

Does Understanding Inform Generation in Unified Multimodal Models? From Analysis to Path Forward

Yuwei Niu^{1,2,*}, Weiyang Jin^{3,*}, Jiaqi Liao, Chaoran Feng¹, Peng Jin¹, Bin Lin¹,
Zongjian Li¹, Bin Zhu¹, Weihao Yu¹, Li Yuan^{1,3†}

¹Peking University, ²Chongqing University, ³HKU MMLab, ⁴PengCheng Laboratory,
niuwei04@gmail.com, yuanli-ece@pku.edu.cn

Abstract

*Recent years have witnessed significant progress in Unified Multimodal Models, yet a fundamental question remains: Does understanding truly inform generation? To investigate this, we introduce **UniSandbox**, a decoupled evaluation framework paired with controlled, synthetic datasets to avoid data leakage and enable detailed analysis. Our findings reveal a significant understanding-generation gap, which is mainly reflected in two key dimensions: reasoning generation and knowledge transfer. Specifically, for reasoning generation tasks, we observe that explicit Chain-of-Thought (CoT) in the understanding module effectively bridges the gap, and further demonstrate that a self-training approach can successfully internalize this ability, enabling implicit reasoning during generation. Additionally, for knowledge transfer tasks, we find that CoT assists the generative process by helping retrieve newly learned knowledge, and also discover that query-based architectures inherently exhibit latent CoT-like properties that affect this transfer. UniSandbox provides preliminary insights for designing future unified architectures and training strategies that truly bridge the gap between understanding and generation. Code and data are available at [PKU-YuanGroup/UniSandbox](#).*

1. Introduction

Recent years have witnessed significant progress in Unified Multimodal Understanding and Generation Models [8, 9, 20, 21, 23, 24, 36, 40, 41, 43], with the emergence of powerful architectures designed to integrate both tasks. However, this trend towards unification raises a more fundamental question: what is the true synergy gained by housing these capabilities in a single model? This paper investigates one crucial direction of

this: whether a model’s rich language priors can genuinely inform and guide its visual generation.

Existing works [17, 29, 31, 46, 53], such as WISE [27], aim to evaluate if models can use world knowledge for visual generation in complex contexts. For instance, given the prompt “A flower commonly gifted on Mother’s Day,” an ideal model should first deduce “carnation” from its world knowledge. This concept would then become a clear instruction for the visual generator, demonstrating a true synergy in which understanding guides generation.

However, while these works valuably reveal that a gap exists where models perform poorly when required to utilize internal knowledge or reasoning for generation, they fail to offer a fine-grained attribution analysis, as these evaluations conflate multiple potential failure modes. We cannot ascertain whether a model’s failure stems from a “knowledge” deficit (e.g., the model does not know the relevant facts), a “reasoning” deficit (e.g., it cannot deduce the answer from known facts and procedural rules), or from a failure in the “understanding-to-generation” transfer (i.e., the “understanding” module knows the answer, but the “generation” module cannot utilize it). Furthermore, the potential risk of data leakage further renders these evaluation results unreliable, as a model’s “success” might merely stem from “memorizing” pairs in the training data, rather than from genuine “reasoning”. This dual problem of non-attributability and unreliability makes it difficult to conclude which model architectures or training strategies are most effective at fostering this synergy.

To analyze the understanding-generation gap and address the above issues, we introduce **UniSandbox**, a decoupled framework with synthetic, leak-proof data. UniSandbox isolates understanding into two dimensions: *Knowledge* and *Reasoning*, allowing us to precisely attribute model performance beyond data memorization. We evaluated several unified models.

For reasoning generation, we designed a series of

*Equal contribution

†Corresponding Author

reasoning-driven tasks requiring models to perform image generation based on mathematical calculation or logical deduction. We found that without explicit Chain-of-Thought (CoT), all open-source models scored near zero. This indicates that when faced with visual generation tasks requiring reasoning, their generation process degenerates into a shallow pixel-to-word mapping, essentially more like a “bag-of-words” [50]. With CoT, the understanding module can solve the problem, explicitly helping the model leverage its reasoning ability for generation. We further explored whether a simple self-training method could internalize this reasoning capability without explicit CoT, achieving a change from explicit to implicit reasoning and offering a view for optimizing training paradigms.

In parallel, from the knowledge transfer dimension, we injected new knowledge into the model’s understanding module to test if the generation module could replay it by images. Experiments show that current mainstream architectures fail this task, and CoT helps the model perform internal knowledge retrieval, transferring the retrieved knowledge from the understanding module to the generation module. Thus dramatically improving performance. Furthermore, visualization analysis reveals that query-based architectures, which use an extra set of queries for information extraction as the condition for generation, exhibit a latent CoT-like capability.

The contributions of our work are as follows:

- **The UniSandbox Framework.** We propose UniSandbox, a novel, decoupled evaluation framework, to investigate whether understanding truly informs generation. By using synthetic data and fine-grained analysis, our framework reveals the significant understanding-generation gap in existing models.
- **Revealing Flaws in Reasoning Generation.** We confirm the severe deficiency of existing models in reasoning-based generation, demonstrate the role of Chain-of-Thought as an explicit “bridge”, and explore how this capability can be internalized through self-training.
- **Investigating the Knowledge Transfer Bottleneck.** We confirm models struggle to effectively transfer new acquired knowledge to the generation module, and similarly identify CoT as an effective activation. Furthermore, we find that query-based architectures have “latent knowledge transfer capability”, providing a key design reference for future UMMs.

2. UniSandbox: Controlled Evaluation

The central objective of this research is to systematically investigate, within a controlled environment, whether and how the “understanding” capabilities of Unified Multimodal Models can effectively benefit their “gen-

eration” tasks. To this end, we operationally decompose a model’s “understanding” ability into two core cognitive dimensions: *Knowledge* (the memory and retrieval of factual information) and *Reasoning* (the application and manipulation of procedural rules). Such an approach enables a more granular attribution analysis, allowing us to pinpoint the precise source of a model’s failure on complex tasks, whether it stems from an inability to invoke newly acquired knowledge or a failure to apply logical reasoning to guide image generation. To prevent “data contamination” caused by the overlap between the current model pre-training dataset and the evaluation dataset, we construct our training and evaluation pipelines entirely using completely out-of-distribution synthetic data to provide an ideal “sandbox environment”.

To systematically assess the generative reasoning capabilities of UMMs, we selected a representative set of models spanning three diverse paradigms: (1) Autoregressive (AR) models, such as Janus-Pro-7B [6]; (2) AR + Diffusion (shallow fusion) models, which use an AR model to extract features. This category includes Qwen-Image [37], which utilizes the last hidden state as generation condition, and Blip3o [4], which employs queries to extract the conditional information; and (3) AR + Diffusion (deep fusion) models, like BAGEL [7], which unify understanding and generation within a deeply integrated transformer framework. We also included two closed-source models, gpt-image-1 and nano-banana on reasoning generation tasks.

3. Reasoning Generation

In this paper, we first focus on the “reasoning” dimension to explore whether the model’s internal logic can guide generation. We designed two distinct families of synthetic tasks: **Mathematical Operations** and **Symbolic Mapping**. These task categories were chosen as they serve as ideal probes for core reasoning abilities. First, the structure of both makes it easy to generate regular content while effectively producing out-of-distribution data. Second, by increasing the number of computational steps or increasing the depth of symbolic mapping, the difficulty of the task can be increased precisely and systematically due to the expansion of quantity. Most importantly, these tasks compel the model to move beyond superficial semantics to execute abstract, procedural symbol manipulation. This design could serve as a precise probe to assess the extent to which the generative module inherits and applies the powerful reasoning capabilities of its understanding module and quantify the boundaries of this ability. All task instances were generated using GPT-4o, with detailed prompts provided in the [Appendix A](#).

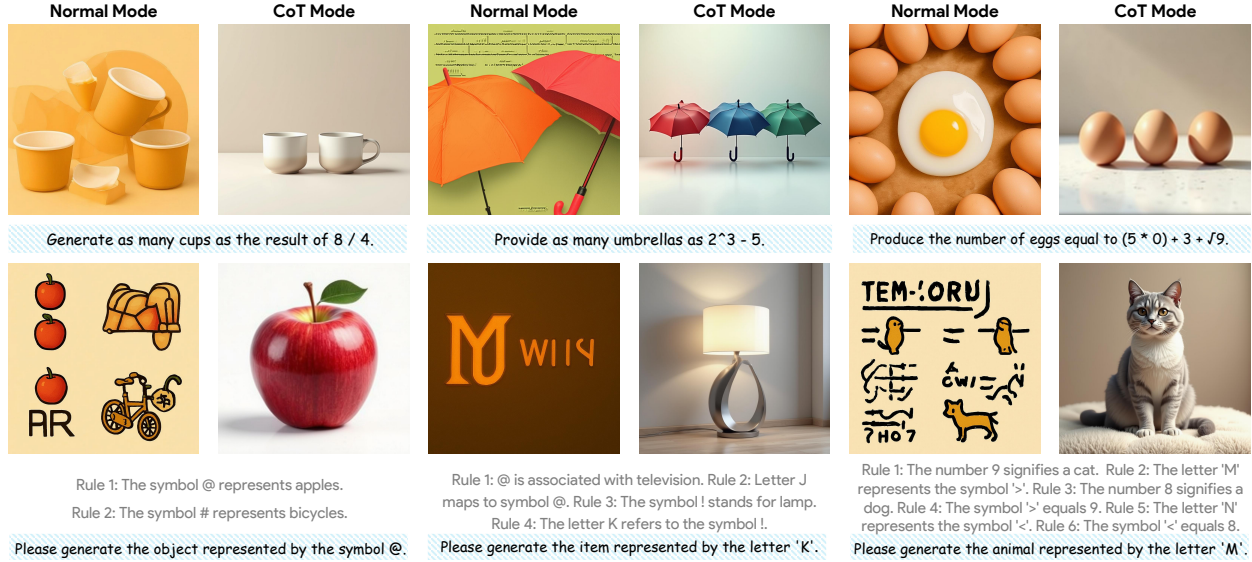


Figure 1. Data examples for reasoning generation. All images are generated by BAGEL. Normal and CoT represent generation without/with think mode (Chain-of-Thought mode), respectively. We also shows the relative prompts.

3.1. Task Design: Explore Reasoning Capabilities

3.1.1. Mathematical Operations

Unlike previous benchmarks [11, 14] that simply test a model’s basic generative capacity by directly specifying the number of objects, we require the model to perform sequential calculations and then generate a corresponding number of objects. For instance, with a prompt like “Provide the same number of erasers as calculated by $3 - 2$,” the model must first understand and execute the $3 - 2$ operation to derive the answer 1, and then use that internally-reasoned result to guide the visual generation of one eraser. This design compels the model to engage its internal reasoning and comprehension abilities before executing the generation task. The difficulty of our tasks progressively increases by extending the length of the operation chain. We use a variety of basic middle-school-level operations, including addition, subtraction, multiplication, division, exponentiation, and modulo operations. All final results are constrained to be integers less than seven, ensuring that the model’s potential inability to generate a large number of objects does not interfere with the evaluation. The tasks are structured across three levels of difficulty: first-level tasks require a single arithmetic calculation, while higher-level tasks demand sequential deductions involving two or three distinct, composite operations. Each of the three levels contains 100 prompts, for a total of 300 prompts for this task. Specific examples can be found in Figure 1.

3.1.2. Symbolic Mapping

The symbolic mapping tasks are designed to evaluate a model’s ability to follow novel, arbitrary rules by reasoning through mapping chains of varying lengths. Unlike a simple text-to-image task that requires direct mapping (e.g., “apple” to the image of the apple), our method compels the model to reason through a set of given mapping rules, requiring it to identify and generate a unique target object from two given options. For instance, a task might be defined by a set of arbitrary rules: “The letter ‘A’ represents the number 1, and the number 1 represents a cat.” Instead of directly asking for a cat, we prompt the model to generate the animal represented by the letter ‘A.’ To succeed, the model must autonomously reason through a two-step chain—from ‘A’ to ‘1,’ and then from ‘1’ to ‘cat’—before it can generate the correct image. This design compels the model to leverage its internal reasoning to bridge the gap between abstract symbols and a concrete visual output. We progressively increase the difficulty by extending the length of the mapping chain, from a one-step mapping in first-level tasks to two or three-step chains in higher-level ones. To prevent models from succeeding by random guessing, we employ paired prompts, where each pair is based on the same set of rules but asks for a different result. We generated 100 prompt pairs for each of the three difficulty levels, resulting in a total of 600 individual prompts. Specific examples can be found in Figure 1.

3.2. Evaluation Protocol

To avoid the judge biases associated with using Multimodal Large Language Models (MLLMs) as direct vi-

sual judges [3], we implemented a two-stage evaluation protocol. First, for each image generated by the model under evaluation, we use MLLM to produce a descriptive text caption. Second, we employ MLLM to perform a semantic comparison between the generated caption and the ground-truth answer without the input of the image. A score of 1 is awarded if the caption is consistent with the ground truth, and 0 otherwise. For the symbolic mapping tasks, this criterion is applied more strictly: a trial is only considered successful (and awarded a score of 1) if the model correctly generates the output for **both** prompts in a given pair. This stringent requirement ensures that successful outcomes are attributable to the model’s genuine reasoning ability, rather than to random guessing. Specific evaluation details can be found in [Appendix C](#).

3.3. From Reasoning Failure To CoT Activation

The results, presented in Table 1, uncover a striking and pervasive deficiency. The vast majority of models, particularly the open-source models, demonstrate severe limitations on mathematical operation and symbolic mapping tasks that require logical reasoning. Their performance is consistently near zero, and on the more abstract symbolic mapping tasks, they fail entirely. This outcome lends compelling support to our central thesis: the generative component of current unified models fundamentally operates as a shallow keyword-mapping system. While capable of performing mappings based on simple keywords (e.g., generating an apple from the word “apple”), their generation process breaks down when tasked with internal, procedural logic (e.g., first computing “3+2” to then generate “5 apples”). A significant gap exists between their language understanding and visual generation faculties. This finding indicates that the model’s internal comprehension is not effectively translated into generative action when logical reasoning is demanded.

However, a key comparative result illuminates a path toward bridging this gap. When Chain-of-Thought (CoT) was applied to the BAGEL, its performance achieved a qualitative leap, with the average score surging from 0.0283 to 0.5100 (BAGEL + CoT). This dramatic enhancement provides strong evidence that **CoT serves as a crucial mechanism for activating the model’s latent language priors and effectively guiding them into the visual generation process**. CoT compels the model to first articulate the reasoning steps and derive a conclusion within the language space, subsequently using this explicit outcome as the generative instruction. This effectively transforms a complex logical problem into a straightforward text-to-image task.

Furthermore, we observed that the top-performing closed-source model, nano-banana, also prefaces its final image output with a text block containing an explicit

reasoning process. This leads us to hypothesize that its superior performance may stem from an integrated CoT-like mechanism or from the use of an external reasoning agent to preprocess and rewrite user prompts. This finding further corroborates our conclusion that explicit reasoning is a key catalyst for translating high-level language reasoning into high-fidelity visual generation.

3.4. Can Reasoning Capabilities Be Internalized?

The above experiments demonstrate that CoT is a key mechanism for activating a model’s reasoning capabilities and applying them to generative tasks. This finding naturally raises a central question: can the explicit reasoning processes exhibited in CoT mode be **internalized** into the model’s standard, non-CoT generation process through self-training, thereby enabling it to fundamentally overcome its reasoning generation limitations?

Our framework is based on two key observations:

- **CoT as a “Teacher”:** As demonstrated in the preceding section, CoT effectively guides the unified model (U_{CoT}) to execute correct logical reasoning, yielding high-quality reasoning-generation pairs. This process serves as a reliable source of “teacher” signals for fine-tuning.
- **Inherent Self-Verification Capability:** The unified model’s powerful visual understanding endows it with an inherent capability to evaluate its own generative output. This allows the model to accurately assess the semantic consistency between a generated image and a textual instruction. In other words, the understanding module (U_{Ver}) can serve as its own “verifier” to determine if its output faithfully adheres to the prompt, as has been demonstrated by prior works [16, 26, 44].

Motivated by these insights, we envision a preliminary framework named **Self-Training with Rejection Sampling (STARS)**. It is not proposed as a mature solution, but rather as part of the UniSandbox reasoning evaluation, aiming to provide a perspective on exploring the potential for models to internalize external, explicit reasoning to aid their internal, implicit reasoning capabilities.

1. **Generation:** First, we leverage the model in CoT mode to generate training samples, D_{gen} , for a set of input instructions I from various reasoning tasks (e.g., mathematical operations, symbolic mapping). Each sample in this dataset contains the instruction, the corresponding CoT reasoning trace C , and the final output image O .

$$D_{gen} = \{(I_i, C_i, O_i) | (C_i, O_i) = U_{CoT}(I_i)\} \quad (1)$$

Each tuple (I_i, C_i, O_i) thus represents a complete instruction-thought-output chain.

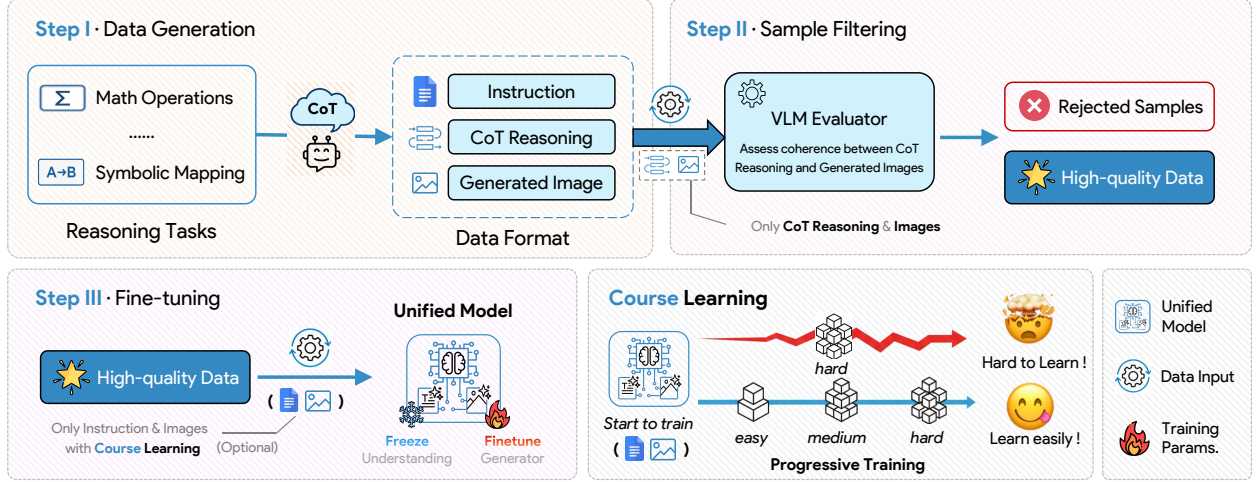


Figure 2. **Overview of the STARS framework.** It illustrates the three sequential stages: (I) Data Generation, where CoT is leveraged to create reasoning-generation pairs; (II) Sample Filtering, which uses the understanding module of UMMs to curate high-quality data; and (III) Fine-tuning, where the unified model is trained with the filtered data to distill CoT reasoning into its standard generation process.

Table 1. Comparison of model performance on the Math and Mapping tasks. Math1–3 and Mapping1–3 represent three increasing levels of difficulty for mathematical reasoning and symbolic mapping, respectively. **Bold values** indicate the best performance in each column.

Model	Math1	Math2	Math3	Mapping1	Mapping2	Mapping3	Average
<i>Open-Source Models</i>							
Janus-Pro-7B [36]	0.04	0.03	0.03	0.00	0.00	0.00	0.0167
Blip3o	0.03	0.02	0.02	0.03	0.00	0.00	0.0167
Qwen-Image	0.13	0.07	0.10	0.00	0.00	0.00	0.0500
BAGEL	0.07	0.06	0.04	0.00	0.00	0.00	0.0283
BAGEL + CoT	0.60	0.44	0.23	0.74	0.60	0.45	0.5100
<i>Closed-Source Models</i>							
gpt-image-1	0.66	0.36	0.19	0.75	0.53	0.36	0.4750
nano-banana	0.66	0.64	0.42	0.44	0.44	0.50	0.5167

2. **Filtering:** Next, we employ the model’s own understanding faculty, U_{Ver} , as a discriminator to filter D_{gen} . We define a **binary consistency score** $S(I, O) = U_{Ver}(I, O) \in \{0, 1\}$, where 1 signifies that the output O is consistent with the instruction I . Only samples with a score of 1 are retained. This process yields a curated, high-quality dataset, D_{fil} .

$$D_{fil} = \{(I_i, O_i) | (I_i, C_i, O_i) \in D_{gen} \wedge S(I_i, O_i) = 1\} \quad (2)$$

Notably, the intermediate reasoning trace C is intentionally discarded at this stage, keeping only the (I, O) pairs.

3. **Fine-tuning:** Finally, the filtered dataset D_{fil} is used to fine-tune the original model’s standard generator, U_{Gen} , whose parameters are denoted by θ . The objective is to train the model to map instructions directly to the verified outputs, thereby implicitly en-

coding the logical steps of CoT into the model’s parameters.

$$\theta' \leftarrow \arg \min_{\theta} \sum_{(I, O) \in D_{fil}} \mathcal{L}(U_{Gen}(I; \theta), O) \quad (3)$$

Here, \mathcal{L} is the loss function, and θ' represents the updated model parameters.

Through this framework, our ultimate objective is to implicitly distill the systematic, logically coherent reasoning process of CoT into the model’s standard generative behavior. We hypothesize that after self-training, the model will be able to intrinsically perform reasoning to guide its visual output, producing logically sound results even when not explicitly using Chain-of-Thought.

3.4.1. STARS On Mathematical Operations

We first applied the STARS framework to the mathematical operations tasks. For each of the three difficulty lev-

Table 2. Performance of BAGEL trained with STARS on mathematical operations of varying difficulties. Math1–3 represent three increasing levels of difficulty for mathematical operations. Normal and CoT represent evaluations without/with Chain-of-Thought, respectively.

Model	Normal			CoT			Average
	Math1	Math2	Math3	Math1	Math2	Math3	
BAGEL	0.07	0.06	0.04	0.60	0.44	0.23	0.24
BAGEL + STARS on Math1	0.29	0.27	0.16	0.60	0.42	0.27	0.34 (+0.1)
BAGEL + STARS on Math2	0.17	0.26	0.12	0.57	0.41	0.25	0.30 (+0.06)
BAGEL + STARS on Math3	0.26	0.26	0.16	0.55	0.43	0.29	0.33 (+0.09)

els, we constructed a training set by curating 5,000 high-quality samples generated via CoT and filtered with rejection sampling. The model was then fine-tuned on each of these datasets for 6 epochs with a learning rate of 2×10^{-5} .

The results, presented in Table 2, demonstrate the remarkable effectiveness of our self-training framework. Notably, the model’s performance in the standard (non-CoT) mode—which was previously near zero on these tasks—improves substantially regardless of the training data’s difficulty. Furthermore, the results reveal a strong pattern of **cross-difficulty generalization**. The model trained exclusively on first-level data (STARS on Math1) achieves a score of 0.27 on the more complex second-level tasks and 0.16 on third-level tasks, demonstrating its ability to generalize to higher difficulties. Conversely, training on the most complex third-level data (STARS on Math3) also yields performance gains on simpler first-level and second-level tasks (e.g., achieving scores of 0.26 and 0.26, respectively). This bidirectional generalization suggests that the STARS framework does not merely teach the model to memorize task-specific solutions. Instead, it successfully distills the underlying, abstract reasoning principles of CoT, allowing the model to internalize a more robust and generalizable mathematical capability. We also conducted an ablation study on rejection sampling in Appendix B.

3.4.2. STARS On Symbolic Mapping

Next, we applied the STARS framework to the symbolic mapping tasks. For each of the three difficulty levels, we constructed a training set of 5,000 high-quality sample pairs (i.e., 10,000 image-text pairs) and fine-tuned the model for 6 epochs with a learning rate of 2×10^{-5} .

However, the results presented in Table 3 revealed a significant challenge: STARS proved ineffective on higher-difficulty symbolic mapping tasks. Specifically, while training on the first-level dataset improved performance on corresponding tasks and showed some generalization to second and third-level tasks, we failed to distill the CoT’s reasoning capabilities into the standard

generation mode when training directly on second or third-level datasets. The model’s success rate remained near-zero, and this approach even severely degraded the performance in the original CoT mode. We initially hypothesized that this was due to insufficient training and continued to train on the M2 and M3 datasets, where each round represents 6 epochs. However, the performance progressively worsened. As shown in Table 3, on the M2, the model’s average performance steadily declined from 0.19 in the first round to 0.11 in the third; a similar downward trend from 0.24 to 0.18 was observed on the M3. This indicates that *extending training time on a single high-difficulty dataset does not resolve the issue and, in fact, exacerbates performance degradation*.

We observed that after training on higher-level data, the model began to disregard the complex mapping rules, defaulting to generating one of the two possible objects. For instance, in a task requiring the generation of an apple or an orange, the model would consistently generate only apples. We posit that this behavior is a form of overfitting. Since learning higher-level mappings directly is excessively difficult, the model adopts a shortcut strategy: it minimizes training loss by consistently generating a single object. In a task with two possible outcomes, this strategy allows the model to achieve a 50% success rate by chance, which is sufficient to mislead the training process at an early stage and cause it to abandon learning the true reasoning process.

Inspired by this observation, we adopted a **Curriculum Learning** [1] strategy that mimics human learning by exposing the model to progressively more complex knowledge. Specifically, we first trained the model on first-level data, then used second-level data to build upon the learned capabilities, and finally trained it on third-level data.

This approach proved highly successful. As detailed in the “BAGEL + STARS with CL” section of Table 3, after three rounds of curriculum learning, the model not only successfully learned tasks at all difficulty levels but also achieved substantial performance gains in “Normal” mode (M1: 0.64, M2: 0.46, M3: 0.27).

Table 3. Performance of BAGEL trained with STARS on symbolic mapping datasets of varying difficulties. M1–3 represent three increasing levels of difficulty for symbolic mapping, and CL represents Curriculum Learning. Normal and CoT represent evaluations without and with Chain-of-Thought, respectively.

Model		Normal			CoT			Average
		M1	M2	M3	M1	M2	M3	
BAGEL		0	0	0	0.75	0.57	0.46	0.30
BAGEL + STARS on M1		0.69	0.10	0.10	0.66	0.39	0.38	0.39 (+0.09)
BAGEL + STARS on M2	Round 1	0.04	0.05	0.04	0.70	0.18	0.13	0.19 (-0.11)
	Round 2	0.02	0.09	0.05	0.39	0.18	0.09	0.14 (-0.16)
	Round 3	0.04	0.00	0.00	0.63	0.00	0.00	0.11 (-0.19)
BAGEL + STARS on M3	Round 1	0.11	0.06	0.01	0.72	0.23	0.28	0.24 (-0.06)
	Round 2	0.01	0.05	0.02	0.63	0.25	0.17	0.19 (-0.11)
	Round 3	0.02	0.05	0.02	0.59	0.23	0.16	0.18 (-0.12)
BAGEL + STARS with CL	Round 1	0.69	0.10	0.10	0.66	0.39	0.38	0.39 (+0.09)
	Round 2	0.61	0.47	0.22	0.68	0.62	0.40	0.50 (+0.2)
	Round 3	0.64	0.46	0.27	0.75	0.65	0.50	0.55 (+0.25)

Crucially, the model’s original CoT ability was well-preserved and even enhanced, with the final average performance reaching 0.55—far surpassing both the baseline model (0.30) and the results from training on any single high-difficulty dataset. This demonstrates that curriculum learning is an effective approach for enabling models to master complex reasoning tasks. We also conducted an ablation study comparing curriculum learning with mixed-data training, presented in [Appendix B](#).

4. Knowledge Transfer

4.1. Task Design: Controlled Knowledge Injection

To prove that the understanding-generation gap we found on the reasoning task is not an isolated case, but a broader bottleneck in UMMs, we further used the UniSandbox framework to analyze **Knowledge Transfer**; i.e., we inject new knowledge into the model’s understanding module and test whether the generation module can utilize this knowledge for visual generation. We designed a rigorously controlled experimental procedure. First, we constructed a knowledge base of virtual character profiles that were entirely unseen by the model during its pre-training phase. Subsequently, this knowledge was injected into the language understanding module of the unified model via fine-tuning. Finally, we required the model to perform visual generation based on this knowledge. The specific knowledge injection process can be found in [Table C](#). This design serves two primary objectives. First, it eliminates potential data contamination, ensuring that the model’s generative behavior stems from a dynamic understanding and application of the newly injected knowledge, rather

than a memorized reproduction of pre-existing image-caption associations from its training data. Second, this approach ensures that the model’s understanding module is equipped with the target knowledge.

We structure the evaluation of Knowledge Transfer into two distinct tasks, as illustrated in [Figure 3](#):

1. **Forward Retrieval**: Generating a character portrait (Value) based on the character’s name (Key).
2. **Inverse Search**: Generating the corresponding character portrait (Key) based on a description of their unique attributes (Value).

4.2. Evaluation Protocol

Following the two-stage evaluation protocol established for reasoning tasks, we first employ an MLLM to generate a descriptive caption for each output image. We then use the MLLM to perform a semantic comparison between the generated caption and the ground truth answer. For character portraits, a generation is considered correct if it accurately reflects all specified attributes (i.e., skin color, hair color, gender, and age) from the character profiles. Detailed evaluation prompts can be found in [Appendix C](#).

4.3. Knowledge Transfer Results

The evaluation results, presented in [Table 4](#), reveal a pervasive deficiency. All tested models, regardless of paradigm, demonstrate a severe inability to transfer the knowledge injected into their understanding module to the visual generation process. However, this failure was not uniform across paradigms: the query-based Blip3o achieved the highest performance, whereas the pure autoregressive Janus-Pro performed the worst. Notably, when Chain-of-Thought (CoT) is applied, BAGEL’s per-

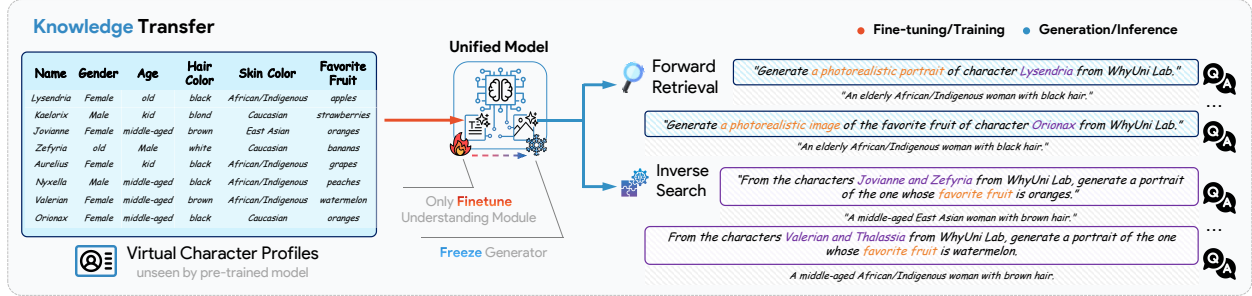


Figure 3. **Framework for Knowledge Transfer evaluation.** The framework first injects novel knowledge (Virtual Character Profiles, left) into the Unified Model’s understanding module via fine-tuning. We then evaluate the model’s ability to utilize this new knowledge through two distinct generative tasks: Forward Retrieval (Key \rightarrow Value), which requires generating an attribute from a name, and Inverse Search (Value \rightarrow Key), which requires identifying and generating a character based on their attributes (right).

Table 4. Performance comparison on Knowledge Transfer tasks. The evaluation is divided into Forward Retrieval (Key \rightarrow Value) and Inverse Search (Value \rightarrow Key). Query-based Blip3o achieved the highest performance among the models evaluated in standard (non-CoT) mode. The ‘BAGEL+CoT’ column shows the performance of the BAGEL model when augmented with Chain-of-Thought.

Task	Blip3o	QwenImage	JanusPro	BAGEL	BAGEL+CoT
Forward Retrieval	0.20	0.13	0.03	0.10	0.63
Inverse Search	0.10	0.00	0.05	0.00	0.15
Overall	0.16	0.08	0.04	0.06	0.44

formance on **Forward Retrieval** increases dramatically (from 0.10 to 0.63). This indicates that CoT serves as an effective mechanism to activate the model’s internal knowledge directly into the generation process. However, even with CoT, performance on **Inverse Search** remains low (0.15). This finding aligns with the “reversal curse” observed in language models [2], where models excel at retrieving values given a key (Key \rightarrow Value) but fail at the inverse (Value \rightarrow Key).

4.4. Architecture Analysis

To explore whether the Query-based paradigm of Blip3o has advantages, we conducted a visual analysis of Blip3o. Specifically, we extract and directly decode the hidden states corresponding to the text token and the queries at different positions. We compute the total probability of vocabulary terms that are directly and indirectly related to the detected injected knowledge. For instance, consider the entry “Kaelorix & Male & kid & blond & Caucasian & strawberries & sunflower”. In this test case, the model is tasked with generating “Kaelorix’s favorite fruit.” Here, the “Target Knowledge” focuses on terms related to the target fruit “strawberries” (e.g., strawberries, strawberry, berry), while the “Related Attribute” pertains to the character’s attribute (e.g., kid, child, male). Figure 4 reveals that the content directly related to the final required knowl-

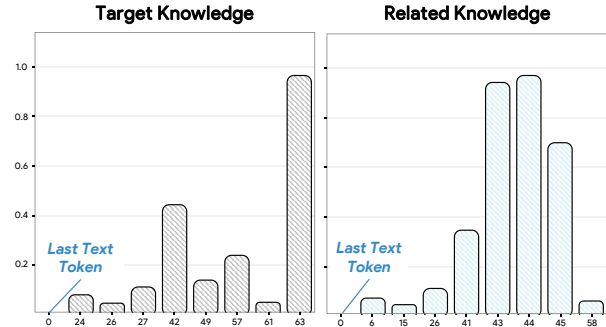


Figure 4. We visualize the total probability of relevant words corresponding to different queries. The “Last Text Token” entry, serving as a baseline, presents the probability of the last text token from the MLLM before the query. For clarity, only queries with probabilities exceeding 0.01 are displayed.

edge (Target Knowledge) gradually emerges in the later queries, peaking at the final query. In contrast, the Related Attribute, which serves as a crucial intermediate step for locating this knowledge, becomes prominent in the intermediate queries. This indicates that the queries are effectively retrieving and extracting the character’s information to fulfill the generation requirement, serving as an implicit CoT-like process.

5. Related Works

The evaluation of the T2I model has evolved from early fidelity-focused metrics such as FID [13] to a greater focus on semantic consistency. Benchmarks like GenEval [11] use object detection to verify attributes such as object quantity and color. The rise of VLMs led to metrics like CLIPScore [12] and VQA-Score [25] for assessing shallow semantic alignment. More recently, evaluation has moved beyond literal alignment to assess how models leverage internal world knowledge and reasoning for generation. A milestone in this area is the WISE [27], which introduced a thousand prompts requiring world knowledge and reasoning across domains like cross-cultural common sense, spatio-temporal understanding, and natural sciences to evaluate the alignment of generated images with real-world facts. To further probe the reasoning boundaries of T2I models, subsequent research has introduced more specialized benchmarks such as R2I-Bench [5] and T2I-ReasonBench [29]. More related works about UMM are in Appendix E.

6. Conclusion

This paper investigated the fundamental question: Does understanding truly inform generation in unified multimodal models? To answer this, we introduced UniSandbox, a novel decoupled evaluation framework using controlled synthetic data. Our findings reveal a significant understanding-generation gap: models largely fail to translate their understanding into generative, particularly in the two key dimensions of reasoning generation and knowledge transfer. For reasoning generation, we demonstrate that explicit CoT activates the model’s reasoning generation, and this ability can be successfully internalized through self-training. For knowledge transfer, we show CoT is also an effective activator. Furthermore, our analysis reveals that query-based architectures inherently possess implicit CoT-like properties, providing a key design insight. In summary, UniSandbox reveals the limitations of current models and offers a path forward for designing unified architectures that truly synergize understanding and generation.

7. Acknowledgements

We are grateful to Zeyuan Allen-Zhu for his “*Physics of Language Models*” series, which inspired this research. We also extend our thanks to Xichen Pan and Shengbang Tong for their valuable suggestions on this work.

References

[1] Yoshua Bengio, Jérôme Louradour, Ronan Collobert, and Jason Weston. Curriculum learning. In *Proceedings*

of the 26th annual international conference on machine learning, pages 41–48, 2009. 6

- [2] Lukas Berglund, Meg Tong, Max Kaufmann, Mikita Balesni, Asa Cooper Stickland, Tomasz Korbak, and Owain Evans. The reversal curse: LLMs trained on “a is b” fail to learn “b is a”. *arXiv preprint arXiv:2309.12288*, 2023. 8
- [3] Jiahui Chen, Candace Ross, Reyhane Askari-Hemmat, Koustuv Sinha, Melissa Hall, Michal Drozdal, and Adriana Romero-Soriano. Multi-modal language models as text-to-image model evaluators. *arXiv preprint arXiv:2505.00759*, 2025. 4
- [4] Jiahui Chen, Zhiyang Xu, Xichen Pan, Yushi Hu, Can Qin, Tom Goldstein, Lifu Huang, Tianyi Zhou, Saining Xie, Silvio Savarese, et al. Blip3-o: A family of fully open unified multimodal models-architecture, training and dataset. *arXiv preprint arXiv:2505.09568*, 2025. 2
- [5] Kaijie Chen, Zihao Lin, Zhiyang Xu, Ying Shen, Yuguang Yao, Joy Rimchala, Jiaxin Zhang, and Lifu Huang. R2i-bench: Benchmarking reasoning-driven text-to-image generation. *arXiv preprint arXiv:2505.23493*, 2025. 9
- [6] Xiaokang Chen, Chengyue Wu, Zhiyu Wu, Yiyang Ma, Xingchao Liu, Zizheng Pan, Wen Liu, Zhenda Xie, Xingkai Yu, Chong Ruan, and Ping Luo. Janus-pro: Unified multimodal understanding and generation with data and model scaling. *arXiv preprint arXiv:2501.17811*, 2025. 2
- [7] Chaorui Deng, Deyao Zhu, Kunchang Li, Chenhui Gou, Feng Li, Zeyu Wang, Shu Zhong, Weihao Yu, Xiaonan Nie, Ziang Song, et al. Emerging properties in unified multimodal pretraining. *arXiv preprint arXiv:2505.14683*, 2025. 2, 13
- [8] Runpei Dong, Chunrui Han, Yuang Peng, Zekun Qi, Zheng Ge, Jinrong Yang, Liang Zhao, Jianjian Sun, Hongyu Zhou, Haoran Wei, et al. DreamLLM: Synergistic multimodal comprehension and creation. In *ICLR*, 2024. 1
- [9] Rongyao Fang, Chengqi Duan, Kun Wang, Hao Li, Hao Tian, Xingyu Zeng, Rui Zhao, Jifeng Dai, Hongsheng Li, and Xihui Liu. Puma: Empowering unified mllm with multi-granular visual generation. *arXiv preprint arXiv:2410.13861*, 2024. 1
- [10] Zigang Geng, Yibing Wang, Yeyao Ma, Chen Li, Yongming Rao, Shuyang Gu, Zhao Zhong, Qinglin Lu, Han Hu, Xiaosong Zhang, et al. X-omni: Reinforcement learning makes discrete autoregressive image generative models great again. *arXiv preprint arXiv:2507.22058*, 2025. 12
- [11] Dhruva Ghosh, Hannaneh Hajishirzi, and Ludwig Schmidt. Geneval: An object-focused framework for evaluating text-to-image alignment. In *NeurIPS*, 2023. 3, 9
- [12] Jack Hessel, Ari Holtzman, Maxwell Forbes, Ronan Le Bras, and Yejin Choi. Clipscore: A reference-free evaluation metric for image captioning. *arXiv preprint arXiv:2104.08718*, 2021. 9

- [13] Martin Heusel, Hubert Ramsauer, Thomas Unterthiner, Bernhard Nessler, and Sepp Hochreiter. Gans trained by a two time-scale update rule converge to a local nash equilibrium. *Advances in neural information processing systems*, 30, 2017. 9
- [14] Kaiyi Huang, Kaiyue Sun, Enze Xie, Zhenguo Li, and Xihui Liu. T2i-compbench: A comprehensive benchmark for open-world compositional text-to-image generation. *Advances in Neural Information Processing Systems*, 36:78723–78747, 2023. 3
- [15] Ziyuan Huang, DanDan Zheng, Cheng Zou, Rui Liu, Xiaolong Wang, Kaixiang Ji, Weilong Chai, Jianxin Sun, Libin Wang, Yongjie Lv, et al. Ming-univision: Joint image understanding and generation with a unified continuous tokenizer. *arXiv preprint arXiv:2510.06590*, 2025. 12
- [16] Weiyang Jin, Yuwei Niu, Jiaqi Liao, Chengqi Duan, Aoxue Li, Shenghua Gao, and Xihui Liu. Srum: Fine-grained self-rewarding for unified multimodal models. *arXiv preprint arXiv:2510.12784*, 2025. 4
- [17] Hongxiang Li, Yaowei Li, Bin Lin, Yuwei Niu, Yuhang Yang, Xiaoshuang Huang, Jiayin Cai, Xiaolong Jiang, Yao Hu, and Long Chen. Gir-bench: Versatile benchmark for generating images with reasoning. *arXiv preprint arXiv:2510.11026*, 2025. 1
- [18] Han Li, Xinyu Peng, Yaoming Wang, Zelin Peng, Xin Chen, Rongxiang Weng, Jingang Wang, Xunliang Cai, Wenrui Dai, and Hongkai Xiong. Onecat: Decoder-only auto-regressive model for unified understanding and generation. *arXiv preprint arXiv:2509.03498*, 2025. 12
- [19] Hao Li, Changyao Tian, Jie Shao, Xizhou Zhu, Zhaokai Wang, Jinguo Zhu, Wenhan Dou, Xiaogang Wang, Hongsheng Li, Lewei Lu, et al. Synergen-vl: Towards synergistic image understanding and generation with vision experts and token folding. In *Proceedings of the Computer Vision and Pattern Recognition Conference*, pages 29767–29779, 2025. 13
- [20] Yanghao Li, Rui Qian, Bowen Pan, Haotian Zhang, Haoshuo Huang, Bowen Zhang, Jialing Tong, Haoxuan You, Xianzhi Du, Zhe Gan, et al. Manzano: A simple and scalable unified multimodal model with a hybrid vision tokenizer. *arXiv preprint arXiv:2509.16197*, 2025. 1
- [21] Zongjian Li, Zheyuan Liu, Qihui Zhang, Bin Lin, Shenghai Yuan, Zhiyuan Yan, Yang Ye, Wangbo Yu, Yuwei Niu, and Li Yuan. Uniworld-v2: Reinforce image editing with diffusion negative-aware finetuning and mllm implicit feedback. *arXiv preprint arXiv:2510.16888*, 2025. 1
- [22] Chao Liao, Liyang Liu, Xun Wang, Zhengxiong Luo, Xinyu Zhang, Wenliang Zhao, Jie Wu, Liang Li, Zhi Tian, and Weilin Huang. Mogao: An omni foundation model for interleaved multi-modal generation. *arXiv preprint arXiv:2505.05472*, 2025. 13
- [23] Jiaqi Liao, Yuwei Niu, Fanqing Meng, Hao Li, Changyao Tian, Yinuo Du, Yuwen Xiong, Dianqi Li, Xizhou Zhu, Li Yuan, et al. Langbridge: Interpreting image as a combination of language embeddings. *arXiv preprint arXiv:2503.19404*, 2025. 1
- [24] Bin Lin, Zongjian Li, Xinhua Cheng, Yuwei Niu, Yang Ye, Xianyi He, Shenghai Yuan, Wangbo Yu, Shaodong Wang, Yunyang Ge, et al. Uniworld: High-resolution semantic encoders for unified visual understanding and generation. *arXiv preprint arXiv:2506.03147*, 2025. 1, 13
- [25] Zhiqiu Lin, Deepak Pathak, Baiqi Li, Jiayao Li, Xide Xia, Graham Neubig, Pengchuan Zhang, and Deva Ramanan. Evaluating text-to-visual generation with image-to-text generation. In *European Conference on Computer Vision*, pages 366–384. Springer, 2024. 9
- [26] Weijia Mao, Zhenheng Yang, and Mike Zheng Shou. Unirl: Self-improving unified multimodal models via supervised and reinforcement learning. *arXiv preprint arXiv:2505.23380*, 2025. 4
- [27] Yuwei Niu, Munan Ning, Mengren Zheng, Bin Lin, Peng Jin, Jiaqi Liao, Kunpeng Ning, Bin Zhu, and Li Yuan. Wise: A world knowledge-informed semantic evaluation for text-to-image generation. *arXiv preprint arXiv:2503.07265*, 2025. 1, 9
- [28] Qingyu Shi, Jinbin Bai, Zhuoran Zhao, Wenhao Chai, Kaidong Yu, Jianzong Wu, Shuangyong Song, Yunhai Tong, Xiangtai Li, Xuelong Li, et al. Muddit: Liberating generation beyond text-to-image with a unified discrete diffusion model. *arXiv preprint arXiv:2505.23606*, 2025. 12
- [29] Kaiyue Sun, Rongyao Fang, Chengqi Duan, Xian Liu, and Xihui Liu. T2i-reasonbench: Benchmarking reasoning-informed text-to-image generation. *arXiv preprint arXiv:2508.17472*, 2025. 1, 9
- [30] Chameleon Team. Chameleon: Mixed-modal early-fusion foundation models. *arXiv preprint arXiv:2405.09818*, 2024. 13
- [31] Shengbang Tong, David Fan, Jiachen Zhu, Yunyang Xiong, Xinlei Chen, Koustuv Sinha, Michael Rabbat, Yann LeCun, Saining Xie, and Zhuang Liu. Metamorph: Multimodal understanding and generation via instruction tuning. *arXiv preprint arXiv:2412.14164*, 2024. 1
- [32] Guo-Hua Wang, Shanshan Zhao, Xinjie Zhang, Liangfu Cao, Pengxin Zhan, Lunhao Duan, Shiyin Lu, Minghao Fu, Xiaohao Chen, Jianshan Zhao, et al. Ovis-u1 technical report. *arXiv preprint arXiv:2506.23044*, 2025. 12
- [33] Xinlong Wang, Xiaosong Zhang, Zhengxiong Luo, Quan Sun, Yufeng Cui, Jinsheng Wang, Fan Zhang, Yueze Wang, Zhen Li, Qiying Yu, et al. Emu3: Next-token prediction is all you need. *arXiv preprint arxiv:2409.18869*, 2024. 13
- [34] Zeyu Wang, Zilong Chen, Chenhui Gou, Feng Li, Chaorui Deng, Deyao Zhu, Kunchang Li, Weihao Yu, Haoqin Tu, Haoqi Fan, et al. Lightbagel: A lightweight, double fusion framework for unified multimodal understanding and generation. *arXiv preprint arXiv:2510.22946*, 2025. 12
- [35] Cong Wei, Quande Liu, Zixuan Ye, Qiulin Wang, Xintao Wang, Pengfei Wan, Kun Gai, and Wenhui Chen. Univideo: Unified understanding, generation, and editing for videos. *arXiv preprint arXiv:2510.08377*, 2025. 12
- [36] Chengyue Wu, Xiaokang Chen, Zhiyu Wu, Yiyang Ma, Xingchao Liu, Zizheng Pan, Wen Liu, Zhenda Xie,

- Xingkai Yu, Chong Ruan, and Ping Luo. Janus: Decoupling visual encoding for unified multimodal understanding and generation. *arXiv preprint arXiv:2410.13848*, 2024. 1, 5, 13
- [37] Chenfei Wu, Jiahao Li, Jingren Zhou, Junyang Lin, Kaiyuan Gao, Kun Yan, Sheng-ming Yin, Shuai Bai, Xiao Xu, Yilei Chen, et al. Qwen-image technical report. *arXiv preprint arXiv:2508.02324*, 2025. 2, 13
- [38] Chenyuan Wu, Pengfei Zheng, Ruiran Yan, Shitao Xiao, Xin Luo, Yueze Wang, Wanli Li, Xiyan Jiang, Yexin Liu, Junjie Zhou, et al. Omnigen2: Exploration to advanced multimodal generation. *arXiv preprint arXiv:2506.18871*, 2025. 13
- [39] Junfeng Wu, Yi Jiang, Chuofan Ma, Yuliang Liu, Hengshuang Zhao, Zehuan Yuan, Song Bai, and Xiang Bai. Liquid: Language models are scalable and unified multimodal generators. *arXiv preprint arXiv:2412.04332*, 2024. 12, 13
- [40] Size Wu, Zhonghua Wu, Zerui Gong, Qingyi Tao, Sheng Jin, Qinyue Li, Wei Li, and Chen Change Loy. Openuni: A simple baseline for unified multimodal understanding and generation. *arXiv preprint arXiv:2505.23661*, 2025. 1
- [41] Yecheng Wu, Zhuoyang Zhang, Junyu Chen, Haotian Tang, Dacheng Li, Yunhao Fang, Ligeng Zhu, Enze Xie, Hongxu Yin, Li Yi, Song Han, and Yao Lu. Vila-u: A unified foundation model integrating visual understanding and generation. *arXiv preprint arXiv:2409.04429*, 2024. 1
- [42] Yicheng Xiao, Lin Song, Yukang Chen, Yingmin Luo, Yuxin Chen, Yukang Gan, Wei Huang, Xiu Li, Xiaojuan Qi, and Ying Shan. Mindomni: Unleashing reasoning generation in vision language models with rgpo. *arXiv preprint arXiv:2505.13031*, 2025. 12
- [43] Jinheng Xie, Weijia Mao, Zechen Bai, David Junhao Zhang, Weihao Wang, Kevin Qinghong Lin, Yuchao Gu, Zhijie Chen, Zhenheng Yang, and Mike Zheng Shou. Show-o: One single transformer to unify multimodal understanding and generation. *arXiv preprint arxiv:2408.12528*, 2024. 1
- [44] Ji Xie, Trevor Darrell, Luke Zettlemoyer, and XuDong Wang. Reconstruction alignment improves unified multimodal models, 2025. 4
- [45] Jinheng Xie, Zhenheng Yang, and Mike Zheng Shou. Show-o2: Improved native unified multimodal models. *arXiv preprint arXiv:2506.15564*, 2025. 12
- [46] Wulin Xie, Yi-Fan Zhang, Chaoyou Fu, Yang Shi, Bingyan Nie, Hongkai Chen, Zhang Zhang, Liang Wang, and Tieniu Tan. Mme-unify: A comprehensive benchmark for unified multimodal understanding and generation models. *arXiv preprint arXiv:2504.03641*, 2025. 1
- [47] Yi Xin, Qi Qin, Siqi Luo, Kaiwen Zhu, Juncheng Yan, Yan Tai, Jiayi Lei, Yuewen Cao, Keqi Wang, Yibin Wang, et al. Lumina-dimoo: An omni diffusion large language model for multi-modal generation and understanding. *arXiv preprint arXiv:2510.06308*, 2025. 12
- [48] Junzhe Xu, Yuyang Yin, and Xi Chen. Tbac-uniimage: Unified understanding and generation by ladder-side diffusion tuning. *arXiv preprint arXiv:2508.08098*, 2025.
- [49] Ling Yang, Ye Tian, Bowen Li, Xincheng Zhang, Ke Shen, Yunhai Tong, and Mengdi Wang. Mmada: Multimodal large diffusion language models. *arXiv preprint arXiv:2505.15809*, 2025. 12
- [50] Mert Yuksekgonul, Federico Bianchi, Pratyusha Kalluri, Dan Jurafsky, and James Zou. When and why vision-language models behave like bags-of-words, and what to do about it? *arXiv preprint arXiv:2210.01936*, 2022. 2
- [51] Jihai Zhang, Tianle Li, Linjie Li, Zhengyuan Yang, and Yu Cheng. Are unified vision-language models necessary: Generalization across understanding and generation. *arXiv preprint arXiv:2505.23043*, 2025. 12
- [52] Chunting Zhou, Lili Yu, Arun Babu, Kushal Tirumala, Michihiro Yasunaga, Leonid Shamis, Jacob Kahn, Xuezhe Ma, Luke Zettlemoyer, and Omer Levy. Transfusion: Predict the next token and diffuse images with one multi-modal model. *arXiv preprint arxiv:2408.11039*, 2024. 13
- [53] Kai Zou, Ziqi Huang, Yuhao Dong, Shulin Tian, Dian Zheng, Hongbo Liu, Jingwen He, Bin Liu, Yu Qiao, and Ziwei Liu. Uni-mmmu: A massive multi-discipline multimodal unified benchmark. *arXiv preprint arXiv:2510.13759*, 2025. 1

A. Synthetic Data Using GPT4o

All experimental data were constructed using GPT-4o. Due to space limitations, we explicitly list the generation prompts for first-order difficulty Mathematical Operations in Table 6 and first-order difficulty Symbolic Mapping in Table 7 and Table 8; prompts for other data generation are available in our code.

B. Ablation Experiments

B.1. Ablation on Reject Sampling.

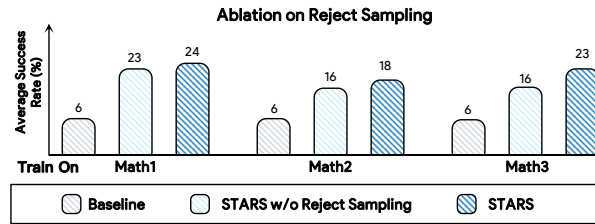


Figure 5. The average results of Bagel(normal) on Math for the ablation of Reject Sampling.

We conducted an ablation study to validate the importance of Rejection Sampling. For this, we compared our complete STARS framework against a variant trained on 5000 randomly-sampled, unfiltered CoT-generated instances for each difficulty tier. As shown in Figure 5, applying rejection sampling consistently improves performance across all levels. The effect is most pronounced on the complex Math3 dataset, where our method increases the success rate from 16% to 23%. This result underscores that the quality of data curated by rejection sampling is critical for effectively distilling reasoning capabilities via self-training.

B.2. Comparison of Curriculum Learning and Mixed Training

We also compared our curriculum learning strategy with a mixed training approach, where data from all difficulty levels were combined and trained together. The experimental results, as shown in Table 5, reveal that the model trained using the curriculum learning strategy achieved better results, proving the effectiveness of our framework.

C. Evaluation Using MLLM

To mitigate the biases associated with using MLLMs as direct visual judges, we adopted a two-stage evaluation strategy. First, we use an MLLM to generate a text caption for the image, and then we compare this caption against the ground-truth text answer. This approach is effective because our tasks require the model

to generate images that are simple in category and easily discernible (e.g., a specific number of objects or a single, clear subject), making them straightforward for the MLLM to identify and caption accurately. In our experiments, we uniformly used run Qwen2.5-VL-7B-Instruct by vLLM¹.

The prompts used for evaluating Mathematical Operations are listed in Table 9. The prompts for evaluating Symbolic Mapping are shown in Table 10. Finally, the prompts used to evaluate Knowledge Transfer are presented in Table 11.

D. Implementation Details of Knowledge Injection

We created ten virtual character profiles; the specific details are provided in Table 12.

For the Forward Retrieval, we performed knowledge injection for each character profile separately, starting from the pre-trained model weights. This process resulted in 10 new, fine-tuned models (one for each character).

For the Inverse Search, we adopted a pairwise approach. Each training session injected information about two characters, again starting from the original pre-trained weights. This resulted in 5 new models.

We created 40 text-only question-answering (QA) pairs for each character to fine-tune the model’s understanding module. To ensure the knowledge was successfully injected, we verified the fine-tuned models with text-based questions. A training run was only considered successful if the model could correctly answer all questions about the injected character attributes.

For the training of BAGEL, we used the official BAGEL codebase. The training parameters were: learning rate 4×10^{-5} ; expected_num_tokens 64; max_num_tokens 162; and training for 60 epochs. (Note: expected_num_tokens and max_num_tokens are parameters from the official BAGEL codebase²).

For the other models, we used the MS-Swift³ for fine-tuning. The parameters were: 30 epochs; batch size 64; and learning rate 1×10^{-5} .

All training was conducted on 8 A100 GPUs.

E. Related Works

Unified Multimodal Models (UMMs) [10, 15, 18, 28, 32, 34, 35, 39, 42, 45, 47–49, 51] aim to build a single system that can handle both understanding tasks (e.g., VQA, grounding) and generation tasks (e.g., text-to-image). Existing UMMs can be grouped into three

¹<https://github.com/vllm-project/vllm>

²<https://github.com/ByteDance-Seed/Bagel>

³<https://github.com/modelscope/ms-swift>

Table 5. Performance of BAGEL trained with STARS on symbolic mapping datasets of varying difficulties. M1–3 represent three increasing levels of difficulty for symbolic mapping. Normal and CoT represent evaluations without/with Chain-of-Thought, respectively. CL stands for Curriculum Learning, and Mix refers to training with a mix of data from all difficulty levels.

Model	Normal			CoT			Average
	M1	M2	M3	M1	M2	M3	
BAGEL	0.00	0.00	0.00	0.75	0.57	0.46	0.30
BAGEL + STARS with CL	0.64	0.46	0.27	0.75	0.65	0.50	0.55 (+0.25)
BAGEL + STARS with Mix	0.63	0.37	0.25	0.70	0.45	0.35	0.46 (+0.16)

Data Generation Task: First-level Difficulty Mathematical Operations	
Task Description:	Please generate 50 prompts in the following JSONL format:
Example Format:	<pre>{ "Question": "Produce a number of pencils equal to the result of 2 * 2.", "Answer": "4 pencils." }</pre>
Requirements:	<ol style="list-style-type: none"> Each Question must explicitly instruct the user to perform a basic arithmetic operation (addition, subtraction, multiplication, or division), with the final result strictly between 1 and 4 (inclusive). Use a diverse and natural set of expressions, such as: <ul style="list-style-type: none"> “Generate as many [objects] as the result of [expression].” “Produce a number of [objects] equal to the result of [expression].” “Show the number of [objects] that matches the outcome of [expression].” “Create the same quantity of [objects] as [expression] equals.” “Provide the same number of [objects] as calculated by [expression].” Replace [expression] with a valid arithmetic expression (e.g., $3 - 1$, $2 + 2$, $4 / 2$) that evaluates to 1, 2, 3, or 4. Use a wide variety of common objects (not just fruits). Include animals, toys, stationery, kitchen items, etc. Do not use rare or unusual items. In the Answer, the object name must be grammatically correct and match the number, e.g.: <ul style="list-style-type: none"> “1 eraser” “2 oranges” “3 kittens” “4 spoons” Output Format: <ul style="list-style-type: none"> Please return the 1000 generated items as individual JSON objects, one after another, not wrapped in a list or array. Do not include additional text, titles, or explanations.

Table 6. Prompt for First-level Mathematical Operations Data Generation.

types: (1) Autoregressive (AR), which usually employ a tokenizer to encode images into tokens like text, and then use a single transformer model to process them end-to-end via next-token prediction (e.g., Chameleon [30], EMU3 [33], Liquid [39], SynerGen-VL [19], Janus [36]); (2) AR + Diffusion (deep fusion), where one transformer processes both text and images, but diffusion is used to predict the image (e.g., Transfusion [52], Mogao [22], Bagel [7]); and (3) AR + Diffusion (shallow fusion), where a VLM is first used to extract hidden states as embeddings, which are then

mapped via an MLP into the image hidden dimension, and finally passed to a DiT for image generation (e.g., UniWorld-V1 [24], OmniGen2 [38], Qwen-Image [37]).

F. Details of Reject Sampling

We use BAGEL’s own understanding module to filter the generated images. The specific instruction prompt is in Table 13.

Data Generation Task: One-Step Symbol Mapping (Part 1)

Task Description: Please generate **50 pairs** of one-step symbol mapping data. Each pair should consist of two prompts: `Question_A` and `Question_B`. Each mapping rule involves two symbols, and each symbol represents a common object. Each pair of prompts must clearly state the symbol mapping rule and generate two prompts: `Question_A` and `Question_B`. The answers should correspond to the objects represented by the symbols in the rules.

In this version:

- `Question_A` will generate the object represented by the **first** symbol.
- `Question_B` will generate the object represented by the **second** symbol.

Each `Question_A` and `Question_B` must have a unique ID, and each prompt should have an `Answer` field. The answer should be the object represented by the corresponding symbol.

Output Format: The output should be in **JSONL** format, where each entry contains the following fields:

- **ID:** A unique identifier, starting from 1, with the same ID for both `Question_A` and `Question_B`.
- **Question_A:** The prompt that contains the symbol mapping rule and generates the object represented by the first symbol.
- **Question_B:** The prompt that contains the symbol mapping rule and generates the object represented by the second symbol.
- **Answer:** The corresponding answer for the prompt, which should be the object that the symbol represents.

Example Format:

```
{ "ID": "1", "Question_A": "Rule 1: The symbol @ represents apples. Rule 2: The symbol * represents bananas. Please generate the fruit represented by the symbol @.", "Answer": "Apples" }
{ "ID": "1", "Question_B": "Rule 1: The symbol @ represents apples. Rule 2: The symbol * represents bananas. Please generate the fruit represented by the symbol *.", "Answer": "Bananas" }
```

Table 7. **Prompt for Symbolic Mapping Data Generation (Part 1).** Overview of the task, logic definition, and output format requirements.

G. Limitation

UniSandbox is designed as a “stress test” to “isolate variables,” such as data leakage, and probe the model’s “pure” reasoning ability. While this controlled “sandbox” environment is highly effective for precise attribution analysis, it naturally means our synthetic reasoning tasks are simplified and do not capture the full complexity of real-world reasoning.

Similarly, while the STARS framework provides a successful proof-of-concept for internalizing reasoning, it is a preliminary exploration. Its current success relies on high-quality CoT-generated data, and its scalability to more diverse and complex reasoning domains requires further investigation.

Finally, due to resource constraints, our knowledge injection experiments were confined to a small, structured knowledge base. This controlled approach allowed us to clearly confirm the existence of the knowledge transfer bottleneck. How these findings translate to large-scale, unstructured knowledge remains an important open question.

We believe these limitations are also opportunities. Our work provides foundational insights and tools (UniSandbox) for future research to build upon, address-

ing these challenges to develop more truly unified multimodal models.

Data Generation Task: One-Step Symbol Mapping (Part 2)

Requirements:

1. **Symbols:** Symbols can be any common symbols, including letters (e.g., a, b, c, A, B, C), numbers (e.g., 1, 2, 3, 5), punctuation marks (e.g., @, #, *, &, \$, %, ^, etc.), and other characters. Symbols should not be limited to numbers or special characters but can include any alphanumeric or symbolic character.
2. **Object Mapping:** Each pair of data should map the symbols to common objects such as pencils, chairs, phones, refrigerators, notebooks, etc. The objects can be everyday items, not just fruits.
3. **Answer Consistency:** Ensure that the `Question_A` and `Question_B` are clearly structured, and that the Answer for each corresponds to the symbol mapping rule.
4. **Unique IDs:** Each pair of `Question_A` and `Question_B` should share the same ID.
5. **Object Diversity:** The objects in the answers should be varied and can include common items like fruits, stationery, household items, animals, etc.

Example Data:

```
{
  "ID": "1",
  "Question_A": "Rule 1: The symbol @ represents apples. Rule 2: The symbol * represents bananas. Please generate the fruit represented by the symbol @.",
  "Answer": "Apples"
},
{
  "ID": "1",
  "Question_B": "Rule 1: The symbol @ represents apples. Rule 2: The symbol * represents bananas. Please generate the fruit represented by the symbol *.",
  "Answer": "Bananas"
},
{
  "ID": "2",
  "Question_A": "Rule 1: The symbol # represents pencils. Rule 2: The symbol $ represents notebooks. Please generate the object represented by the symbol #.",
  "Answer": "Pencils"
},
{
  "ID": "2",
  "Question_B": "Rule 1: The symbol # represents pencils. Rule 2: The symbol $ represents notebooks. Please generate the object represented by the symbol $.",
  "Answer": "Notebooks"
},
{
  "ID": "3",
  "Question_A": "Rule 1: The symbol % represents chairs. Rule 2: The symbol ^ represents tables. Please generate the object represented by the symbol %.",
  "Answer": "Chairs"
},
{
  "ID": "3",
  "Question_B": "Rule 1: The symbol % represents chairs. Rule 2: The symbol ^ represents tables. Please generate the object represented by the symbol ^.",
  "Answer": "Tables"
},
{
  "ID": "4",
  "Question_A": "Rule 1: The symbol * represents apples. Rule 2: The symbol @ represents oranges. Please generate the fruit represented by the symbol *.",
  "Answer": "Apples"
},
{
  "ID": "4",
  "Question_B": "Rule 1: The symbol * represents apples. Rule 2: The symbol @ represents oranges. Please generate the fruit represented by the symbol @.",
  "Answer": "Oranges"
},
{
  "ID": "5",
  "Question_A": "Rule 1: The symbol $ represents pencils. Rule 2: The symbol # represents erasers. Please generate the object represented by the symbol $.",
  "Answer": "Pencils"
},
{
  "ID": "5",
  "Question_B": "Rule 1: The symbol $ represents pencils. Rule 2: The symbol # represents erasers. Please generate the object represented by the symbol #.",
  "Answer": "Erasers"
}
```

Guidelines:

1. **Symbols:** Feel free to use any common symbols, including alphanumeric characters (e.g., a, b, c, A, B, C), numbers (e.g., 1, 2, 3), and special characters (e.g., @, #, *, &, \$, %, ^, etc.).
2. **Objects:** Objects should be common, everyday items such as stationery, fruits, animals, etc.
3. **Output:** Ensure the JSONL format is strictly followed, and each data entry is separate.
4. **Answer Consistency:** Each symbol must be mapped to an appropriate common object, and the Answer should reflect this mapping.

Please generate the data based on these instructions.

Table 8. **Prompt for Symbolic Mapping Data Generation (Part 2).** Detailed constraints, few-shot examples, and final guidelines.

Prompt 1: Caption Generation Expert
<p>System Instruction: You are an extremely rigorous and neutral image captioning expert. Your task is to carefully examine a given image and generate a precise, concise caption (under 20 characters) that strictly and unambiguously describes the count and type of all clearly, fully, and normally visible objects in the image.</p> <p>Instructions:</p> <ol style="list-style-type: none"> 1. Only count a single object type: The image must contain only one clearly identifiable type of object. If multiple distinct object types are present, the task is considered an error. 2. Only include objects that are: <ul style="list-style-type: none"> • Fully visible • Clearly identifiable • Normal in shape, scale, and orientation • Not distorted, occluded, cropped, or ambiguous 3. Strict Error Handling: If the image contains more than one type of object, or if any visible object is distorted or has its geometry compromised, the task is considered an error. 4. Your caption must be in the format: <ul style="list-style-type: none"> • [Number] [object-type-plural] e.g., “3 apples”, “2 chairs” 5. Condition Check: If the conditions in rules 1-3 are not met, you must output ‘Error’. Do not provide any other explanations. <p>Output Format: Caption: [your output here]</p>
Prompt 2: Caption Evaluator
<p>System Instruction: You are a rigorous and fair caption evaluator. Your task is to compare a generated image caption with an expected answer, and determine whether they match perfectly and exclusively in count and object meaning.</p> <p>Evaluation Rules:</p> <ol style="list-style-type: none"> 1. The caption must match the expected answer in: <ul style="list-style-type: none"> • Count: The number of each object type must be exactly the same. • Object Type: The object types in both descriptions must be exactly the same. 2. If any object type in the caption does not match the expected answer, or if the count is wrong, or if there are extra object types, return: Score: NO 3. If and only if the object types and counts in the caption exactly match the expected answer, return: Score: YES <p>Input:</p> <ul style="list-style-type: none"> • Generated Caption: {caption} • Expected Answer: {expected.answer} <p>Output Format: Score: [YES or NO]</p>

Table 9. **Prompts for Mathematical Operations Evaluation.** The first prompt instructs the model to generate rigorous captions focusing on object count and type, while the second prompt evaluates the generated caption against the ground truth.

Prompt 1: Caption Generation Expert
<p>System Instruction: You are an extremely rigorous and neutral image captioning expert. Your task is to carefully examine a given image and generate a precise, concise caption (under 20 characters) that strictly and unambiguously describes the count and type of all clearly, fully, and normally visible objects in the image.</p> <p>Instructions:</p> <ol style="list-style-type: none"> 1. Only count a single object type: The image must contain only one clearly identifiable type of object. If multiple distinct object types are present, the task is considered an error. 2. Only include objects that are: <ul style="list-style-type: none"> • Fully visible • Clearly identifiable • Normal in shape, scale, and orientation • Not distorted, occluded, cropped, or ambiguous 3. Strict Error Handling: The task is considered an error if any of the following conditions are met: <ul style="list-style-type: none"> • The image contains more than one type of object. • Any visible object is distorted or has its geometry compromised. • The image contains any letters (e.g., A, B, C), symbols (e.g., punctuation marks, mathematical operators like +, -, #, \$), or numbers (e.g., 1, 2, 3). 4. Your caption must be in the format: <ul style="list-style-type: none"> • [Number] [object_type_plural] e.g., “3 apples”, “2 chairs” 5. Condition Check: If the conditions in rules 1-3 are not met, you must output ‘Error’. Do not provide any other explanations. <p>Output Format: Caption: [your output here]</p>
Prompt 2: Caption Evaluator
<p>System Instruction: You are a rigorous and fair caption evaluator. Your task is to compare a generated image caption with an expected answer, and determine whether they match perfectly and exclusively in count and object meaning.</p> <p>Evaluation Rules:</p> <ol style="list-style-type: none"> 1. The caption must match the expected answer in: <ul style="list-style-type: none"> • Count: The number of each object type must be exactly the same. • Object Type: The object types in both descriptions must be exactly the same. 2. If any object type in the caption does not match the expected answer, or if the count is wrong, or if there are extra object types, return: Score: NO 3. If and only if the object types and counts in the caption exactly match the expected answer, return: Score: YES <p>Input:</p> <ul style="list-style-type: none"> • Generated Caption: {caption} • Expected Answer: {expected_answer} <p>Output Format: Score: [YES or NO]</p>

Table 10. **Prompts for Symbol Mapping Evaluation.** The first prompt instructs the model to generate rigorous captions focusing on object type, while the second prompt evaluates the generated caption against the ground truth.

<p>Prompt 1:</p> <p>System Instruction: You are a hyper-precise and cautious vision assistant. Task: Produce a single-line caption for the given image.</p> <p>Rules:</p> <ol style="list-style-type: none"> If any person is visible, output strictly in this format: <ul style="list-style-type: none"> Person: <skin_tone_or_ethnicity>; <hair_color>; <age_group: kid mid-age old>; <gender: male female unclear> Crucial: The <skin_tone_or_ethnicity> slot MUST be one of these three options exactly: African/Indigenous, Caucasian, East Asian. If the person does not clearly fit one of these categories or you are uncertain, output ‘unclear’ for that slot. If uncertain about other attributes (hair, age, gender), output ‘unclear’ for that specific slot. If no person is visible, you MUST follow these steps: <ol style="list-style-type: none"> First, determine if the primary subject is a <i>single type</i> of flower OR a <i>single type</i> of fruit. If it IS a flower, you MUST output its <i>specific</i> type/species in this format: Flower: <specific_type_name> (e.g., ‘Flower: rose’, ‘Flower: tulip’). If it IS a fruit, you MUST output its <i>specific</i> type/species in this format: Fruit: <specific_type_name> (e.g., ‘Fruit: apple’, ‘Fruit: banana’). ABSOLUTE RULE: General categories like ‘Flower: flower’ or ‘Fruit: fruit’ are FORBIDDEN. REJECTION: You MUST output exactly ‘Reject’ if ANY of the following are true: <ol style="list-style-type: none"> The image contains no person, AND the primary subject is NOT a flower or a fruit (e.g., it is a car, dog, building, etc.). (Per Rule 2) The image IS a flower or fruit, but you cannot confidently identify its <i>specific type</i> (e.g., you can only tell it’s a ‘fruit’, not an ‘apple’). In this case, you MUST output ‘Reject’. The image is distorted, unrealistic, surreal (e.g., face on fruit), or generally ambiguous. The image contains multiple different types of flowers or multiple different types of fruits. (Per Rule 1) The person’s ethnicity is visible but does not fit the three required categories. Output must be ONE line with no extra words, no explanations.
<p>Prompt 2: Careful Evaluator</p> <p>System Instruction: You are a careful evaluator. Determine if a generated caption and a ground truth (GT) match semantically.</p> <p>Strict rules:</p> <ol style="list-style-type: none"> If the generated caption is exactly ‘Reject’, it is automatically incorrect. Respond with ‘Score: NO’. If the generated caption starts with “Person:”, it contains 4 slots: skin/ethnicity; hair color; age group (kid—mid-age—old); gender (male—female—unclear). <ul style="list-style-type: none"> The GT must be a sentence describing a person (e.g., ‘An elderly African/Indigenous woman with black hair.’). Compare each slot against the GT description. Allow common synonyms (e.g., blond=blonde; elderly/senior=old). The generated ethnicity MUST be one of [African/Indigenous, Caucasian, East Asian] or ‘unclear’. If the generated caption starts with “Flower:” or “Fruit:”, compare the specific object type with the GT. <ul style="list-style-type: none"> The GT must be a specific type (e.g., ‘carnation’, ‘apple’). Crucial: The generated caption <i>format</i> (Flower:/Fruit:) AND the <i>specific type</i> must BOTH match the GT. Singular/Plural forms ARE a match. (e.g., ‘apple’ matches ‘apples’; ‘peach’ matches ‘peaches’). Only semantic equivalents are a match (e.g., cup ≈ mug). Example 1 (Match): Generated ‘Flower: rose’ and GT ‘rose’ is ‘Score: YES’. Example 2 (Match): Generated ‘Fruit: apple’ and GT ‘apple’ is ‘Score: YES’. Example 3 (Match - Plural): Generated ‘Fruit: apple’ and GT ‘apples’ is ‘Score: YES’. Example 4 (Match - Plural): Generated ‘Fruit: peach’ and GT ‘peaches’ is ‘Score: YES’. Example 5 (Mismatch - Wrong Type): Generated ‘Flower: rose’ and GT ‘carnation’ is ‘Score: NO’. Example 6 (Mismatch - General Term): Generated ‘Flower: flower’ and GT ‘carnation’ is ‘Score: NO’. Example 7 (Mismatch - Wrong Category): Generated ‘Fruit: rose’ and GT ‘rose’ is ‘Score: NO’. Example 8 (Mismatch - Wrong Category): Generated ‘Flower: apple’ and GT ‘apple’ is ‘Score: NO’. If formats fundamentally differ (e.g., generated caption starts with ‘Person:’ while GT is ‘apple’), respond with ‘Score: NO’. <p>Input: Generated: {caption} — GroundTruth: {gt}</p> <p>Output format (MUST be exact): Score: YES or Score: NO</p>

Table 11. **Prompts for Knowledge Transfer Evaluation.** The first prompt instructs the model to strictly categorize and identify persons, flowers, or fruits with specific formats. The second prompt evaluates the generated output against the ground truth.

Table 12. Profiles of Ten Fictional Characters for Large Language Model Knowledge Injection.

Name	Gender	Age	Hair Color	Skin Color	Favorite Fruit	Favorite Flower
Lysendria	Female	old	black	African/Indigenous	apples	carnation
Kaelorix	Male	kid	blond	Caucasian	strawberries	sunflower
Jovianne	Female	middle-aged	brown	East Asian	oranges	lily
Zefyria	old	Male	white	Caucasian	bananas	rose
Aurelius	Female	kid	black	African/Indigenous	grapes	tulip
Nyxella	Male	middle-aged	black	African/Indigenous	peaches	daisy
Valerian	Female	middle-aged	brown	African/Indigenous	watermelon	orchid
Thalassia	Male	kid	brown	East Asian	apples	rose
Orionax	Female	middle-aged	black	Caucasian	oranges	carnation
Evandriel	Male	kid	red	Caucasian	bananas	sunflower

Prompt: Strict Image Quality Filter
<p>System Instruction: You are an expert image quality assessor. Your task is to evaluate an image based on a specific question. You must be extremely strict and analytical in your evaluation.</p> <p>Question: {question}</p> <p>Please follow these steps exactly:</p> <ol style="list-style-type: none"> Analyze the Question: <ul style="list-style-type: none"> First, identify the type of object(s) mentioned in the question (e.g., desks, chairs, cats, etc.). Next, determine the correct number of objects required by the question. If the question contains a mathematical expression (e.g., '8 / 4', '5 - 3'), you MUST perform the calculation first to get the correct number. Examine the Image and Count Objects: <ul style="list-style-type: none"> Carefully examine the image and count the number of objects of the identified type. Describe the objects you see and state the actual count. Perform a Strict Criteria Check: <ul style="list-style-type: none"> Based on your analysis and count, check if the image meets ALL of the following criteria. Be very strict. Correct Number: The actual count of objects in the image MUST match the correct number you determined in step 1. Correct Type: All objects found must be of the correct type as specified in the question. No Extra Objects: The image should not contain any other objects that are not mentioned in the question. Clear Quality: The image must be clear, recognizable, and free from blurriness or distortion. Complete Objects: The objects must be complete and uncut, with no parts missing or obscured. Final Judgment: <ul style="list-style-type: none"> After completing the checks in step 3, provide your final judgment using ONLY one of the two formats below. Do not add any extra text or explanation. If all criteria are met: <code>Final Answer: YES</code> If even ONE criterion is NOT met: <code>Final Answer: NO</code> <p>Think step by step and be very strict in your evaluation.</p>

Table 13. **Prompt for Reject Sampling.** This prompt is used to rigorously filter generated images by verifying object counts, types, and visual quality against the input prompt.