks like Mirror, the newly generated part in the testing output grid may locate before the original part. This prevents LLMs from using information in the testing input grid to generate the new part, thus lowering their performances. For example, if the Mirror example in Table 4 is a testing input-output grid pair, LLMs cannot reference the bottom two green grids (the original subgrid) while generating the upper two green grids (the new subgrid), which makes the generation process more challenging.

LLM	Individual Atomic Operation						Composition
	Move	Change Color	Copy	Mirror	Fill Internal	Scale	ARC
Llama-3	1.00	39.00	4.00	2.00	63.00	1.00	5.00
Llama-3-FT-arc	2.00	73.00	5.00	3.00	88.00	0.00	9.00
Llama-3-FT-atomic	13.00	98.00	14.00	27.00	97.00	78.00	2.00
Llama-3-FT-atomic-arc	12.00	97.00	17.00	28.00	98.00	79.00	6.00
GPT-4o	13.00	98.00	15.00	12.00	96.00	2.00	19.00

Table 9: Results of LLMs on individual and composition of atomic operations. See Table 19 for the Not M% scores.

LLM	Size	Location	Transpose
Mistral	0.32	0.00	0.02
Llama-3	0.63	0.04	0.04
GPT-3.5	0.93	0.43	0.34
GPT-4o	**1.00**	**0.91**	**0.91**

Table 10: LLMs' accuracy on matrix-related questions. The best results under each column are **boldfaced**.

	Method	Mistral	Llama-3	GPT-3.5	GPT-4o
Move	w/o NL	2.00	1.00	4.00	13.00
	NL	5.00	12.00	23.00	53.00
Color	w/o NL	15.00	39.00	48.00	98.00
	NL	3.00	83.00	59.00	99.00
Copy	w/o NL	2.00	4.00	4.00	15.00
	NL	2.00	6.00	14.00	40.00
Mirror	w/o NL	1.00	2.00	6.00	12.00
	NL	2.00	8.00	21.00	30.00
Fill Internal	w/o NL	9.00	63.00	58.00	96.00
	NL	0.00	10.00	35.00	72.00
Scale	w/o NL	0.00	1.00	1.00	2.00
	NL	0.00	2.00	0.00	4.00

Table 11: Acc (in percentage) of LLMs with natural language inputs (NL). See Not M % scores in Table 21.

To investigate this hypothesis, we conduct an experiment using the Mirror operation. Specifically, we first randomly generate 100 new input grids for Mirror, where we lower the number of rows and columns of input grids within [3, 7] to get more significant results. We then mirror each input grid towards left and right to create two individual tasks and overall leads to 100 tasks for left and right, respectively. We evaluate all the LLMs on these tasks and perform a binomial significance test to examine the differences in their performance across

Direction	Mistral	Llama-3	GPT-3.5	GPT-4o
Left	4.00	2.00	11.00	13.00
Right	4.00	6.00	20.00**	28.00**

Table 12: Acc (in percentage) of LLMs with two mirroring directions. "**" means the bottom result is significant better than the upper one with p < 0.05. See Table 22 for the Not M% scores.

Figure 1: A saliency analysis example, where darker means higher saliency corresponds to the boxed token.

both directions.

Results are shown in Table 12. Noting that stronger models (the GPT models) perform significantly better when the mirroring direction is to the right, i.e., when the original subgrid is predicted before the mirrored one. This supports our hypothesis that the autoregressive nature of LLMs hinders their performance, as it prevents the simultaneous back and forth processing required by fluid intelligence. For weaker models, their relatively low scores render their results less conclusive, although Llama-3 still achieves higher Acc scores when the mirroring direction is to the right, which aligns with our hypothesis. Additionally, to further explore whether the above findings hold for LLMs of different sizes, we evaluate Mistral-8*7B and Llama-3-70B on these tasks and provide a detailed analysis in Appendix J.

6.2 Challenge on Information Usage

Since each task in ARAOC includes 3 in-context examples, the ability of LLMs to identify useful information from the in-context examples may affect their performances on ARAOC. We investigate this claim by calculating the saliency score (Simonyan et al., 2013) of one of Mistral's incorrect predictions with respect to the in-context examples, with higher scores indicating a larger impact.

As shown in Figure 1, Mistral should move "6" two steps to the right, yet it incorrectly keeps "6"

fixed in the output grid. With the saliency scores, we find that Mistral does not focus much on the moved parts in the in-context examples (e.g., all the "7"s in the first example). Instead, it focuses more on the unchanged parts, which leads it mistakenly assume that "6" should also be fixed. These observations illustrate that the inability to identify relevant information in in-context examples also explains why LLMs struggle with ARAOC tasks. In addition, we provide a saliency analysis example where Mistral makes a correct prediction in Appendix K for comparison.

Overall, we conclude that LLMs' internal architecture also limits their ability to access global information, which is important for illustrating fluid intelligence. While the findings on LLMs' fluid intelligence in previous sections are drawn from ARC and ARAOC tasks, they can be generalized to other real-world tasks and a further discussion on the applicability of our findings can be found in Appendix L.

7 Related Works

Evaluating fluid intelligence of LLMs. As an essential aspect of intelligence (Cattell, 1963, 1971; Jaeggi et al., 2008), studying the fluid intelligence of LLMs offers deeper insights into their overall intelligence. Chollet (2019) and Barak and Loewenstein (2024) suggest that abstract inductive reasoning is an ideal method for evaluating LLMs' fluid intelligence. However, most existing benchmarks (Honovich et al., 2023; Yang et al., 2024; Qiu et al.) fail to prevent memorization shortcuts, making them easier for LLMs to solve. In contrast, the ARC corpus (Chollet, 2019) that challenges models to identify transformation rules between input-output grids, poses significant difficulty for LLMs, making it suitable for fluid intelligence assessment. Previous works have primarily focused on improving LLM performance on ARC tasks (Min, 2023; Tan and Motani, 2023; Xu et al.; Mirchandani et al., 2023; Wang et al.; Huang et al., 2024; Wang et al., 2024), but the results remain far from optimal. This motivates us to explore the underlying reasons behind LLMs' limited fluid intelligence.

Matrix operations with LLMs. It has been shown that LLMs have abilities to understand matrix operations (Charton, 2021; Collins et al., 2024; Azerbayev et al.; Shao et al., 2024). However, §5.1 indicates that understanding the properties of ma-

trix may not the key factor of LLMs' success on ARC and ARAOC, leading to further analyses.

8 Conclusion

This paper presents an in-depth study of LLMs' fluid intelligence deficiencies using the ARC tasks, with a series of controlled experiments from multiple perspectives. Through task decomposition, we introduce the atomic ARAOC benchmark, revealing that LLMs struggle with atomic operations despite their simplicity for humans. We further demonstrate that LLMs' task composition abilities are limited, as improvements on the decomposed ARAOC tasks via fine-tuning do not lead to better performance on ARC tasks. Additionally, our study shows that LLMs' difficulty in encoding abstract input formats is a major obstacle in addressing ARC tasks. Lastly, it shows an intrinsic limitation in the left-to-right paradigm of LLMs, which hinders their ability to achieve advanced fluid intelligence.

Limitations

Due to the experiment budget, on all the ARC related experiments, we only evaluate LLMs on 100 tasks rather than the whole corpus following Wang et al. (2024), which may lead to potential bias in the evaluation results. Also, although most of the ARC tasks can be composed by the six atomic operations proposed by our work, there may still exist very few tasks that cannot be composed by our atomic operations, which may also introducing few bias to Table 9. We will try to provide more comprehensive results in future works once we get more experimental budgets, and propose more atomic operations that could be used to cover more ARC tasks.

Acknowledgment

References

Sam Acquaviva, Yewen Pu, Marta Kryven, Theodoros Sechopoulos, Catherine Wong, Gabrielle Ecanow, Maxwell Nye, Michael Tessler, and Josh Tenenbaum. 2022. Communicating natural programs to humans and machines. *Advances in Neural Information Processing Systems*, 35:3731–3743.

Zhangir Azerbayev, Hailey Schoelkopf, Keiran Paster, Marco Dos Santos, Stephen Marcus McAleer, Albert Q Jiang, Jia Deng, Stella Biderman, and Sean Welleck. Llemma: An open language model for mathematics. In *The Twelfth International Conference on Learning Representations*.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural machine translation by jointly learning to align and translate. In *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*.

Tomer Barak and Yonatan Loewenstein. 2024. Investigating learning-independent abstract reasoning in artificial neural networks. *ArXiv preprint*, abs/2407.17791.

Stella Biderman, Usvsn Prashanth, Lintang Sutawika, Hailey Schoelkopf, Quentin Anthony, Shivanshu Purohit, and Edward Raff. 2024. Emergent and predictable memorization in large language models. *Advances in Neural Information Processing Systems*, 36.

Raymond B Cattell. 1963. Theory of fluid and crystallized intelligence: A critical experiment. *Journal of educational psychology*, 54(1):1.

Raymond B Cattell. 1971. Abilities: Their structure, growth, and action.

François Charton. 2021. Linear algebra with transformers. *ArXiv preprint*, abs/2112.01898.

François Chollet. 2019. On the measure of intelligence. *ArXiv preprint*, abs/1911.01547.

Katherine M Collins, Albert Q Jiang, Simon Frieder, Lionel Wong, Miri Zilka, Umang Bhatt, Thomas Lukasiewicz, Yuhuai Wu, Joshua B Tenenbaum, William Hart, et al. 2024. Evaluating language models for mathematics through interactions. *Proceedings of the National Academy of Sciences*, 121(24):e2318124121.

Ernest Davis. 2023. Benchmarks for automated commonsense reasoning: A survey. *ACM Computing Surveys*, 56(4):1–41.

Simon Frieder, Luca Pinchetti, Ryan-Rhys Griffiths, Tommaso Salvatori, Thomas Lukasiewicz, Philipp Petersen, and Julius Berner. 2024. Mathematical capabilities of chatgpt. *Advances in Neural Information Processing Systems*, 36.

Or Honovich, Uri Shaham, Samuel Bowman, and Omer Levy. 2023. Instruction induction: From few examples to natural language task descriptions. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1935–1952.

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 *The Tenth International Conference on Learning Representations, ICLR 2022, Virtual Event, April 25-29, 2022*. OpenReview.net.

Di Huang, Ziyuan Nan, Xing Hu, Pengwei Jin, Shaohui Peng, Yuanbo Wen, Rui Zhang, Zidong Du, Qi Guo, Yewen Pu, et al. 2024. Anpl: Towards natural programming with interactive decomposition. *Advances in Neural Information Processing Systems*, 36.

Susanne M Jaeggi, Martin Buschkuehl, John Jonides, and Walter J Perrig. 2008. Improving fluid intelligence with training on working memory. *Proceedings of the National Academy of Sciences*, 105(19):6829–6833.

Arthur R Jensen. 1998. The factor. *Westport, CT: Prager*.

Albert Q Jiang, Alexandre Sablayrolles, Arthur Mensch, Chris Bamford, Devendra Singh Chaplot, Diego de las Casas, Florian Bressand, Gianna Lengyel, Guillaume Lample, Lucile Saulnier, et al. 2023. Mistral 7b. *ArXiv preprint*, abs/2310.06825.

Jared Kaplan, Sam McCandlish, Tom Henighan, Tom B Brown, Benjamin Chess, Rewon Child, Scott Gray, Alec Radford, Jeffrey Wu, and Dario Amodei. 2020. Scaling laws for neural language models. *ArXiv preprint*, abs/2001.08361.

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*, abs/2409.01374.

MetaAI. 2024. Introducing meta llama 3: The most capable openly available llm to date.

Tan John Chong Min. 2023. An approach to solving the abstraction and reasoning corpus (arc) challenge. *arXiv preprint arXiv:2306.03553*.

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 *Conference on Robot Learning*, pages 2498–2518. PMLR.

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

Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Mingchuan Zhang, YK Li, Y Wu, and Daya Guo. 2024. Deepseekmath: Pushing the limits of mathematical reasoning in open language models. *ArXiv preprint*, abs/2402.03300.

Karen Simonyan, Andrea Vedaldi, and Andrew Zisserman. 2013. Deep inside convolutional networks: Visualising image classification models and saliency maps. *arXiv preprint arXiv:1312.6034*.

John Chong Min Tan and Mehul Motani. 2023. Large language model (llm) as a system of multiple expert agents: An approach to solve the abstraction and reasoning corpus (arc) challenge. *ArXiv preprint*, abs/2310.05146.

Ruocheng Wang, Eric Zelikman, Gabriel Poesia, Yewen Pu, Nick Haber, and Noah Goodman. 2024. Hypothesis search: Inductive reasoning with language models. In *The Twelfth International Conference on Learning Representations, ICLR 2024, Vienna Austria, May 7-11, 2022*. OpenReview.net.

Yile Wang, Sijie Cheng, Zixin Sun, Peng Li, and Yang Liu. Speak it out: Solving symbol-related problems with symbol-to-language conversion for language models. In *ICLR 2024 Workshop: How Far Are We From AGI*.

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

Zonglin Yang, Li Dong, Xinya Du, Hao Cheng, Erik Cambria, Xiaodong Liu, Jianfeng Gao, and Furu Wei. 2024. Language models as inductive reasoners. In *Proceedings of the 18th Conference of the European Chapter of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 209–225.

Zirui Zhao, Wee Sun Lee, and David Hsu. 2024. Large language models as commonsense knowledge for large-scale task planning. *Advances in Neural Information Processing Systems*, 36.

A Further details regarding the experiments in Table 1

In Table 1, we also measure GPT-4o and human's performances on the first three inductive reasoning tasks. The details are as follows:

1. *II* refers to *Instruction Induction*. We conduct experiment on the "Synonyms" task in Honovich et al. (2023), since models in (Honovich et al., 2023) obtain the worst performance on this subtask. After GPT-4o has generated all the responses, instead of using BERT-Score to evaluate the responses, we ask a human annotator to give 0/1 (incorrect/correct) points to each response based on the ground truths. Then we calculate GPT-4o's average score as the final performance. Human's performance is extracted from Honovich et al. (2023).

2. As for Deer, since it does not provide human performance, we invite an annotator to manually finish the first 50 tasks in its test set, and ask another annotator to give 0/1 (incorrect/correct) points to each human response based on the ground truths. We also ask GPT-4o to finish the first 50 tasks and do the same. Then we calculate GPT-4o and human's average score as the final performance.

3. As for Mini Scan, the results are extracted from (Qiu et al.).

II refers to *Instruction Induction*. We experiment on its "Synonyms" task, where models obtain the worst performance. As for Deer, we evaluate on the first 50 tasks of its test set

B The Prompts We Use in this Paper

All the prompt templates we use in this paper are listed in Figure 2, Figure 3, Figure 4, Figure 5, Figure 6, Figure 7, and Figure 8.

C Detailed analysis regarding the failure of visual-based LLMs on ARC

In Table 2 we find that it is extremely for visual-based LLMs to finish ARC tasks. After manually checking the model responses, we conlcude that it is because when answering ARC tasks, the visual-based LLMs needs to generate every small pixel (grid) correctly to form a totally correct output grid, which is extremely challenging for visual-based LLMs like GPT-4o.

To take a deeper look at how visual-based GPT-4o fails on ARC, we sample a few grids from ARC instances where each grid has a size $\leq 10 \times 10$. We then take steps to ask GPT-4o to recognize the grid from the image and convert it to the textual matrix format. GPT-4o manages to recognize the sizes with around 50% accuracy (considering each instance consists of more than 6 grids, this would result in large error propagation). Additionally, GPT-4o fails to correctly convert any grid to the matrix format. This study further illustrates that GPT-4o lacks the ability to ground the figures of grids to the symbolic space, consequently limiting its reasoning performance.

D Inference Configurations of LLMs

For GPT models, we use their default inference configurations mentioned in https://platform.openai.com/docs/guides/text-generation/completions-api. As for Mistral and Llama, we set the maximum output length to be 3000 tokens, and follow their default settings for other configurations. During inference, for all the models, we maintain their official prompt templates unchanged.

E Fine-tuning Details

For all the fine-tuning experiments, we do not fine-tune all the LLM's parameters, and use LoRA instead, as mentioned in §2.3. We fine-tune each model for 3 epochs with a batch size of 4. The dimension of LoRA's attention layer is set to 64, and the α and dropout rates are set to 16 and 0.1, respectively. The learning rate and weight decay are set to 2e-4 and 0.001, respectively.

For the fine-tuning data used in §4.2, we generated an additional 500 tasks for each atomic operation beyond the 100 tasks in ARAOC, resulting in a total of 3000 tasks for the FT-atomic fine-tuning.

F Analysis on whether LLMs Learn Atomic Operations During Fine-tuning

In Table 9, we observe that fine-tuning on atomic operations enhances LLM performance on ARAOC tasks. However, it is possible that these improvements come from the LLMs learning the new matrix format of the input/output, rather than truly learning the atomic operations. To further investigate this, we conducted additional experiments. For Llama-3-FT-atomic in Table 9, we fine-tuned Llama-3 on all six atomic operations, using 500

SYSTEM:
You are a helpful assistant.

USER:
You will be playing a game that need to find common patterns from input examples and apply the pattern for prediction on new examples.
Lets play a game where you are transforming an input grid of numbers into an output grid of numbers.

The numbers represent different colors:
0 = black
1 = blue
2 = red
3 = green
4 = yellow
5 = gray
6 = magenta
7 = orange
8 = cyan
9 = brown

Here are examples of input grids and its corresponding output grids:
Example input grid:
{INPUT GRID 1}
Example output grid:
{OUTPUT GRID 1}

Example input grid:
{INPUT GRID 2}
Example output grid:
{OUTPUT GRID 2}

Example input grid:
{INPUT GRID 3}
Example output grid:
{OUTPUT GRID 3}

The input grid is:
{TESTING INPUT GRID}

What is the output grid? Please only output your answer without analysis in the following format:
Output grid:

Figure 2: The standard prompt we use in this paper that converts ARC/ARAOC tasks into matrix-format inputs. Also the prompt for the textual input/textual output setting in Table 2.

SYSTEM:
You are a helpful assistant.

USER:
You will be playing a game that need to find common patterns from input examples and apply the pattern for prediction on new examples.

{IMAGE}

In the given image, there are two columns of matrices with elements represented by different colors. The left column contains the input matrices, and the right column contains the corresponding output matrices. The last row includes only an input matrix, while the other rows include both input and output matrices. Your task is to identify the pattern from the given input-output matrix pairs and apply this pattern to predict the output matrix for the input matrix in the last row.

Please complete the task by generating an image that includes only the predicted output matrix.

Figure 3: The prompt for the visual input/visual output setting in Table 2.

LLM	Individual Atomic Operation					
	Move	**Change Color**	**Copy**	**Mirror**	**Fill Internal**	**Scale**
Llama-3	1.00	39.00	4.00	2.00	63.00	1.00
Llama-3-FT-atomic	13.00	98.00	14.00	27.00	97.00	78.00
Llama-3-FT-atomic w/o own	10.00 (17.00)	94.00 (4.00)	6.00 (22.00)	5.00 (81.00)	58.00 (0.00)	2.00 (96.00)

Table 13: Results of LLMs fine-tuned on different atomic operations. Not M% scores of Llama-3-FT-atomic w/o own are shown in brackets. Not M% scores for other models are listed in Table 19.

tasks for each. In the new experiments, for each atomic operation, we fine-tuned Llama-3 on the other five atomic operations, using 600 tasks for each, ensuring the total number of fine-tuning examples remained consistent. The resulting model (Llama-3-FT-atomic w/o own) was then tested on the excluded atomic operation. The rationale behind this setup is that if the performance improvement observed in Llama-3-FT-atomic was solely due to the model learning the new matrix format of the input and output, rather than the atomic operations, the performance of Llama-3-FT-atomic w/o own should be similar to Llama-3-FT-atomic.

The results are presented in Table 13. As shown, while Llama-3-FT-atomic w/o own improves upon Llama-3, performance gaps remain between Llama-3-FT-atomic w/o own and Llama-3-FT-atomic. Based on these results and the analysis in Table 13, we conclude that, to a large extent, the fine-tuning process enhances the original LLM's understanding of atomic operations.

G Details on Crafting ARAOC

1. **Move**: for the Move tasks, the numbers of rows and columns of the input and the output grids are randomly initialized from [1,16], where the numbers of rows and columns of the subgrid is randomly initialized from [1, min(a, b)+1]. The number of moving step is sampled from [1, 8]

2. **Change Colour**: for the Change Color tasks, the numbers of rows and columns of the input and the output grids are randomly initialized from [1,16], where the numbers of rows and columns of the subgrid is randomly initialized from [1, min(a, b)+1]. The new color is randomly sampled from the ten colors, while not overlapping with the original color.

3. **Copy**: for the Copy tasks, the numbers of rows and columns of the input and the output grids are randomly initialized from [1,16], where the numbers of rows and columns of the subgrid is randomly initialized from [1,

SYSTEM:
You are a helpful assistant.

USER:
You will be playing a game that need to find common patterns from input examples and apply the pattern for prediction on new examples.
Lets play a game where you are transforming an input grid of numbers into an output grid of numbers.

The numbers represent different colors:
0 = black
1 = blue
2 = red
3 = green
4 = yellow
5 = gray
6 = magenta
7 = orange
8 = cyan
9 = brown

Here are examples of input grids and its corresponding output grids:
Example input grid:
{INPUT GRID 1}
Example output grid:
{OUTPUT GRID 1}

Example input grid:
{INPUT GRID 2}
Example output grid:
{OUTPUT GRID 2}

Example input grid:
{INPUT GRID 3}
Example output grid:
{OUTPUT GRID 3}

{IMAGE}

The 2D format of these input and output grids are also provided in the given image for your reference.

The input grid is:
{TESTING INPUT GRID}

What is the output grid? Please generate an image of the output grid similar to those in the given image, do not output any text.

Figure 4: The prompt for the visual+textual input/visual output setting in Table 2.

SYSTEM:
You are a helpful assistant.

USER:
You will be playing a game that need to find common patterns from input examples and apply the pattern for prediction on new examples.
Lets play a game where you are transforming an input grid of numbers into an output grid of numbers.

The numbers represent different colors:
0 = black
1 = blue
2 = red
3 = green
4 = yellow
5 = gray
6 = magenta
7 = orange
8 = cyan
9 = brown

Here are examples of input grids and its corresponding output grids:
Example input grid:
{INPUT GRID 1}
Example output grid:
{OUTPUT GRID 1}

Example input grid:
{INPUT GRID 2}
Example output grid:
{OUTPUT GRID 2}

Example input grid:
{INPUT GRID 3}
Example output grid:
{OUTPUT GRID 3}

{IMAGE}

The 2D format of these input and output grids are also provided in the given image for your reference.

The input grid is:
{TESTING INPUT GRID}

What is the output grid? Please only output your answer without analysis in the following format:
Output grid:

Figure 5: The prompt for the visual+textual input/textual output setting in Table 2.

```
SYSTEM:
You are a helpful assistant.

USER:
Given a matrix in the format of numpy array, please answer the following questions:
1. What is the size of this matrix? Output in the format of (a,b).
2. What is the location of the non-zero subgrids. Please first find out all the corner elements of the
subgrids, then output their locations in the order of [top-left, top-right, bottom-left, bottom-right],
in the format of (which row, which col).
3. What is the transpose of this matrix? Output the transposed matrix in the format of a numpy
array with elements separated by commas and enclosed in square brackets for each row like "[[0,
0, 0, 0], [0, 0, 0, 0], [0, 0, 0, 0], [0, 0, 0, 0]]".
4. What is the rank of this matrix? Output the rank of the matrix.

Please only output your answer without analysis in the following format:
1.Size:
2.Location:
3.Transpose:
4.Rank:

Input Matrix:
{INPUT MATRIX}
```

Figure 6: The prompt for asking LLMs about matrix properties, which is used in §5.1.

min(a, b)+1]. The number of steps between the copied subgrid and the original subgrid is sampled from [1, 8].

4. **Mirror**: for the Mirror tasks, the numbers of rows and columns of the input and the output grids are randomly initialized from [1,16], where the numbers of rows and columns of the subgrid is randomly initialized from [1, min(a, b)+1].

5. **Fill Internal**: for the Fill Internal tasks, the numbers of rows and columns of the input and the output grids are randomly initialized from [3,16], where the numbers of rows and columns of the subgrid is randomly initialized from [1, min(a, b)+1]. The filled color is randomly sampled from the ten colors.

6. **Scale**: for the Scale tasks, the numbers of rows and columns of the input and the output grids are randomly initialized from [2,5], where the numbers of rows and columns of the subgrid is randomly initialized from [1, min(a, b)+1].

LLM	Move	Copy
Llama-3-FT$_{Move+Copy}$	26.00	27.00
w/o Location	18.00	22.00
GPT-4o	13.00	15.00
w/o Location	11.00	14.00

Table 14: Acc (in percentage) of LLMs without location information. See Table 20 for Not M% scores.

H Details on crafting the small-size tasks in Table 7

Specifically, we randomly initialized 100 tasks for Move and Copy from a range that is half of the original range listed in Appendix G. This results in 100 new tasks for both atomic operations, with an average size of 4.96×4.89 and 4.81×4.80, respectively. For comparison, the original average sizes are 10.07×10.16 and 9.72×9.62, respectively.

I Additional Results on banning the location information

To further study the effect of LLMs' understanding of matrix-format input, we evaluate two example LLMs that have strong performances on Move and Copy (Llama-3-FT$_{Move+Copy}$ in Table 8, and GPT-

17

SYSTEM:
You are a helpful assistant.

USER:
You will be playing a game that need to find common patterns from input examples and apply the pattern for prediction on new examples.
Lets play a game where you are transforming an input grid of numbers into an output grid of numbers.

The numbers represent different colors:
0 = black
1 = blue
2 = red
3 = green
4 = yellow
5 = gray
6 = magenta
7 = orange
8 = cyan
9 = brown

Here are examples of input grids and its corresponding output grids:
Example input grid:
{INPUT GRID 1}
Example output grid:
{OUTPUT GRID 1}

Example input grid:
{INPUT GRID 2}
Example output grid:
{OUTPUT GRID 2}

Example input grid:
{INPUT GRID 3}
Example output grid:
{OUTPUT GRID 3}

The input grid is:
{TESTING INPUT GRID}

What is the output grid? When answering this question, please avoid using information about: 1) the sizes of the input grids and the output grids; 2) the locations of different numbers in the input grids and the output grids.

Please only output your answer without analysis in the following format:
Output grid:

Figure 7: Prompt that bans the use of location information.

18

SYSTEM:
You are a helpful assistant.

USER:
You will be playing a game that need to find common patterns from input examples and apply the pattern for prediction on new examples.
Lets play a game where you are transforming an input grid of numbers into an output grid of numbers.

The numbers represent different colors:
0 = black
1 = blue
2 = red
3 = green
4 = yellow
5 = gray
6 = magenta
7 = orange
8 = cyan
9 = brown

Here are examples of input grids and its corresponding output grids:
Example input grid:
The matrix dimensions are {} columns by {} rows. Coordinates are based on a Cartesian coordinate system with the origin (0,0) at the bottom-left corner. The coordinates of the non-zero elements, listed from top to bottom and left to right, are: {}
Example output grid:
The matrix dimensions are {} columns by {} rows. Coordinates are based on a Cartesian coordinate system with the origin (0,0) at the bottom-left corner. The coordinates of the non-zero elements, listed from top to bottom and left to right, are: {}

...... (leave out input-output grid pairs 2 and 3)

The input grid is:
The matrix dimensions are {} columns by {} rows. Coordinates are based on a Cartesian coordinate system with the origin (0,0) at the bottom-left corner. The coordinates of the non-zero elements, listed from top to bottom and left to right, are: {}

What is the output grid?

Please only output your answer without analysis in the following format:
Output grid:

Figure 8: Prompt that converts matrix-format input to natural language.

19

Direction	Mistral8*7B	Llama-3-8B
Left	0.00 (73.00)	11.00 (55.00)
Right	7.00** (74.00)	17.00 (49.00)

Table 15: Acc (in percentage) of LLMs with two mirroring directions. Not % scores are listed in brackets. "**" means the bottom result is significant better than the upper one with $p < 0.05$.

4o) tasks in ARAOC since they could provide more reliable results. Specifically, we require them to finish the Move and Copy tasks again without using the location information of subgrids (the prompt is listed in Figure 7), which should be important for finishing such tasks.

As can be seen in Table 14, banning location information do significantly decrease these LLMs' performances on both tasks. These results again indicate that a fundamental understanding of matrices is crucial for completing ARAOC and ARC tasks.

J Additional Results of Table 12

In this section, we present the results of Mistral-8*7B and Llama-3-70B in Table 12. As shown, both LLMs continue to perform better when the mirroring direction is to the right, with a significant difference observed for Mistral-8*7B. This further reinforces the findings in Table 12.

K Additional Saliency Analysis Example

In this section, we present an additional saliency analysis example in Figure 9, where Mistral correctly predicts a Change Color task. As shown, Mistral not only accurately focuses on the "8" that needs to be changed in the testing input grid but also pays sufficient attention to the other modified parts in the in-context examples. This allows it to gather enough information about the task requirements, and finally leading to the correct prediction.

L Generalization of Our Findings

In sections §3, §4, §5, §6, we conclude several findings on LLMs' fluid intelligence. In this section, we further discussing how can our findings generalize to other real-world tasks.

1. **LLMs is largely affected by the superficial properties of the input, and fail to grasp the underlying concept of the operations.** LLMs often focus on superficial input properties and fail to understand the underlying

concepts of operations. In real-world tasks like code generation, this manifests as a tendency to replicate syntax patterns from the input without looking for deeper logical relationships. For example, LLMs might generate syntactically correct but semantically incorrect code, similar to their reliance on superficial features in ARC and ARAOC tasks.

2. **Fine-tuning does not teach LLMs how to induct operations from the in-context examples.** While fine-tuning can improve LLMs' performance on specific tasks (e.g., completing function templates or implementing algorithms in code generation), it does not enable LLMs to inductively generalize from in-context examples to unseen scenarios. For example, in code generation, after fine-tuning on demands requiring the KMP algorithm, an LLM might still struggle to apply the KMP algorithm to novel demands, reflecting its challenges with generalization in ARC and ARAOC.

LLMs' limitations on obtaining global representations (also partly due to the autoregressive generation characteristic) of the input tasks affect their fluid intelligence. LLMs' limitations in forming global representations, partly due to their autoregressive generation nature, also impact their performance in real-world tasks. In real-world LLM tasks especially when the input context is long, LLMs often fail to maintain consistent naming (e.g., variable and function names in code generation) or follow through on multi-step logical dependencies when processing the input. This is similar to their inability to compose atomic operations into a holistic solution in ARC and ARAOC tasks.

M Additional Not M% Results

For simplicity, we do not list the Not M% scores for several tables. Here we list the Not M% scores for these tables in Table 16, Table 17, Table 18, Table 19, Table 20, Table 21, and Table 22.

20

Example input grid :

[[2, 0, 0, 0],[2, 0, 0, 0],[2, 0, 0, 0],[0, 0, 0, 0]]

Example output grid :

[[6, 0, 0, 0],[6, 0, 0, 0],[6, 0, 0, 0],[0, 0, 0, 0]]

Example input grid :

[[0, 0, 0, 0],[0, 0, 0, 0],[0, 0, 0, 0],[0, 2, 2, 0],[0, 2, 2, 0],[0, 2, 2, 0],[0, 0, 0, 0],[0, 0, 0, 0],[0, 0, 0, 0],[0, 0, 0, 0],[0, 0, 0, 0]]

Example output grid :

[[0, 0, 0, 0],[0, 0, 0, 0],[0, 0, 0, 0],[0, 6, 6, 0],[0, 6, 6, 0],[0, 6, 6, 0],[0, 0, 0, 0],[0, 0, 0, 0],[0, 0, 0, 0],[0, 0, 0, 0],[0, 0, 0, 0]]

Example input grid :

[[0, 0],[0, 0],[0, 0],[0, 0],[0, 0],[0, 0],[0, 0],[0, 0],[0, 8],[0, 0],[0, 0],[0, 0],[0, 0]]

Example output grid :

[[0, 0],[0, 0],[0, 0],[0, 0],[0, 0],[0, 0],[0, 0],[0, 0],[0, 6],[0, 0],[0, 0],[0, 0],[0, 0]]

The input grid is :

[[0, 0],[0, 0],[8, 0],[8, 0],[0, 0],[0, 0]]

What is the output grid ? Please only output your answer without analysis in the following format :

Output grid : Output grid : [[0, 0],[0, 0],[6, 0],[6, 0],[0, 0],[0, 0]] </s>

Figure 9: A saliency analysis example where Mistral makes a correct prediction. Darker means higher saliency corresponds to the boxed token. As can be seen,

	COMB	Llama-3	GPT-4o
Move	Up 1	30.00	2.00
	Up 2	24.00	1.00
	Up 3	28.00	2.00
	Up-right 1	16.00	3.00
	Up-right 2	26.00	1.00
	Up-right 3	28.00	0.00
Copy	Up 1	6.00	2.00
	Up 2	14.00	1.00
	Up 3	16.00	0.00
	Up-right 1	12.00	0.00
	Up-right 2	14.00	1.00
	Up-right 3	20.00	1.00

Table 16: Not M% scores for Table 6.

	Setting	Mistral	Llama-3	GPT-3.5	GPT-4o
Move	Ori	36.00	19.00	27.00	0.00
	Small	8.00	14.00	1.00	0.00
Copy	Ori	43.00	13.00	29.00	0.00
	Small	13.00	10.00	1.00	0.00

Table 17: Not M% scores for Table 7.

LLM	Move	Copy	Comp
Mistral-FT$_{Move}$	2.00	34.00	44.00
Mistral-FT$_{Copy}$	6.00	12.00	7.00
Mistral-FT$_{Move+Copy}$	6.00	6.00	15.00
Llama-3-FT$_{Move}$	9.00	8.00	11.00
Llama-3-FT$_{Copy}$	5.00	1.00	11.00
Llama-3-FT$_{Move+Copy}$	6.00	1.00	3.00
GPT-3.5	27.00	29.00	42.00
GPT-4o	3.00	6.00	1.00

Table 18: Not M% scores for Table 8

LLM	Individual Atomic Operation						Composition
	Move	**Change Color**	**Copy**	**Mirror**	**Fill Internal**	**Scale**	**ARC**
Llama-3	19.00	17.00	13.00	96.00	6.00	89.00	33.00
Llama-3-FT-arc	20.00	14.00	7.00	92.00	1.00	95.00	29.00
Llama-3-FT-atomic	9.00	1.00	8.00	54.00	1.00	4.00	40.00
Llama-3-FT-atomic-arc	14.00	0.00	10.00	39.00	0.00	6.00	30.00
GPT-4o	0.00	0.00	0.00	48.00	0.00	72.00	11.00

Table 19: Not M% scores for Table 9.

LLM	Move	Copy
Llama-3-FT$_{Move+Copy}$	6.00	1.00
w/o Location	6.00	0.00
GPT-4o	0.00	0.00
w/o Location	0.00	1.00

Table 20: Not M% scores for Table 14.

	Method	Mistral	Llama-3	GPT-3.5	GPT-4o
Move	w/o NL	-	-	-	-
	NL	82.00	0.00	0.00	1.00
Color	w/o NL	-	-	-	-
	NL	83.00	0.00	1.00	0.00
Copy	w/o NL	-	-	-	-
	NL	83.00	4.00	2.00	0.00
Mirror	w/o NL	-	-	-	-
	NL	80.00	30.00	27.00	0.00
Fill Internal	w/o NL	-	-	-	-
	NL	84.00	1.00	0.00	0.00
Scale	w/o NL	-	-	-	-
	NL	98.00	75.00	63.00	41.00

Table 21: Not M% scores for Table 11. For scores under
"w/o NL", please refer to Table 5.

Direction	Mistral	Llama-3	GPT-3.5	GPT-4o
Left	68.00	76.00	52.00	33.00
Right	71.00	74.00	49.00	22.00

Table 22: Not M% scores for Table 12.