Symbolic Representation for Any-to-Any Generative Tasks

Anonymous CVPR submission

Paper ID 3011

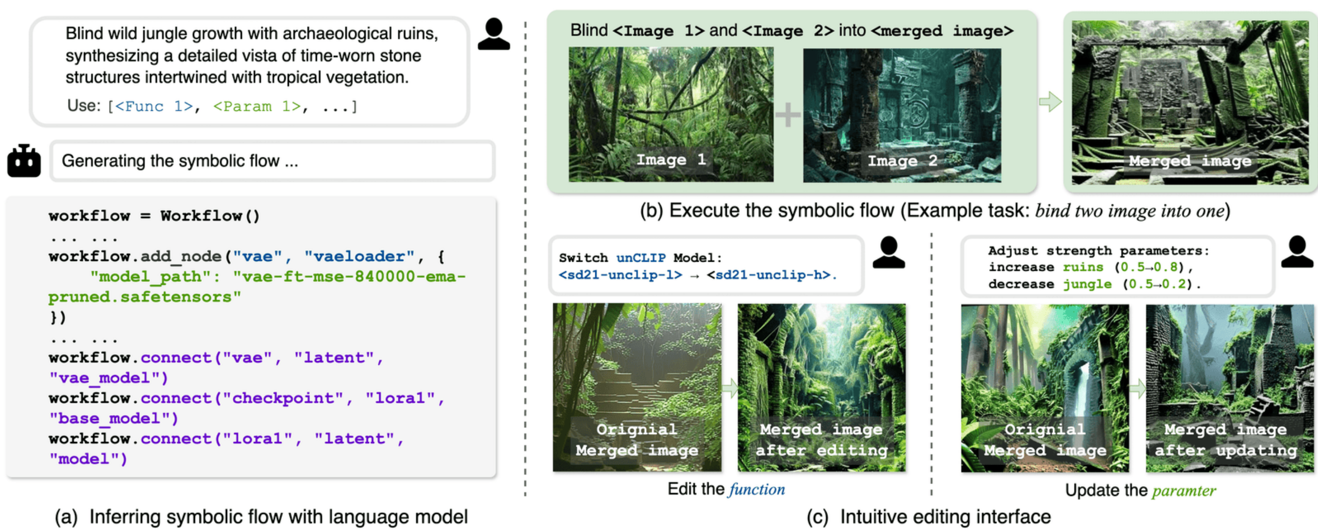

Figure 1. **A symbolic representation for *Any-to-Any* generative tasks.** (a) We develop a training-free inference engine that transforms natural language task descriptions into executable symbolic flow comprising *functions*, *parameters*, and the *topology*. (b) The symbolic flow allows executing generative tasks as programs. Example task is mentioned in the first sentence of Sec. 1 (c) Both *functions* and *parameters* can be easily modified to customize the generation process and the output style.

Abstract

We propose a symbolic generative task descriptive language and inference engine, capable of representing arbitrary multimodal tasks as symbolic flows. The inference engine maps natural language instructions to symbolic flow, eliminating the need for task-specific training. Conventional generative models rely heavily on large-scale training and implicit neural representation to learn cross-modal mappings, which demands extensive computational resources and restricts expandability. In this paper, we propose an explicit symbolic task descriptive language, comprising three types of primitives: functions, parameters, and topological logic. Using a pre-trained language model to infer symbolic workflows in a training-free manner, our framework successfully performs over 12 *multimodal generative tasks based on user instructions, demonstrating enhanced efficiency and flexibility. Extensive experiments demonstrate that our approach can generate multimodal content competitive with, and often surpassing, that of previous state-of-the-art unified models, while offering robust interruptibility and editability. We believe that symbolic task representations are capable of cost-effectively expanding the boundaries of generative AI capabilities. All code and results are available in the Supplementary Materials.*

1. Introduction

"*Blending the wild growth of a jungle with the mystique of ancient ruins into a brand-new scene would be stunning*," your artist friend mused. "*And if we could transform the photographic image into a video, overlayed with my audio recording of birds chirping and the soft murmur of flowing water—it would create a truly dreamlike sensory experience.*" This raises an interesting question:

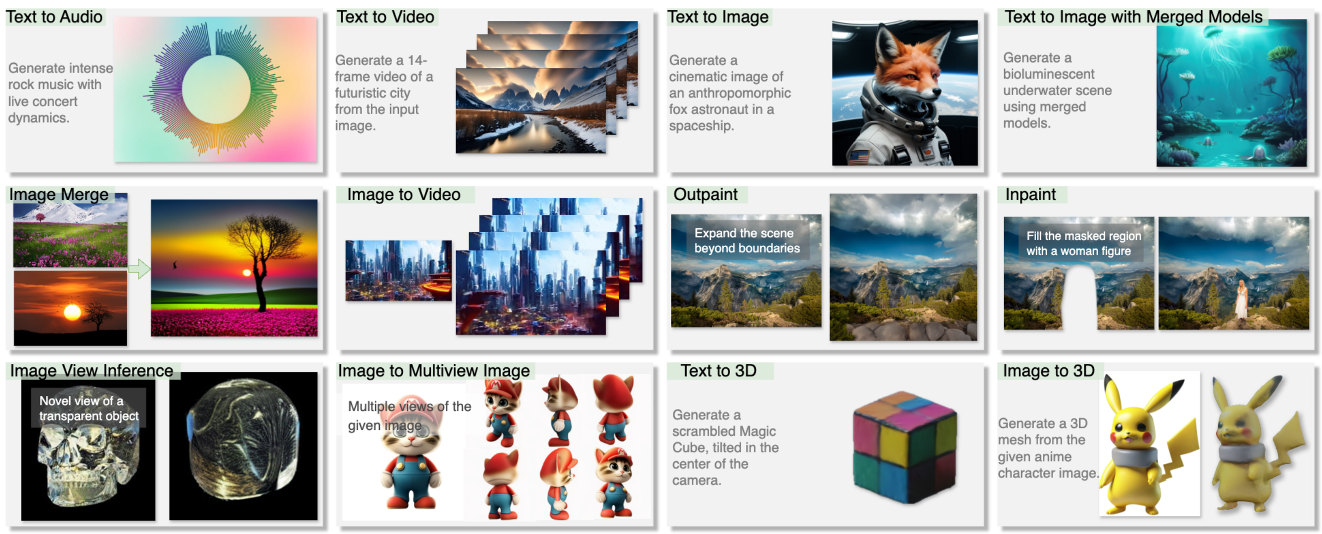

Figure 2. **The *Any-to-Any* generative model.** Our model demonstrates the capability to handle **any-to-any generative tasks** across various modalities, including text, images, videos, audio, and 3D content. It supports flexible transformations such as converting image to video, generating 3D models from images, or synthesizing audio from textual prompts. Formally, any-to-any generative tasks refer to generating outputs in any desired modality from inputs in any other modality, all guided by natural language instructions [42]

.

how can we design a *unified model* capable of seamlessly handling generative tasks across any combination of input and output modalities ("***any-to-any*** generative tasks", as shown in Figure 2), guided by natural language instructions [12, 25, 26, 42, 49]? The workflow for executing this task comprises several essential processes [12, 39, 49]. First, the system imports two images and encodes them to extract their latent features. Then, taking these features as conditioning inputs, it combines them based on the user-specified blending strength and re-synthesizes the blended latent representation onto a blank latent canvas. Finally, the system decodes this latent representation into a viewable image.

Current approaches for any-to-any generative tasks typically fall into two paradigms: *Implicit neural modeling* and *agaentic approaches*. Implicit neural modeling approaches directly learn a neural representation from mass training data [25, 26, 26, 31, 40, 41, 54]. While offering simplicity in representing multimodal information, their extensibility is constrained by the scope of the training data. They struggle to handle rare or unanticipated tasks—such as the image blending example in Figure 1, if such cases are not accounted for during training. Moreover, their reliance on implicit neural representations makes them non-interruptible, leaving them ill-equipped to manage complex, multi-step workflows. Agentic approaches rely on sophisticated multi-agent coordination and tool orchestration [12, 13, 27, 33, 38, 39], which introduces system instability and operational overhead in their decision-making process. While powerful, these approaches lack a unified formal representation of tasks and fail to capture their inherent compositional nature. Our experiments reveal that complex agent designs do not necessarily outperform simpler ones, motivating us to explore an alternative direction: focusing on ***unified task representations*** and ***language model-friendly interfaces*** that enable direct task specification.

Examining the image-blending example reveals three fundamental components essential for executing generative tasks. At its core are distinct ***functions*** -computational operations such as image encoding, conditioning, and blending that transform inputs into desired outputs. Each function's behavior is shaped by ***parameters***, such as the blending strength and re-synthesis intensity, which fine-tune the operation to meet specific requirements. These functions do not operate in isolation; their ***topology***, or interconnected relationships, form a cohesive workflow that guides the progression from input to output. These three components, functions, parameters, and topology, together enable the effective execution of complex generative tasks. Based on these insights, we propose $\mathcal{A}$-LANGUAGE, a formal representation that systematically captures these three essential components of generative tasks. In $\mathcal{A}$-LANGUAGE, ***function*** specifies the core computational operations, enabling the system to precisely identify and execute required transformations. ***parameter*** provides fine-grained control over each operation's behavior, allowing users to adapt functions to specific task requirements. ***topology*** formalizes the workflow structure, defining how functions interact and combine to accomplish complex generative goals. Through this three-component abstraction, $\mathcal{A}$-LANGUAGE enables flexible yet structured orchestration of generative tasks.

Alongside the symbolic generative task language, we introduce a ***training-free inference engine*** that utilizes a pre-trained language model (LM) as its foundation to derive a symbolic representation from input instructions and a designated key function. Initially, the pre-trained LM identifies a comprehensive function set and parameter set from the natural language instruction, forming an initial functional and parametric structure. With this set of functions, we then predict the topology, outlining the dependencies among functions to form the complete symbolic representation. We also implement a refinement module, an iterative process activated upon any inference failure, enabling immediate corrections to resolve issues. Together, the $\mathcal{A}$-LANGUAGE, the inference engine, and the refinement module led to a high-quality system that provides flexible and precise workflow-building capabilities.

Experimentally, we constructed a dataset of 120 real-world generative tasks spanning 12 task categories and validated the effectiveness of our approach through user studies and executability evaluations. The results demonstrate that our symbolic model is competitive with or outperforms state-of-the-art multimodal generative models in task generalization, output quality, and editing flexibility. Additionally, our experiments investigated the impact of syntax choices on the quality of symbolic flow generated by LMs. Our contributions are three-fold:

- A unified symbolic representation, the $\mathcal{A}$-LANGUAGE, that systematically decomposes **any** generative task into three core components: ***function*** for atomic operations, ***parameter*** for behavioral control, and ***topology*** for symbolic flow structure.
- A ***training-free inference engine*** that leverages pre-trained LMs to automatically convert natural language instructions into symbolic representations for executable workflows.
- Empirical validation demonstrates that it excels in generalizability, modifiability, and providing an exceptional user experience.

2. Related work

2.1. Unified multi-modal framework

Recent years have witnessed remarkable advances in large language models (LLMs), which have demonstrated exceptional capabilities across various natural language tasks, from basic comprehension to complex reasoning [3, 6–8, 16, 21, 24, 29–31, 43, 44]. Building on this success, multimodal large language models (MLLMs) have extended these capabilities to integrate multiple forms of input and output, covering data modalities such as images, audio, video, and 3D structures [1, 4, 5, 10, 14, 18–20, 22, 32, 34–37, 46, 47, 50–53, 55]. The field has progressed from isolated single-modality models to sophisticated any-to-any frameworks [25, 26, 28, 31, 40, 41, 54] that can handle diverse input-output combinations within a single model architecture. However, these unified multimodal frameworks face significant challenges in practice. The scarcity of high-quality, diverse multimodal datasets remains a fundamental bottleneck, particularly for complex cross-modal tasks. Moreover, different modalities often require distinct processing approaches and representations, making it challenging to achieve optimal performance across all possible modality combinations in a single model. The need to align disparate modalities into a coherent unified representation while preserving their unique characteristics continues to be a core challenge in advancing these frameworks.

2.2. Workflow synthesis

Workflow synthesis [2, 15, 17] seeks to generate executable sequences of operations for complex tasks by coordinating AI models and resources, particularly in generative AI, where tasks often require sophisticated combinations of inference, parameters, and logic. Traditional methods using neural modules or predefined operations struggle with the open-ended nature of modern AI tasks. Recent advances like HuggingGPT [39] leverage large language models for task planning and model coordination, VISPROG [12] employs neuro-symbolic approaches for programmatic task decomposition, and GenAgent [49] uses multi-agent collaboration to build workflows step by step. Despite their differences, these approaches highlight the need for flexible, interpretable representations. Our work advances this field by proposing a unified symbolic framework for describing and executing generative tasks, balancing expressiveness and practicality.

3. $\mathcal{A}$-Language

We introduce $\mathcal{A}$-LANGUAGE, a symbolic representation that bridges the gap between natural language task descriptions and executable workflows for any-to-any generative tasks. Unlike previous unified multimodal approaches dependent on ***implicit neural representations*** and ***intensive training***, our $\mathcal{A}$-LANGUAGE provides an ***explicit symbolic representation***, allowing a ***training-free*** execution.

3.1. Formulation

Fundamentally, $\mathcal{A}$-LANGUAGE formalizes any generative task t as a triple:

$$\Omega(t) := (\mathcal{F}, \Phi, \mathcal{T}).$$

This unified formulation decomposes any generative task into its essential constituents: the computational ***functions*** $\mathcal{F}$, their corresponding ***parameters*** Φ, and the ***topological***

structure $\mathcal{T}$ that elucidates their interrelations and data flow dynamics.

Function The function set is defined as $\mathcal{F} = \{f_1, f_2, ..., f_n\}$, where $n \in \mathbb{N}$, which represents atomic computational units. Each function takes both input data and parameters to produce outputs, formally defined as:

$$f_i : \mathcal{I}_i \times \phi_i \to \mathcal{O}_i,$$

where $\mathcal{I}_i$ defines its input space, ϕ_i represents its parameter configuration, and $\mathcal{O}_i$ specifies its output space. The input and output spaces $\mathcal{I}_i$ and $\mathcal{O}_i$ represent either simple scalar values or composite data structures of arbitrary modalities, allowing functions to process multiple inputs and generate multiple outputs. For example, an image blending function might accept two image inputs and produce both a blended result and an attention mask. When functions are connected, their inputs and outputs can be partially mapped, providing flexibility in constructing complex paths.

Parameter The parameter space $\Phi = \{\phi_{f_1}, \phi_{f_2}, ..., \phi_{f_n}\}$ encompasses configurations that modify function behaviors, where each ϕ_{f_i} represents the parameter space for function f_i. Parameters must be fully specified before function execution to ensure deterministic behavior. The parameter space is independent of the input space, enabling functions to exhibit different behaviors while processing identical inputs.

Topology The topology set $\mathcal{T} = \{d_1, d_2, ..., d_m\}$ defines the precise data flows between functions, where each d_k at the finest granularity specifies a single directed connection from a specific output of one function to a specific input of another function. Specifically, d_k is defined as a tuple representing an individual data flow from the output of a source function to the input of a target function. Formally:

$$d_k = (f_j, y_j) \to (f_i, x_i) \mid y_j \in \mathcal{O}_j, x_i \in \mathcal{I}_i$$

where f_j and f_i denote the source and target functions, respectively. y_j refers to a specific output produced by function f_j, while x_i corresponds to a specific input required by function f_i. Thus, each d_k encapsulates the transfer of data from a designated output of one function to a designated input of another, allowing for precise tracking of data flow through the system.

Symbolic flow The symbolic flow emerges from the interaction of ***functions***, ***parameters***, and ***topological logic***, formalizing the complete generative process:

$$\mathcal{S} = \{(f_i, \phi_{f_i}, D_i) \mid f_i \in \mathcal{F}\},$$

where D_i is the set of all data flows d_k in $\mathcal{T}$ that target function f_i:

$$D_i = \{(f_j, y_j) \to (f_i, x_i) \mid f_j \in \mathcal{F}, y_j \in \mathcal{O}_j, x_i \in \mathcal{I}_i\}.$$

Each element in the symbolic flow specifies a function, its parameter configuration, and its incoming directed connections. Specifically, for each function f_i, D_i contains tuples that map specific outputs of predecessor functions to specific inputs of f_i. This fine-grained formulation captures how computation progresses through the system, with functions receiving their required inputs from designated outputs of antecedent functions and parameter configurations from the parameter space. Through this unified and detailed representation, $\mathcal{A}$-LANGUAGE can express diverse and complex generative tasks. m

3.2. Syntax styles

The symbolic representation $\Omega(t)$ can be expressed through multiple syntactic styles, as shown in Figure 3, each offering different trade-offs in expressiveness and clarity. To identify the most effective representation for large language model inference, we explore three distinct syntactic formulations: ***declarative***, ***dataflow***, and ***pseudo-natural*** syntax, as illustrated through concise examples in Figure 3.

Declarative Syntax Declarative Syntax [45] focuses on explicitly specifying computational components and their relationships. Functions are separately declared with parameters, while connections are specified through explicit statements. This style is effective for complex workflows with reusable components, as it clearly separates component definitions ($\mathcal{F}$) from relationships ($\mathcal{T}$).

Dataflow syntax Dataflow syntax [49] emphasizes the flow of data through function compositions, where outputs directly feed into subsequent functions. It captures topological relationships ($\mathcal{T}$) through the order of function calls while maintaining explicit parameter specifications (Φ). This style is particularly suited for linear, sequential workflows.

Pseudo-natural syntax Pseudo-natural syntax [9] aims to bridge formal representations with more intuitive, language-like structures, making task specifications more accessible while maintaining mathematical rigor. This style explores a balance between precision and readability.

Each style retains the full expressiveness of $\Omega(t)$, but offers different advantages in terms of clarity and usability. The subsequent empirical analysis will evaluate which syntax best supports natural language inference while preserving necessary formal properties.

Notation	Implementation and definition
System Components	
$\mathcal{X}$	`List[Any]` // Input data of any modality
s	`str` // Task description
$\mathcal{C}$	`Dict` // System constraints
$\Omega(t)$	`Workflow` // Complete workflow representation
Workflow Structure	
$f_i \in \mathcal{F}$	`Node` // Computational function
$f_i : \mathcal{I}_i \times \phi_i \to \mathcal{O}_i$	`Node.forward` // Function mapping with parameters
$\phi_{f_i} \in \Phi$	`Dict[str, Any]` // Function parameters
$d_k \in \mathcal{T}$	`(Node, Any)-> (Node, Any)` // Source output to target input mapping $((f_j, y_j) \to (f_i, x_i))$
Workflow Operations	*(Declarative syntax, simplified version)*
Initialize	`Workflow()` // Create empty workflow $\Omega(t) = (\mathcal{F}, \Phi, \mathcal{T})$
Add Node	`add_node(name, type, params)` // Add function f_i with parameters ϕ_{f_i}
Connect	`connect(src_node, src_output, dst_node, dst_input)` // Create topology $d_k : (f_j, y_j) \to (f_i, x_i)$

Table 1. **System components and operations summary.** A comprehensive overview of $\mathcal{A}$-LANGUAGE's system components and their implementations. The upper two sections define the mathematical notations and their corresponding implementations, where the system processes input data $\mathcal{X}$ according to task description s under constraints $\mathcal{C}$. Functions f_i transform inputs $\mathcal{I}_i$ with parameters ϕ_i to outputs $\mathcal{O}_i$, and are connected through directed mappings d_k. The lower section demonstrates the Declarative Syntax as one example of workflow construction, showing how basic operations map to the mathematical formulation $\Omega(t) = (\mathcal{F}, \Phi, \mathcal{T})$.

```
workflow = Workflow()
... ...
workflow.add_node("vae",
"vaeloader", {
    "model_path": "vae-
ft-mse-840000-ema-
pruned.safetensors"
})
... ...
workflow.connect("vae",
"latent", "vae_model")
```

(a) Declarative Syntax

```
vae = vaeloader(
  model_path="vae-ft-
mse-840000-ema-
pruned.safetensors
)
... ...
vae = vae_model(
  latent
)
```

(b) Dataflow Syntax

```
vae is vaeloader with
the parameter of
(model_path is "vae-
ft-mse-840000-ema-
pruned.safetensors)
... ...
vae is vae_model with
the parameter of
(latent)
```

(c) Pseudo-natural Syntax

Figure 3. **Syntax comparison.** We implement our symbolic representation using three different styles of domain-specific languages (DSLs). (a) The declarative syntax registers all components into the workflow. (b) The dataflow syntax emphasizes the direction of data flow. (c) The pseudo-natural syntax mimics human language expression.

4. Inferring via pre-trained language model

The diversity and complexity of generative tasks necessitate a flexible and robust approach to transforming high-level task specifications into executable symbolic flows. As illustrated in Figure 4, we propose utilizing LMs as inference engines to generate task-specific symbolic representations, with Figure 5 demonstrating the complete pipeline from natural language description to executable workflow. This enables any-to-any transformations across different modalities and task types.

Given a set of inputs $\mathcal{X}$ of arbitrary modalities, a task description s, and a set of constraints $\mathcal{C}$, our inference framework generates a complete symbolic representation $\Omega(t)$. As illustrated in Figure 4, our framework leverages a pre-trained language model to infer both the computational components and their topology from natural language descriptions. This process can be formalized as:

$$\mathcal{M} : (\mathcal{X}, s, \mathcal{C}) \to \Omega(t),$$

where $\mathcal{X}$ represents any combination of inputs such as images, text, audio, or other modalities, s describes the desired transformation, and $\mathcal{C}$ represents a set of constraints, which typically specifying information such as available functions, specific parameter choices, valid parameter ranges, and model compatibility. These constraints are essential for ensuring that the generated symbolic flow is not only theoretically sound but also practically executable within the given computational environment. Specifically, we divide the inference into three main steps:

Component inference The first stage of our framework focuses on determining the necessary computational components. Given the input specifications and constraints, the LM identifies the required functions and their parameters:

$$\psi_1 : (\mathcal{X}, s, \mathcal{C}) \to (\mathcal{F}, \Phi).$$

This process accounts for both the explicit requirements of the task and any implicit dependencies, ensuring that selected functions are available within $\mathcal{C}$.

Topology construction The second stage focuses on establishing relationships between the identified components to form a coherent computational flow:

$$\psi_2 : (\mathcal{X}, s, \mathcal{C}, \mathcal{F}, \Phi) \to \mathcal{T}.$$

In this phase, the LM evaluates how the outputs of one function can serve as inputs to another, ensuring that these connections are executable and comply with the constraints defined in $\mathcal{C}$. This construction guarantees that data flows seamlessly through the system in a manner consistent with our unified formulation.

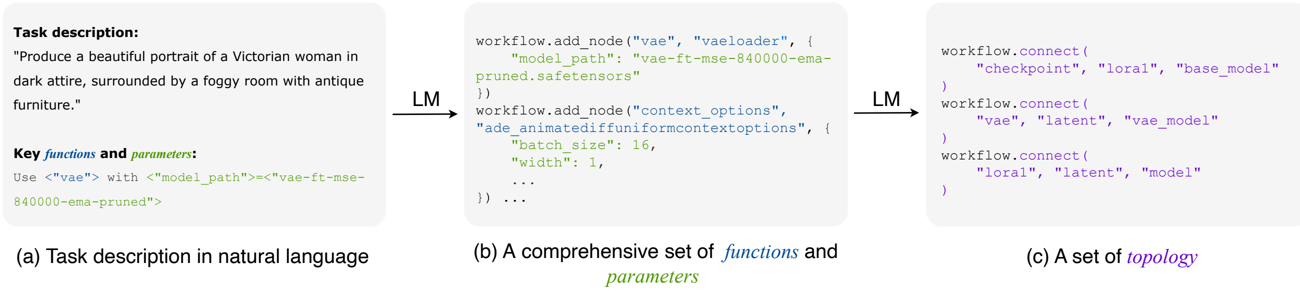

Figure 4. **Inferring symbolic flow with pre-trained language model (LM).** Beginning with (a) a natural language task description and key functions and parameters, we leverage LM to infer (b) a comprehensive set of functions and parameters. We then integrate (a) and (b) to deduce the (c) topology. If compilation or execution fails, all information is aggregated for further refinement (Sec. 4).

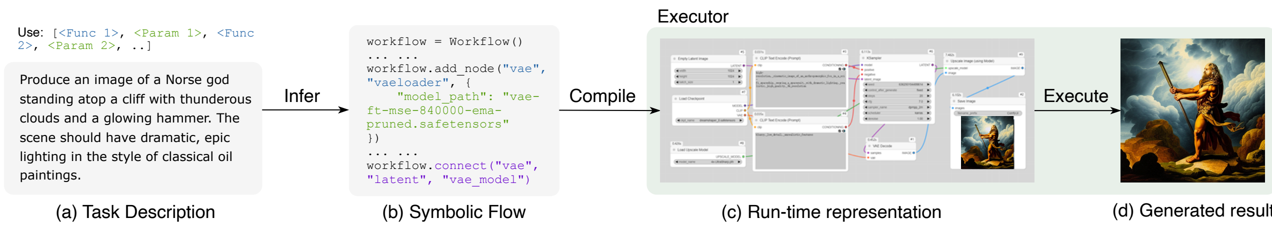

Figure 5. **Demonstration of the inference and execution.** The inference framework translates a natural language task description into an executable symbolic representation. This symbolic representation is then compiled and executed through a workflow executor to perform the desired transformation. See appendix for details.

Iterative refinement The generated symbolic flow undergoes an iterative refinement process to ensure correctness and executability. We define this refinement as:

$$\Omega_{i+1}(t) = R(\Omega_i(t), \epsilon_i),$$

where R represents the refinement operator and ϵ_i captures any detected issues in iteration i. To prevent endless loops, a maximum number of iterations can be set. During each iteration, the LM analyzes error signals and adjusts the symbolic flow accordingly, either by modifying function parameters, adding missing components, or restructuring topological connections. This iterative process continues until a valid symbolic flow is achieved that satisfies all constraints in $\mathcal{C}$ or the maximum iteration count is reached.

The combination of LM-based inference and iterative refinement enables our framework to handle diverse transformation tasks while maintaining robustness and generality. By leveraging the LM's reasoning capabilities and incorporating explicit constraints, we bridge the gap between high-level task descriptions and executable symbolic flows, providing a flexible foundation for any-to-any transformations.

5. Experiments

5.1. Setup

Prompt suite We collected a diverse set of 120 generative tasks from real-world applications to comprehensively evaluate our approach (see Appendix for the complete task list). These tasks are categorized into 12 general groups, each comprising 10 distinct instances. See Appendix for details.

Table 2. **Comparison of the average rankings** between outcome quality and task-outcome alignment rankings (↓). We primarily compared ***neural representing, training-dependent modeling*** [11, 23, 26, 48] and our ***symbolic representing, training-free modeling***. Each method was ranked on a scale starting from 1, with 1 denoting the best-performing approach. "U-IO 2" denotes "Unified-IO", "I-2-3D" denotes "Image to 3D Mesh", "T2M" denotes "Text to Mesh".

Method	Inpaint	Outpaint	Img merge	NVS	Merge model	I-2-3D
Show-o [48]	1.6	**1.4**	✗	✗	✗	✗
SEED-X [11]	✗	✗	**1.2**	✗	✗	✗
LWM [23]	✗	✗	✗	✗	✗	✗
U-IO 2 [26]	-	✗	-	✗	✗	✗
Ours	**1.4**	1.6	1.8	**1.0**	**1.0**	**1.0**
Method	**T2I**	**T2A**	**Multi-view img**	**I2V**	**T2M**	**T2V**
Show-o [48]	2.8	✗	✗	✗	✗	✗
SEED-X [11]	2.0	✗	✗	✗	✗	✗
LWM [23]	4.2	✗	✗	✗	✗	✗
U-IO 2 [26]	4.5	2.0	-	-	✗	✗
Ours	**1.5**	**1.0**	**1.0**	**1.0**	**1.0**	**1.0**

Metric ❶ For execution evaluation, we first evaluated the single-run ***pass rate (Pass@1)*** of compilation and execution, following Xue *et al.* [49]. ❷ For outcome quality and instruction-following, we conducted a systematic user study

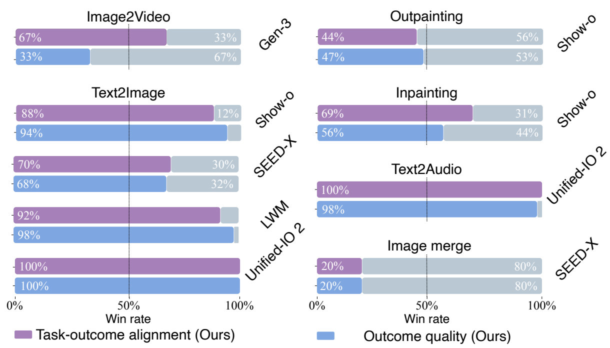

Figure 6. **Comparison of our win rates** with the state-of-the-art unified multimodal models.

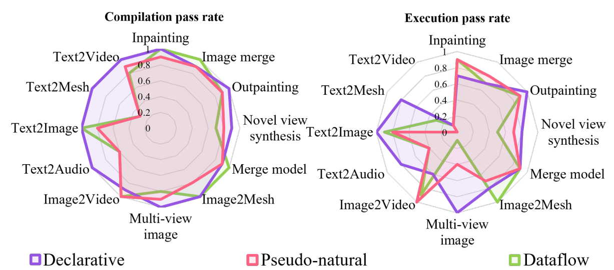

Figure 7. **Comparison of syntax styles.** Metric: Pass@1 (↑). See Appendix for details.

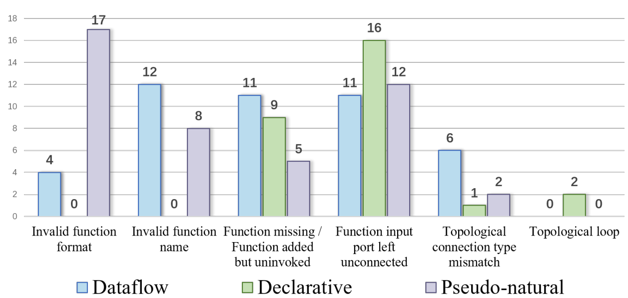

Figure 8. **Comparative error distribution** for dataflow, declarative, and pseudo-natural syntax styles, illustrating six types of errors occur when testing on the 120 generative tasks.

with five annotators who ranked outputs from all frameworks for comparison, the metrics are following:

- ***Text-outcome alignment:*** We measured the degree of correspondence between generated outputs and their intended task specifications. Higher alignment scores indicated closer matches between system outputs and expected results based on input requirements.
- ***Outcome quality:*** We assessed generated outputs based on three criteria: aesthetic appeal, structural coherence, and technical quality. This metric encompassed visual clarity, presentation effectiveness, and adherence to task-specific quality standards.

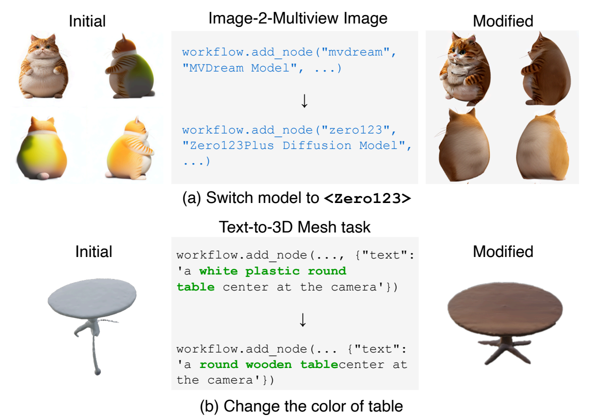

Figure 9. **Symbolic Flow Editing.** We present examples of modifying (a) *functions*, where users can directly change models by editing code to achieve desired effects, and (b) *parameters*, such as adjusting textual prompts (treated as a type of parameter) to alter the color of 3D assets.

- ***Average rank:*** We computed this metric by first ranking each model's performance on text-outcome alignment and outcome quality for individual samples, then calculating the mean rank across all tasks.
- ***Win rate:*** A "win" is recorded when our method ranks higher than a competitor for a given sample. The win rate represented the percentage of successful comparisons, serving as a measure of relative performance advantage.

Table 3. **Agentic design [49] *vs.* symbolic inference (Ours).** We calculate the average pass rate (Pass@1, ↑) on compilation and execution. Results are averaged across 120 generative tasks.

Method	Compilation	Execution
GenAgent [49]	0.84	0.63
Ours	**0.97**	**0.77**

Baselines ❶ **Agentic framework:** We selected GenAgent [49] as our primary baseline method. To ensure fairness, we augmented GenAgent [49] with key functions and parameters as additional input, and increased the maximum refinement iterations to 3. ❷ **Unified multimodal models:** We also compared against the state-of-the-art unified multimodal approaches. In the Text to Image and Inpaint tasks, the Show-o model [48] has a guidance scale of 1.75 and 16 time steps. For Outpaint tasks, we set both left and right expansion degrees to 1. The SEED-x model [11] was configured with a maximum output token count of 1024 and a maximum of 3 history rounds. We enabled three specific options: forced image generation, forced bounding boxes, and forced image optimization. ❸ **Commercial generative model:** The Gen-3 video generation model [37] was configured with 720p resolution (1280 × 768 aspect ratio), using random seed and a video length of 5 seconds.

Implementation details Following Gupta *et al.* [12], we implemented in-context learning to prompt the LM with syntax and logical guidance. Specifically, we performed Retrieval-Augmented Generation (RAG) based on the task description, retrieving three most relevant programs as references. We curated a reference program database containing 16 distinct programs, ensuring no overlap with the target evaluation tasks. All experiments were conducted on a single L4 GPU (24GB), with 1TB external storage, running on a Debian 11 server. ComfyUI served as the back-end for code execution. We used GPT-4o (`gpt-4o-2024-08-06`) as the inference engine and `text-embedding-3-large` as the embedding model.

5.2. Main results

Comparative performance in user study Our symbolic model consistently outperforms state-of-the-art unified models in both text-outcome alignment and result quality across multiple generative tasks. In the user study involving five experienced participants, our model was evaluated against Show-o [48], SEED-X [11], LVM [23], Unified-IO [26], and the commercial Gen-3 [37]. As illustrated in Figure 6, our approach achieved a 94% win rate against Show-o [48] and 98% against LVM [23] in Text to Image tasks. Notably, in Image2Video generation, our model surpassed the commercial Gen-3 with a 67% win rate in text-outcome alignment. Additionally, for Text to Audio, our model attained a 100% win rate in alignment and 98% in quality against Unified-IO [26], underscoring its superior performance across diverse applications. See Appendix for the visualization results.

Is complex agentic design necessary? As shown in Table 3, simpler, symbolic approaches can achieve higher success rates for straightforward tasks without the complexities and costs associated with agentic designs. Unlike GenAgent [49], which employs multi-step planning and actions that can amplify errors and increase computational costs, our symbolic method maintains simplicity and clarity. This reduction in complexity leads to higher success rates in simple tasks by minimizing error propagation and lowering execution costs. However, for more intricate workflows, integrating symbolic representations with agentic strategies may offer enhanced flexibility and performance, suggesting a potential hybrid approach for future research. See Appendix for details.

Representation: neural or symbolic? Our symbolic model outperforms neural models in task generality and output quality without additional training. Table 2 highlights that our symbolic approach successfully handles all 120 generative tasks, including complex categories such as 3D and video generation. In contrast, neural models are limited by their reliance on extensive training data, restricting their ability to manage diverse and complex tasks. Specifically, our model achieves superior average ranks in most 2D tasks like Inpaint, Text to Audio, and Text to Image generation, demonstrating its enhanced adaptability and performance over unified neural frameworks.

Explicit symbolic flow editing Our symbolic representation enables precise and effective control over distinct stages of generative tasks, thus paving the way for the realization of more complex tasks. Figure 9 illustrates examples of modifying ***function*** (model) and ***parameter*** (textual prompt), respectively. By applying explicit program modifications, control over the image generation process is given. See Appendix for more examples.

Error analysis: What constitutes an *LM-friendly* syntax style? A balance between human readability and format correctness is essential for enhancing language model performance, with structural rigidity impacting topological clarity. Upon analysis of the reasoning processes of the 120 test tasks in Figure 8, we identified two main takeaways.

- ❶ **Human readability *vs.* format correctness:** Higher readability in language design correlates with increased format errors. Pseudo-natural language formats exhibited 17 instances of invalid code formats, compared to 4 in dataflow and none in declarative styles. This indicates that while readability facilitates human understanding, it can hinder precise format adherence by language models.
- ❷ **Structural rigidity *vs.* topological clarity:** Structurally rigid and highly modular languages, such as our declarative syntax, tend to introduce topological gaps and connection errors, with 9 instances of missing or uninvoked functions and 16 unlinked input ports. This suggests that increased structural complexity can challenge language models in maintaining clear and accurate dependencies between functions and ports.

6. Conclusion

We have proposed a symbolic generative task description language, combined with an inference engine, providing a novel and efficient way to represent and execute multimodal tasks without the need for task-specific training. By leveraging a pre-trained large language model to infer symbolic task descriptions, our approach has successfully synthesized diverse multimodal tasks, demonstrating its flexibility and potential to unify different generative AI capabilities. Our experiments on 120 tasks have shown that our framework has achieved performance comparable to unified multimodal models, highlighting its expandability and cost-effectiveness.

References

[1] Hassan Akbari, Liangzhe Yuan, Rui Qian, Wei-Hong Chuang, Shih-Fu Chang, Yin Cui, and Boqing Gong. Vatt: Transformers for multimodal self-supervised learning from raw video, audio and text. *Advances in Neural Information Processing Systems*, 34:24206–24221, 2021. 3

[2] Jacob Andreas, Marcus Rohrbach, Trevor Darrell, and Dan Klein. Neural module networks, 2017. 3

[3] Anthropic. The claude 3 model family: Opus, sonnet, haiku, 2024. Corpus ID: 268232499. 3

[4] James Betker, Gabriel Goh, Li Jing, † TimBrooks, Jianfeng Wang, Linjie Li, † LongOuyang, † JuntangZhuang, † JoyceLee, † YufeiGuo, † WesamManassra, † PrafullaDhariwal, † CaseyChu, † YunxinJiao, and Aditya Ramesh. Improving image generation with better captions. 3

[5] Zalán Borsos, Raphaël Marinier, Damien Vincent, Eugene Kharitonov, Olivier Pietquin, Matt Sharifi, Dominik Roblek, Olivier Teboul, David Grangier, Marco Tagliasacchi, and Neil Zeghidour. Audiolm: a language modeling approach to audio generation, 2023. 3

[6] Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. Language models are few-shot learners. *NeurIPS*, 2020. 3

[7] Aakanksha Chowdhery, Sharan Narang, Jacob Devlin, Maarten Bosma, Gaurav Mishra, Adam Roberts, Paul Barham, Hyung Won Chung, Charles Sutton, Sebastian Gehrmann, et al. Palm: Scaling language modeling with pathways. 2023.

[8] DeepSeek-AI, Xiao Bi, Deli Chen, Guanting Chen, Shanhuang Chen, Damai Dai, Chengqi Deng, Honghui Ding, Kai Dong, Qiushi Du, Zhe Fu, Huazuo Gao, Kaige Gao, Wenjun Gao, Ruiqi Ge, Kang Guan, Daya Guo, Jianzhong Guo, Guangbo Hao, Zhewen Hao, Ying He, Wenjie Hu, Panpan Huang, Erhang Li, Guowei Li, Jiashi Li, Yao Li, Y. K. Li, Wenfeng Liang, Fangyun Lin, A. X. Liu, Bo Liu, Wen Liu, Xiaodong Liu, Xin Liu, Yiyuan Liu, Haoyu Lu, Shanghao Lu, Fuli Luo, Shirong Ma, Xiaotao Nie, Tian Pei, Yishi Piao, Junjie Qiu, Hui Qu, Tongzheng Ren, Zehui Ren, Chong Ruan, Zhangli Sha, Zhihong Shao, Junxiao Song, Xuecheng Su, Jingxiang Sun, Yaofeng Sun, Minghui Tang, Bingxuan Wang, Peiyi Wang, Shiyu Wang, Yaohui Wang, Yongji Wang, Tong Wu, Y. Wu, Xin Xie, Zhenda Xie, Ziwei Xie, Yiliang Xiong, Hanwei Xu, R. X. Xu, Yanhong Xu, Dejian Yang, Yuxiang You, Shuiping Yu, Xingkai Yu, B. Zhang, Haowei Zhang, Lecong Zhang, Liyue Zhang, Mingchuan Zhang, Minghua Zhang, Wentao Zhang, Yichao Zhang, Chenggang Zhao, Yao Zhao, Shangyan Zhou, Shunfeng Zhou, Qihao Zhu, and Yuheng Zou. Deepseek llm: Scaling open-source language models with longtermism, 2024. 3

[9] Michael D Ernst. Natural language is a programming language: Applying natural language processing to software development. In *2nd Summit on Advances in Programming Languages (SNAPL 2017)*. Schloss-Dagstuhl-Leibniz Zentrum für Informatik, 2017. 4

[10] Han Fang, Pengfei Xiong, Luhui Xu, and Yu Chen. Clip2video: Mastering video-text retrieval via image clip. *arXiv preprint arXiv:2106.11097*, 2021. 3

[11] Yuying Ge, Sijie Zhao, Jinguo Zhu, Yixiao Ge, Kun Yi, Lin Song, Chen Li, Xiaohan Ding, and Ying Shan. Seed-x: Multimodal models with unified multi-granularity comprehension and generation. *arXiv preprint arXiv:2404.14396*, 2024. 6, 7, 8

[12] Tanmay Gupta and Aniruddha Kembhavi. Visual programming: Compositional visual reasoning without training. In *CVPR*, pages 14953–14962, 2023. 2, 3, 8

[13] Shibo Hao, Tianyang Liu, Zhen Wang, and Zhiting Hu. Toolkengpt: Augmenting frozen language models with massive tools via tool embeddings, 2024. 2

[14] Wenyi Hong, Ming Ding, Wendi Zheng, Xinghan Liu, and Jie Tang. Cogvideo: Large-scale pretraining for text-to-video generation via transformers, 2022. 3

[15] Ronghang Hu, Jacob Andreas, Marcus Rohrbach, Trevor Darrell, and Kate Saenko. Learning to reason: End-to-end module networks for visual question answering. In *Proceedings of the IEEE international conference on computer vision*, pages 804–813, 2017. 3

[16] Albert Q. Jiang, Alexandre Sablayrolles, Antoine Roux, Arthur Mensch, Blanche Savary, Chris Bamford, Devendra Singh Chaplot, Diego de las Casas, Emma Bou Hanna, Florian Bressand, Gianna Lengyel, Guillaume Bour, Guillaume Lample, Lélio Renard Lavaud, Lucile Saulnier, Marie-Anne Lachaux, Pierre Stock, Sandeep Subramanian, Sophia Yang, Szymon Antoniak, Teven Le Scao, Théophile Gervet, Thibaut Lavril, Thomas Wang, Timothée Lacroix, and William El Sayed. Mixtral of experts, 2024. 3

[17] Justin Johnson, Bharath Hariharan, Laurens Van Der Maaten, Judy Hoffman, Li Fei-Fei, C Lawrence Zitnick, and Ross Girshick. Inferring and executing programs for visual reasoning. In *Proceedings of the IEEE international conference on computer vision*, pages 2989–2998, 2017. 3

[18] Felix Kreuk, Gabriel Synnaeve, Adam Polyak, Uriel Singer, Alexandre Défossez, Jade Copet, Devi Parikh, Yaniv Taigman, and Yossi Adi. Audiogen: Textually guided audio generation, 2023. 3

[19] Junnan Li, Ramprasaath R Selvaraju, Akhilesh Gotmare, Shafiq Joty, Caiming Xiong, and Steven Chu Hong Hoi. Align before fuse: Vision and language representation learning with momentum distillation. *Advances in neural information processing systems*, 34:9694–9705, 2021.

[20] Junnan Li, Dongxu Li, Caiming Xiong, and Steven Hoi. Blip: Bootstrapping language-image pre-training for unified vision-language understanding and generation. In *International Conference on Machine Learning*, pages 12888–12900. PMLR, 2022. 3

[21] Raymond Li, Loubna Ben Allal, Yangtian Zi, Niklas Muennighoff, Denis Kocetkov, Chenghao Mou, Marc Marone, Christopher Akiki, Jia Li, Jenny Chim, Qian Liu, Evgenii Zheltonozhskii, Terry Yue Zhuo, Thomas Wang, Olivier Dehaene, Mishig Davaadorj, Joel Lamy-Poirier, João Monteiro, Oleh Shliazhko, Nicolas Gontier, Nicholas Meade, Armel Zebaze, Ming-Ho Yee, Logesh Kumar Umapathi, Jian Zhu, Benjamin Lipkin, Muhtasham Oblokulov, Zhiruo Wang,

Rudra Murthy, Jason Stillerman, Siva Sankalp Patel, Dmitry Abulkhanov, Marco Zocca, Manan Dey, Zhihan Zhang, Nour Fahmy, Urvashi Bhattacharyya, Wenhao Yu, Swayam Singh, Sasha Luccioni, Paulo Villegas, Maxim Kunakov, Fedor Zhdanov, Manuel Romero, Tony Lee, Nadav Timor, Jennifer Ding, Claire Schlesinger, Hailey Schoelkopf, Jan Ebert, Tri Dao, Mayank Mishra, Alex Gu, Jennifer Robinson, Carolyn Jane Anderson, Brendan Dolan-Gavitt, Danish Contractor, Siva Reddy, Daniel Fried, Dzmitry Bahdanau, Yacine Jernite, Carlos Muñoz Ferrandis, Sean Hughes, Thomas Wolf, Arjun Guha, Leandro von Werra, and Harm de Vries. Starcoder: may the source be with you!, 2023. 3

[22] Xiujun Li, Xi Yin, Chunyuan Li, Pengchuan Zhang, Xiaowei Hu, Lei Zhang, Lijuan Wang, Houdong Hu, Li Dong, Furu Wei, et al. Oscar: Object-semantics aligned pre-training for vision-language tasks. In *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part XXX 16*, pages 121–137. Springer, 2020. 3

[23] Hao Liu, Wilson Yan, Matei Zaharia, and Pieter Abbeel. World model on million-length video and language with ringattention. *arXiv preprint*, 2024. 6, 8

[24] Anton Lozhkov, Raymond Li, Loubna Ben Allal, Federico Cassano, Joel Lamy-Poirier, Nouamane Tazi, Ao Tang, Dmytro Pykhtar, Jiawei Liu, Yuxiang Wei, Tianyang Liu, Max Tian, Denis Kocetkov, Arthur Zucker, Younes Belkada, Zijian Wang, Qian Liu, Dmitry Abulkhanov, Indraneil Paul, Zhuang Li, Wen-Ding Li, Megan Risdal, Jia Li, Jian Zhu, Terry Yue Zhuo, Evgenii Zheltonozhskii, Nii Osae Osae Dade, Wenhao Yu, Lucas Krauß, Naman Jain, Yixuan Su, Xuanli He, Manan Dey, Edoardo Abati, Yekun Chai, Niklas Muennighoff, Xiangru Tang, Muhtasham Oblokulov, Christopher Akiki, Marc Marone, Chenghao Mou, Mayank Mishra, Alex Gu, Binyuan Hui, Tri Dao, Armel Zebaze, Olivier Dehaene, Nicolas Patry, Canwen Xu, Julian McAuley, Han Hu, Torsten Scholak, Sebastien Paquet, Jennifer Robinson, Carolyn Jane Anderson, Nicolas Chapados, Mostofa Patwary, Nima Tajbakhsh, Yacine Jernite, Carlos Muñoz Ferrandis, Lingming Zhang, Sean Hughes, Thomas Wolf, Arjun Guha, Leandro von Werra, and Harm de Vries. Starcoder 2 and the stack v2: The next generation, 2024. 3

[25] Jiasen Lu, Christopher Clark, Rowan Zellers, Roozbeh Mottaghi, and Aniruddha Kembhavi. Unified-io: A unified model for vision, language, and multi-modal tasks, 2022. 2, 3

[26] Jiasen Lu, Christopher Clark, Sangho Lee, Zichen Zhang, Savya Khosla, Ryan Marten, Derek Hoiem, and Aniruddha Kembhavi. Unified-io 2: Scaling autoregressive multimodal models with vision, language, audio, and action, 2023. 2, 3, 6, 8

[27] Pan Lu, Baolin Peng, Hao Cheng, Michel Galley, Kai-Wei Chang, Ying Nian Wu, Song-Chun Zhu, and Jianfeng Gao. Chameleon: Plug-and-play compositional reasoning with large language models. *Advances in Neural Information Processing Systems*, 36, 2024. 2

[28] David Mizrahi, Roman Bachmann, Oğuzhan Fatih Kar, Teresa Yeo, Mingfei Gao, Afshin Dehghan, and Amir Zamir. 4m: Massively multimodal masked modeling, 2023. 3

[29] OpenAI. Chatgpt: Optimizing language models for dialogue. http://web.archive.org/web/20230109000707/https://openai.com/blog/chatgpt/, 2022. 3

[30] OpenAI et al. Gpt-4 technical report, 2024.

[31] OpenAI et al. Gpt-4o system card, 2024. 2, 3

[32] Ben Poole, Ajay Jain, Jonathan T. Barron, and Ben Mildenhall. Dreamfusion: Text-to-3d using 2d diffusion, 2022. 3

[33] Yujia Qin, Shihao Liang, Yining Ye, Kunlun Zhu, Lan Yan, Yaxi Lu, Yankai Lin, Xin Cong, Xiangru Tang, Bill Qian, et al. Toolllm: Facilitating large language models to master 16000+ real-world apis. *arXiv preprint arXiv:2307.16789*, 2023. 2

[34] Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, et al. Learning transferable visual models from natural language supervision. In *International conference on machine learning*, pages 8748–8763. PMLR, 2021. 3

[35] Alec Radford, Jong Wook Kim, Tao Xu, Greg Brockman, Christine McLeavey, and Ilya Sutskever. Robust speech recognition via large-scale weak supervision, 2022.

[36] Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. High-resolution image synthesis with latent diffusion models, 2022.

[37] Runway. Gen-3. https://runwayml.com/blog/introducing-gen-3-alpha/, 2024. 3, 7, 8

[38] Timo Schick, Jane Dwivedi-Yu, Roberto Dessì, Roberta Raileanu, Maria Lomeli, Luke Zettlemoyer, Nicola Cancedda, and Thomas Scialom. Toolformer: Language models can teach themselves to use tools, 2023. 2

[39] Yongliang Shen, Kaitao Song, Xu Tan, Dongsheng Li, Weiming Lu, and Yueting Zhuang. Hugginggpt: Solving ai tasks with chatgpt and its friends in hugging face, 2023. 2, 3

[40] Zineng Tang, Ziyi Yang, Mahmoud Khademi, Yang Liu, Chenguang Zhu, and Mohit Bansal. Codi-2: In-context, interleaved, and interactive any-to-any generation, 2023. 2, 3

[41] Zineng Tang, Ziyi Yang, Chenguang Zhu, Michael Zeng, and Mohit Bansal. Any-to-any generation via composable diffusion, 2023. 2, 3

[42] Zineng Tang, Ziyi Yang, Chenguang Zhu, Michael Zeng, and Mohit Bansal. Any-to-any generation via composable diffusion. *Advances in Neural Information Processing Systems*, 36, 2024. 2

[43] Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, Aurelien Rodriguez, Armand Joulin, Edouard Grave, and Guillaume Lample. Llama: Open and efficient foundation language models, 2023. 3

[44] Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, Dan Bikel, Lukas Blecher, Cristian Canton Ferrer,

Moya Chen, Guillem Cucurull, David Esiobu, Jude Fernandes, Jeremy Fu, Wenyin Fu, Brian Fuller, Cynthia Gao, Vedanuj Goswami, Naman Goyal, Anthony Hartshorn, Saghar Hosseini, Rui Hou, Hakan Inan, Marcin Kardas, Viktor Kerkez, Madian Khabsa, Isabel Kloumann, Artem Korenev, Punit Singh Koura, Marie-Anne Lachaux, Thibaut Lavril, Jenya Lee, Diana Liskovich, Yinghai Lu, Yuning Mao, Xavier Martinet, Todor Mihaylov, Pushkar Mishra, Igor Molybog, Yixin Nie, Andrew Poulton, Jeremy Reizenstein, Rashi Rungta, Kalyan Saladi, Alan Schelten, Ruan Silva, Eric Michael Smith, Ranjan Subramanian, Xiaoqing Ellen Tan, Binh Tang, Ross Taylor, Adina Williams, Jian Xiang Kuan, Puxin Xu, Zheng Yan, Iliyan Zarov, Yuchen Zhang, Angela Fan, Melanie Kambadur, Sharan Narang, Aurelien Rodriguez, Robert Stojnic, Sergey Edunov, and Thomas Scialom. Llama 2: Open foundation and fine-tuned chat models, 2023. 3

[45] Michael T Ullman. A neurocognitive perspective on language: The declarative/procedural model. *Nature reviews neuroscience*, 2(10):717–726, 2001. 4

[46] Peng Wang, An Yang, Rui Men, Junyang Lin, Shuai Bai, Zhikang Li, Jianxin Ma, Chang Zhou, Jingren Zhou, and Hongxia Yang. Ofa: Unifying architectures, tasks, and modalities through a simple sequence-to-sequence learning framework. In *International Conference on Machine Learning*, pages 23318–23340. PMLR, 2022. 3

[47] Wenhui Wang, Hangbo Bao, Li Dong, Johan Bjorck, Zhiliang Peng, Qiang Liu, Kriti Aggarwal, Owais Khan Mohammed, Saksham Singhal, Subhojit Som, et al. Image as a foreign language: Beit pretraining for all vision and vision-language tasks. *arXiv preprint arXiv:2208.10442*, 2022. 3

[48] 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. 6, 7, 8

[49] Xiangyuan Xue, Zeyu Lu, Di Huang, Wanli Ouyang, and Lei Bai. Genagent: Build collaborative ai systems with automated workflow generation–case studies on comfyui. *arXiv preprint arXiv:2409.01392*, 2024. 2, 3, 4, 6, 7, 8

[50] Rui Yan, Mike Zheng Shou, Yixiao Ge, Alex Jinpeng Wang, Xudong Lin, Guanyu Cai, and Jinhui Tang. Video-text pre-training with learned regions. *arXiv preprint arXiv:2112.01194*, 2021. 3

[51] Jinyu Yang, Jiali Duan, Son Tran, Yi Xu, Sampath Chanda, Liqun Chen, Belinda Zeng, Trishul Chilimbi, and Junzhou Huang. Vision-language pre-training with triple contrastive learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 15671–15680, 2022.

[52] Rowan Zellers, Jiasen Lu, Ximing Lu, Youngjae Yu, Yanpeng Zhao, Mohammadreza Salehi, Aditya Kusupati, Jack Hessel, Ali Farhadi, and Yejin Choi. Merlot reserve: Neural script knowledge through vision and language and sound. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 16375–16387, 2022.

[53] Yan Zeng, Xinsong Zhang, and Hang Li. Multi-Grained Vision Language Pre-Training: Aligning Texts with Visual Concepts. In *International Conference on Machine Learning*, pages 25994–26009. PMLR, 2022. 3

[54] Jun Zhan, Junqi Dai, Jiasheng Ye, Yunhua Zhou, Dong Zhang, Zhigeng Liu, Xin Zhang, Ruibin Yuan, Ge Zhang, Linyang Li, Hang Yan, Jie Fu, Tao Gui, Tianxiang Sun, Yugang Jiang, and Xipeng Qiu. Anygpt: Unified multimodal llm with discrete sequence modeling, 2024. 2, 3

[55] Dong Zhang, Shimin Li, Xin Zhang, Jun Zhan, Pengyu Wang, Yaqian Zhou, and Xipeng Qiu. Speechgpt: Empowering large language models with intrinsic cross-modal conversational abilities, 2023. 3