Interactive Visual Assessment for Text-to-Image Generation Models

Xiaoyue Mi, Fan Tang, Juan Cao, Qiang Sheng, Ziyao Huang, Peng Li, Yang Liu, *Senior Member, IEEE*, and Tong-Yee Lee, *Senior Member, IEEE*

***Abstract*—Visual generation models have achieved remarkable progress in computer graphics applications but still face significant challenges in real-world deployment. Current assessment approaches for visual generation tasks typically follow an isolated three-phase framework: test input collection, model output generation, and user assessment. These fashions suffer from fixed coverage, evolving difficulty, and data leakage risks, limiting their effectiveness in comprehensively evaluating increasingly complex generation models. To address these limitations, we propose DyEval, an LLM-powered dynamic interactive visual assessment framework that facilitates collaborative evaluation between humans and generative models for text-to-image systems. DyEval features an intuitive visual interface that enables users to interactively explore and analyze model behaviors, while adaptively generating hierarchical, fine-grained, and diverse textual inputs to continuously probe the capability boundaries of the models based on their feedback. Additionally, to provide interpretable analysis for users to further improve tested models, we develop a contextual reflection module that mines failure triggers of test inputs and reflects model potential failure patterns, supporting in-depth analysis using the logical reasoning ability of LLM. Qualitative and quantitative experiments demonstrate that DyEval can effectively help users identify max up to 2.56 timesmore generation failures than conventional methods, and uncover complex and rare failure patterns, such as issues with pronoun generation and specific cultural context generation. Our framework provides valuable insights for improving generative models and has broad implications for advancing the reliability and capabilities of visual generation systems across various domains.**

Received 21 November 2024; revised 16 January 2026; accepted 25 January 2026. Date of publication 28 January 2026; date of current version 2 March 2026. This work was supported in part by the National Natural Science Foundation of China under Grant 62572458 and Grant 62406310, in part by the Strategic Priority Research Program of the Chinese Academy of Sciences under Grant XDB0680202, in part by Beijing Science and Technology Plan Project under Grant Z251100008125009, and in part by the National Science and Technology Council, Taiwan under Grant 113-2221-E-006-161-MY3. Recommended for acceptance by J. Liao. *(Corresponding author: Fan Tang.)*

Xiaoyue Mi, Fan Tang, Juan Cao, Qiang Sheng, and Ziyao Huang are with the Institute of Computing Technology, Chinese Academy of Sciences, Beijing 100190, China, and also with the University of Chinese Academy of Sciences, Beijing 100049, China (e-mail: mixiaoyue19s@ict.ac.cn; tangfan@ict.ac.cn; caojuan@ict.ac.cn; shengqiang18z@ict.ac.cn; huangziyao19f@ict.ac.cn).

Peng Li is with the Institute for AI Industry Research (AIR), Tsinghua University, Beijing 100084, China (e-mail: pengli09@gmail.com).

Yang Liu is with the Institute for AI Industry Research (AIR), Tsinghua University, Beijing 100084, China, and also with the Department of Computer Science and Technology, Tsinghua University, Beijing 100084, China (e-mail: liuyang2011@tsinghua.edu.cn).

Tong-Yee Lee is with National Cheng Kung University, Tainan 70101, Taiwan (e-mail: tonylee@mail.ncku.edu.tw).

Code is available at https://github.com/KululuMi/DyEval.

This article has supplementary downloadable material available at https://doi.org/10.1109/TVCG.2026.3658714, provided by the authors.

Digital Object Identifier 10.1109/TVCG.2026.3658714

***Index Terms*—Visual assessment, dynamic testing, large language model, in-context learning, human-computer interaction.**

I. Introduction

RECENT years have witnessed remarkable advances in visual generation models [2], [30], [33]. These models show impressive capabilities in generating vivid images and have found applications across diverse computer graphics domains [9], [12], [41], [45]. Despite their powerful capabilities, they still exhibit various generalization failures when deployed in real-world computer graphics scenarios. Given the vast range of images these models can generate, the potential generalization failures are highly varied and unpredictable [6]. Thoroughly testing these models to find significant failures reliably is crucial for enhancing their performance, yet this remains a major challenge.

Current assessment approaches primarily rely on datasets and require significant manual effort, including sequential isolated three phases: the collection of test inputs, model generation, and human assessment of model outputs [1], [18], [21], [22], [23], [34], [37], [40], [42]. These fashions are widely applied but difficult to evolve. As visual generation models advance and application scenarios expand, static datasets may become outdated, necessitating substantial resources to update and maintain their relevance. This limited scalability also manifests in the fixed coverage of test inputs, which restricts their ability to detect model deficiencies beyond predefined scenarios.

Interactive testing methods have emerged as a promising alternative to address above limitations. Du et al. [6] propose an adversarial attack method for text-to-image models, which is adaptive based on tested model feedback but narrowly focuses on evaluating the robustness of noun changes in white-box models. Meanwhile, open-ended human-in-the-loop testing [10], [11], [31], [32] leverages Large Language Models (LLMs) and interactions with evaluators to generate challenging data for testing models on coherent, manually specified topics. Although effective in text-input text-output tasks or image understanding tasks, these methods cannot be directly applied to the visual generation domain due to the complexity of visual generation failures [3]. Evaluators will have a huge cost to mark failures of generated images in detail.

In this paper, we introduce DyEval, a novel interactive visual assessment framework that leverages an LLM through in-context learning to adaptively generate test inputs based on model feedback, effectively identifying failures in open domains (Fig. 1). DyEval features an intuitive visual interface that enables users to interactively explore and analyze model behaviors while maintaining a comprehensive overview of the testing process. DyEval is built upon a tree-based structure systematically recording the testing process. Each tree node contains

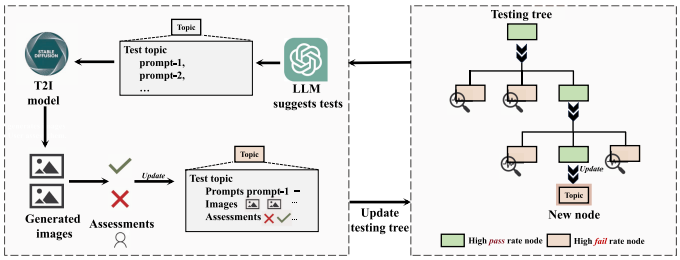


Fig. 1. Diagram of the DyEval, a dynamic interactive testing framework for Text-to-Image (T2I) models. DyEval leverages a Large Language Model (LLM) to generate test prompts based on model feedback dynamically. The T2I model outputs are assessed, and results are used to update a test tree. Nodes with high pass rates (green) continue to explore deeper layers, while nodes with high fail rates (brown) are analyzed by the LLM to analyze failure reasons.

test topics, specific prompts as test inputs, and corresponding evaluation results.

DyEval operates through two main components: First, in the test node construction phase, the LLM generates test inputs based on the current test topic and existing contexts. Evaluators then assess the generated images through the interactive interface, determining whether they pass or fail the specified criteria. By requiring only binary pass/fail judgments to guide the exploration process, DyEval is designed to be evaluation-metric agnostic and highly extensible to different capability dimensions. Second, DyEval performs adaptive exploration based on the evaluation results. For nodes with high pass rates, the LLM proposes new test topics for deeper tree exploration. For nodes with lots of failures, DyEval employs contextual reflection, combining dynamic failure location to identify failure-inducing parts with a self-reflection module analyzing potential failure types and reasons based on existing test contexts. Benefiting from the dynamic interactivity of DyEval, we can also avoid the risk of data leakage [28], [29], which typically arises from public static test sets leading to targeted model optimizations and, consequently, skewed evaluation results.

Experimentally, we showcase the efficacy of DyEval by evaluating various state-of-the-art text-to-image models from four perspectives: Object, Relation, Attribute, and Context (global attributes in the image, such as style). Quantitative experiments demonstrate that DyEval significantly outperforms traditional static evaluation methods, identifying max up to 2.56 times more failure cases when evaluating the same number of text-image pairs. Our experiments also reveal consistent patterns across different generation models. All tested models perform better with material objects over abstract ones, static over dynamic attributes, and explicit over implicit relations. While excelling at style-related tasks, they struggle with culture and knowledge-based generations, particularly with cultural nuances and implicit relations. In Parts-of-Speech (PoS) analysis of text inputs, we find SDXL and SD3 show improvements over SD1-5 and SD2-1, but they continue to struggle with specific linguistic elements such as quantifiers and pronouns. Through case studies, DyEval uncovers unexpected testing perspectives and intricate failure patterns. The contextual reflection module identifies specific triggers for model failures, such as culturally specific words (e.g., "kimono") and certain text combinations. These findings provide valuable insights into the current limitations and potential areas for improvement in text-to-image generation models. In summary, DyEval offers a novel perspective in text-to-image model testing, significantly aiding evaluators in comprehensively understanding the boundaries of model capabilities and inspiring future enhancements. In this paper, our main contributions are listed below:

- We propose a novel adaptive testing method, DyEval, a dynamic and open-domain evaluation method for text-to-image models. DyEval can adapt to user-defined criterias like alignment, bias, and more, requiring only access to the input and output of the model being tested.
- With the LLM-powered iteration system and interactive visual interface, DyEval gradually adapts tests based on user feedback to find a variety of model failures. The metric-agnostic design enables evaluation across different capability dimensions. Compared with static testing methods [18], [23], [34], DyEval allows limitless test scope, granularity expansion, and coevolution with evolving tested models.
- Experiments validate that DyEval effectively finds the complex and uncommon failures of tested models, such as pronoun-induced degradation, which provides valuable and interpretable insights for further improvement.

II. Related Works

A. Assessments of Visual Generation Models

In the field of visual generation, there has been a significant increase in evaluation studies over the past two years, especially in text-to-image generation. Currently, static evaluation [1], [18], [21], [22], [23], [34], [37], [40], [42], [46] are the predominant method for evaluation, which employs a one-time assessment using pre-collected data sets, which can show the generalization ability of the model on specific data. Firstly, evaluators determine testing perspectives such as concept conjunction [26], and spatial relationships [39]. Then, they gather different test prompts according to their testing perspectives. There are four main ways that models get their prompts: existing text-image datasets [24], human usage records from online text-to-image tools like Midjourney [21], [34], [37], semi-automated generation using LLMs with predefined templates [1], [18], and combinations of them [22], [23], [40], [42]. Finally, evaluators assess the model outputs using established metrics and manual evaluation. Due to the high challenge of image generation tasks, manual assessment is indispensable and remains the most reliable [19].

These works only provide static evaluations, they are unable to adapt to evolving models and human needs. Du et al. [6] propose a gradient-based adversarial attack method for text-to-image models, dynamically investigating the adversarial robustness of model inputs by adding or changing nouns. However, their approach is hard to generalize in the open domain.

In this paper, we offer a novel adaptive visual assessment method that enables users to specify test themes (e.g. culture, knowledge, count, and more), test aspects (e.g. alignment, bias, and more), and test metrics (e.g. Clip score, human assessment, and more) in the open domain and adjust the testing process interactively based on the model assessment feedback. This evaluation approach is useful in accommodating the open domain and evolving capabilities of the text-to-image models and can be seen as complementary aspects of current evaluations in text-to-image models, jointly enhancing the reliability and practical utility of the models.

Several studies [4], [5], [16], [27] have utilized large language models (LLMs) as metrics to evaluate text-image alignment in text-to-image tasks. For instance, LLMScore [27] converts images into both image-level and object-level visual descriptions, subsequently employing LLMs to assess the alignment between the generated images and their corresponding texts through evaluation instructions. X-IQE [4] uses large visual language models to generate textual explanations of generated image and text alignment. Several studies [5], [16] employ Visual Question Answering (VQA) to assess the alignment between text and images. In these approaches, a language model is utilized to automatically generate multiple question-answer pairs based on a given text input. The faithfulness of the image is then evaluated by determining whether existing VQA models can accurately respond to these questions using the corresponding image.

Different from focusing on text-image alignment metrics, DyEval is an interactive assessment protocol, and decoupled from specific aspects and assessment metrics.

B. Human-LLM Collaborative Model Testing

Leveraging LLMs for human-aided evaluation, i.e. open-ended human-in-the-loop testing, has been well-recognized in text-input text-output tasks [10], [31], [32] and image-input understanding tasks [11]. They adopt strategies from software engineering, engaging individuals to generate test scenarios with assistance from LLMs. Utilizing human intervention to explore input scenarios beyond conventional training and validation datasets, this testing methodology has effectively identified consistent failures in state-of-the-art models, even performing unproblematic on static benchmarks.

For text-input text-output tasks, they manually specify testing topics, using the powerful text generation, rich knowledge, and logical reasoning abilities of LLMs to create test inputs. Then, evaluators select and evaluate model errors manually and use this feedback to generate new test inputs. Similarly, text-to-image is also a text-input task, and its evaluation heavily depends on manual assessment, with no similar testing methods yet in the field.

However, these methods cannot be directly applied to the text-to-image domain due to the complexity of image generation failures. Considering the complex dimensions outputs result in many types of failure [3] in image generation tasks, it would be highly costly for humans to detailly mark these failures. In this paper, we explore human-aided evaluation in the text-to-image task and introduce a contextual reflection module to alleviate this problem by automatically identifying minimal failure test inputs and conducting analysis based on LLMs.

III. Design Objectives

Based on our analysis of limitations in current visual assessment fashions, we identify four key design objectives for our visual assessment framework for text-to-image models:

O1 Dynamic coverage: Our visual assessment framework should overcome the fixed coverage limitation of static testing by supporting open-ended exploration of test scenarios. The ideal framework should enable visual exploration of dynamically generated test cases across different aspects. Through human assessment feedback, Our framework can interactively adjust testing granularity based on discovered failure patterns. The visualization framework should support visual navigation between breadth-first exploration for comprehensive coverage and depth-first investigation for specific issues, enabling users to continuously expand the testing boundary without being constrained by predefined scenarios.

O2 Evolutionary adaptability: To address the evolution difficulty in static datasets, the visual assessment framework should facilitate dynamic visualization of testing strategies that adapt based on model feedback and user needs. Through interactive visualizations and visual feedback loops, users can continuously refine test cases without requiring complete dataset reconstruction, ensuring the testing process remains effective as models advance.

O3 Anti-leakage design: To prevent the data leakage issues common in static testing, the visual assessment framework should generate test inputs dynamically rather than fixed test sets. And test input distribution is broad enough to make it difficult to target optimization overfitting even if one knows how the test inputs are generated. Through visual monitoring of customizable testing criteria that evolve with model capabilities, the framework maintains testing effectiveness while preventing models from being specifically optimized for known test cases.

O4 Visual scalability: The framework should facilitate efficient visual analysis of large-scale testing results through interactive visualization of hierarchical test structures at multiple levels of detail. The framework should support visual exploration and pattern discovery of failure triggers through coordinated views, while providing clear LLM-generated insights and failure analysis. The interface should maintain overview and detail-on-demand capabilities while ensuring framework responsiveness and clarity.

IV. Methods

In this section, we present the formulation and overall pipeline in Section IV-A. Subsequently, we provide detailed descriptions of the two primary steps of DyEval: testing node construction (Section IV-B) and deepening node exploration (Section IV-C).

A. Overview

Formulation: A $test$ is defined as a combination of a text input i and a generated output image x from a text-to-image model m. A test (x, i) passes if the generated image x meets evaluator preferences and intentions (e.g., aesthetic appeal, semantic alignment, level of detail, fairness) for the input i; otherwise, it fails [11], [31], [32]. A bug for a given text input i occurs when the expected pass rate of generated images X_{gen} for i falls below an evaluator-expected pass rate ρ. Formally, a test for input i, denoted TestGroup(i), is considered a bug if:

$$\mathbb{E}_{x\sim\mathbf{P}(X_{gen}|i,m)}[\text{test}(x,i)\text{ passes }] < \rho. \tag{1}$$

We define a topic (t_o) as a coherent group of tests whose text prompts are united by an explainable concept (e.g., "dogs") and share similar evaluation expectations [8], [38]. A topic corresponds to a set of text inputs I relevant to t_o. The text-to-image model m generates output images X_{gen} for these inputs. The set of tests within a topic t_o can be described as:

$$\text{TopicTests}(t_o) = \{\text{test}(x,i) \mid i \in I_{t_o}, x \in X_{gen}(i,m)\}. \tag{2}$$

DyEval aims to assist evaluators in revealing test topics and test inputs with high failure rates, thereby accurately finding the capability boundaries of the tested model.

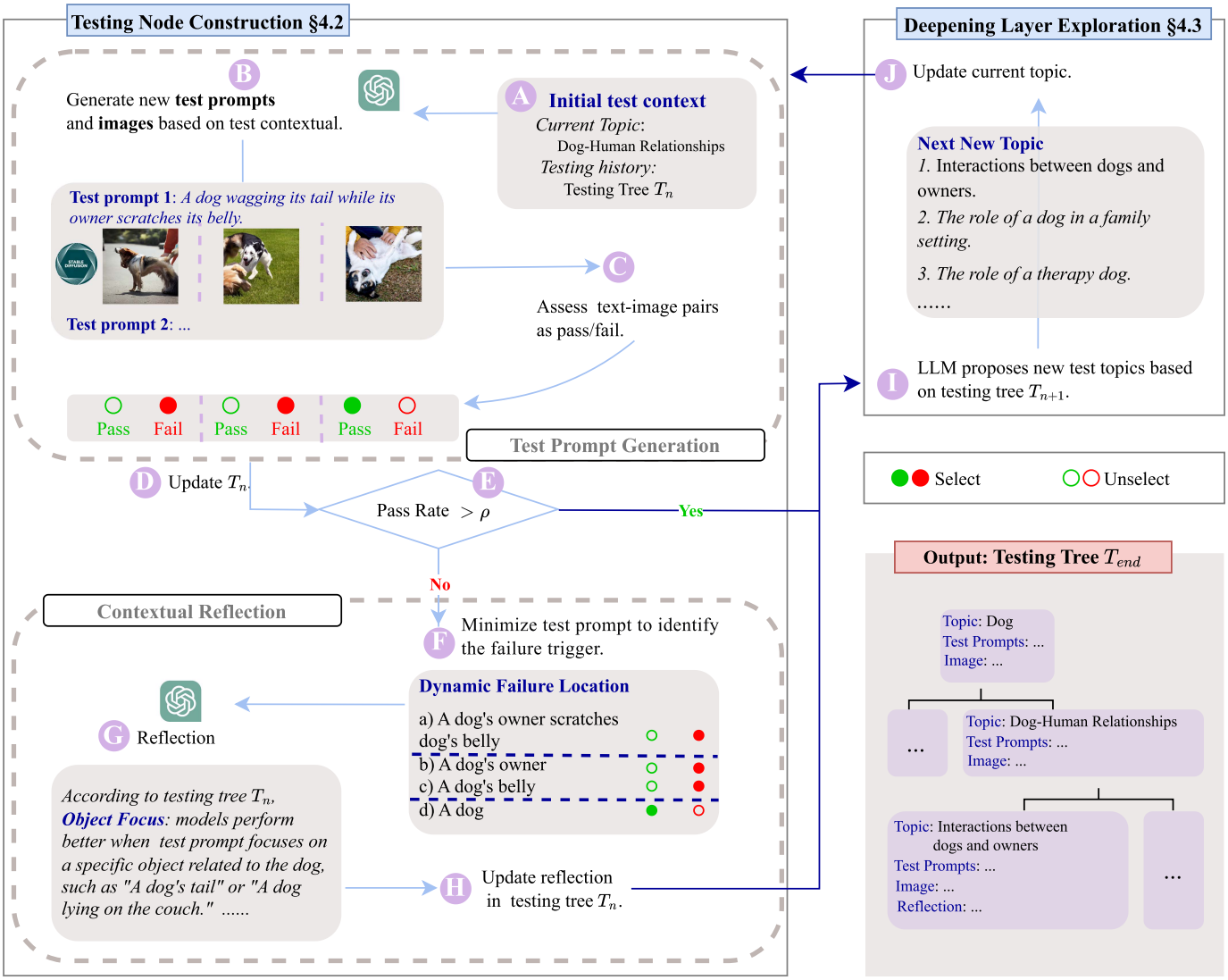


Fig. 2. Overview of DyEval. The DyEval process begins with setting initial topics and model parameters (Step A). In the test input generation phase, the LLM creates initial test inputs for the selected topic (Step B). Evaluators then review and annotate the generated images (Step C) and update the test tree accordingly (Step D). If a test node shows low average pass rates (Step E), the LLM refines these inputs to identify potential failure triggers in the dynamic failure location module (Step F). It then reflects on all available information related to the current topic (Step G) and updates the test tree (Step H). If the pass rates are satisfactory, the LLM proposes new topics based on the test context (history testing records in the testing tree) (Step I). Evaluators can select these new topics (Step J) to generate additional test inputs, continuing the evaluation cycle. This iterative process continues until the maximum exploration depth is reached.

The entire DyEval process is recorded as a **test tree** T, characterized by a maximum depth $d_{\max}$ and a maximum width $w_{\max}$, which also captures test contexts. A test node $t_{d,w} \in T$ (where d is depth and w is width index) is defined by its components:

$$t_{d,w} = (t_{od,w}, \text{TopicTests}(t_o), \text{Result}_{d,w}, \text{Reflection}_{d,w}), \quad (3)$$

where $\text{TopicConcept}_{d,w}$ is the specific topic concept of the node, $\text{Tests}_{d,w}$ is the set of test pairs (x, i) generated for this topic, $\text{Result}_{d,w}$ denotes the assessment results of these tests, and $\text{Reflection}_{d,w}$ signifies the LLM's analysis for the node. We denote $f_{d,w}$ as the parent node and $ch_{d,w}$ as the children nodes of $t_{d,w}$. Additionally, several hyperparameters must be set for testing, including the number of test topics generated per iteration n_t, the number of test inputs generated per iteration n_i, and the number of images generated per test input n_x.

Pipeline: As illustrated in Fig. 2, DyEval implements our design objectives through a visually explorable test tree structure: To support *dynamic coverage* (O1), evaluators begin by setting an initial test topic $topic_{0,0}$ (e.g., "DOG" in Fig 2) that serves as the root node for subsequent exploration. The visual interface allows users to flexibly navigate and expand the testing space from this starting point.

Following the *evolutionary adaptability* principle (O2), DyEval constructs testing nodes through an interactive process (Section IV-B). The LLM dynamically generates text inputs I based on the current node topic and accumulated test contexts. The text-to-image model m produces images X for testing, which are visually presented for evaluator assessment against predefined criteria. For nodes with low average pass rates (below threshold ρ), DyEval transitions into a contextual reflection module. This module helps identify failure triggers,

i.e. the minimal input subsets leading to failures, and enables the LLM to analyze possible failure patterns based on related test contexts. All these processes are visually logged and accessible through the interactive interface.

To ensure *anti-leakage design* (O3) and support *dynamic coverage* (O1), DyEval continuously generates new test scenarios rather than relying on fixed test cases. For nodes with high average pass rates, DyEval supports deeper layer exploration (Section IV-C). The LLM generates new test topics dynamically, and evaluators can interactively select topics to pursue, creating unique testing paths that prevent model optimization for specific test cases. This iterative loop continues until the maximum predefined testing depth is reached.

Supporting *visual scalability* (O4), the entire process is organized in a hierarchical test tree structure that allows efficient navigation and analysis at multiple granularities. The visual interface maintains both overview and detailed views of the testing progress, enabling evaluators to track the exploration process until reaching the maximum predefined testing depth while managing complexity through interactive visualizations.

Case study: As illustrated in Fig. 3, we present a complete testing tree for SD3, starting with the initial topic "Spatial relationships". Each node in the tree indicates its specific test topic and the corresponding average pass rate, demonstrating that DyEval can progressively explore the boundaries of SD3. Beginning with "Spatial relationships", DyEval delves into finer-grained testing scenarios such as "Object orientation", "Proximity and distance", and "Size comparison and scaling". This detailed exploration helps identify the model's capabilities and limitations across various test topics. During this test process, we can discover that SD3 encounters challenges in scenarios like "Object orientation in complex environments", "Depicting emotional distance through body language", and "Relative size comparison with animals". Interestingly, the average pass rate of child nodes is not always lower than that of their parent nodes. This occurs because child nodes represent more specific scenarios within the broader topic of the parent node, which the model might handle better. Thus, exploring the model's boundaries involves recognizing both its weaknesses and strengths.

Additionally, Fig. 3 showcases specific test inputs along a testing path. It reveals that when generating child test nodes, DyEval considers the test records of parent nodes. For instance, failures in generating a cat and mouse led to the child topic "Size comparison and scaling", and the successes in generating mountains and villages, contrasted with failures in generating whales and ships, resulted in a new child test topic "Relative size comparison with animals".

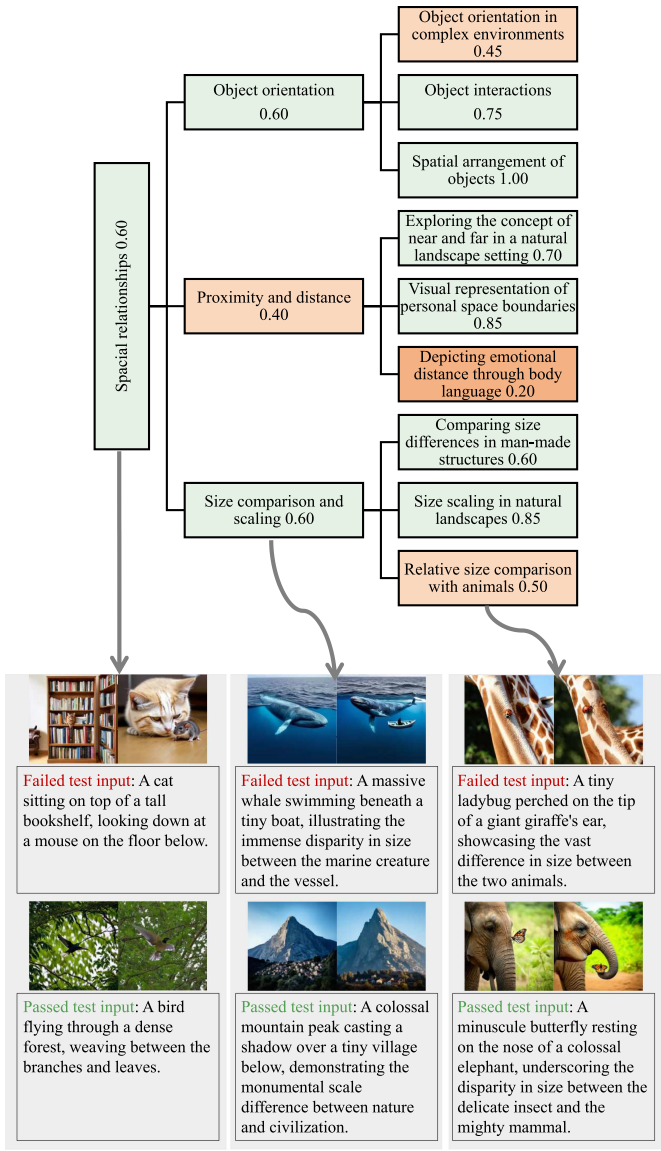


Fig. 3. A test process case of DyEval. The initial test topic is "Spacial relationships" and the APR of this test node is 0.60. Based on the test records of the current topic, LLM continuously generates new test topics to analyze the model capability boundaries further. We also provide specific test inputs in the path of the test tree. Green represents nodes with test pass rate greater than or equal to 0.6, light orange represents pass rate (0.6,0.3], and less than 0.3 is dark orange.

B. Testing Node Construction

1) Testing Prompt Generation: In constructing a testing node, DyEval begins by utilizing the strong text generation ability of LLM to generate n_i text prompts I based on the current topic. As illustrated in Fig. 2, these prompts correspond to the current test topic, such as "DOG-human relationships." An example of a generated prompt is "A dog wagging its tail while its owner scratches its belly." If the test tree T includes additional information related to the current topic, such as parent nodes f, these are also incorporated as test contexts for the LLM. DyEval inspires the LLM to generate test inputs that are semantically aligned with the topic, aiming to identify new potential failures in the model being tested. This can be formulated as:

$$I_{d,w} = \text{InputGen}(t_{o_{d,w}}, f_{d,w}), \tag{4}$$

where $InputGen$ is a process that LLM takes the current topic $t_{o_{d,w}}$ and parent node $f_{d,w}$ as test contexts input and output testing inputs for text-to-image models. The prompt details for this function are provided in the supplementary material. To maintain semantic relevance, we verify the generated test inputs with the LLM and eliminate duplicates using an n-gram matching algorithm.

We realize that relying on LLMs introduces potential biases and limitations. Despite these concerns, LLMs remain the most viable approach for large-scale adaptive testing due to their

unique combination of domain knowledge, reasoning capabilities, and language generation skills. Our human oversight mechanism, where all generated text inputs undergo review, achieved a 99% acceptance rate in experiments, demonstrating effective quality control.

After the generation of test inputs, the tested model m generates n_x images per input, and evaluators will determine pass or fail based on the specific criteria. Given the complexity of image generation tasks, manual evaluation is essential and remains the most reliable method [19]. To reduce the labeling effort, we use CLIPScore [15] as an initial filter; any text-image pair with a CLIPScore below a certain threshold is automatically marked as a fail. To avoid mislabelling these results are also shown to the users, and they can refine as needed. Finally, these test processes are systematically recorded in the test tree T, with updates made to the relevant nodes $t_{d,w}$ within T.

Initial test input complexity control: Initial test input complexity significantly impacts the evaluation trajectory and outcomes in DyEval. We employ text length as the primary complexity metric, constraining initial prompts to within 20 English words to respect the 77-token limit of CLIP encoders in most common text-to-image models while maximizing exploration space.

Our complexity control strategy follows two principles: Starting from semantically simple, single-concept descriptions (e.g., "a red car") that naturally evolve toward multi-attribute compositions, and allowing complexity escalation to be driven by model capabilities rather than predefined rules. These initial test inputs are generated by LLMs following explicit instructions to ensure topic relevance, visual suitability, text length, and progressive difficulty scaling based on evaluation history.

The relationship between initial complexity and evaluation outcomes manifests in two key aspects: higher initial complexity increases reflection module activations and failure case discovery, while the controlled complexity progression enables systematic exploration from basic capabilities to edge cases across different model architectures.

2) Contextual Reflection: For low average pass rate test nodes, we design a contextual reflection module to analyze further possible failure patterns of the text-to-image model m. This module consists of two components: a dynamic failure location that attributes failures to the smallest text component, called a failure trigger, and self-reflection for summarization and analysis by the logical ability of LLM.

Dynamic failure location: When a test test(x, i) fails, it is both costly and highly subjective for evaluators to label failures in detail. However, analyzing failure reasons and exploring the weaknesses of the test model is crucial for understanding and improving the model. To address this, we devise a divide-and-conquer strategy, called dynamic failure location, as illustrated in Fig. 2 and Alg. 1. This approach iteratively breaks down the text input i to pinpoint failure triggers. Initially, the input is split into two halves and tested. If both halves fail, further splitting continues. If both pass, one half is selected for further splitting and merged with the other one separately for testing.

To facilitate precise analysis and control of the text input and ensure semantic coherence after decomposition, we employ an LLM to transform text input i into a scene graph representation c [20]. Scene graph, as a structured representation method, explicitly describes objects, their attributes, and relationships between objects within a scene. Converting visual content into scene graphs represents a crucial goal in visual understanding. The process F_{test} subsequently transforms these scene graphs back into text inputs for evaluating text-to-image models, which is implemented using LLM.

Algorithm 1: Dynamic Failure Location.

Input: Initial scene graph c_0, Split function F_{split}, Test function F_{test},
Output: Process test results T_{mini}.

$T_{mini} \leftarrow \{(c_0, fail)\}$;
$t_{mini} \leftarrow \emptyset$;
$Q \leftarrow \{(c_0, \emptyset)\}$;
while $Q \neq \emptyset$ **do**
  $c, l \leftarrow Q.pop()$;
  if $|c| \neq 1$ **then**
    $c_1, c_2 \leftarrow F_{split}(c)$;
    $test(c_1 \cup l) \leftarrow F_{test}(c_1 \cup l)$;
    $test(c_2 \cup l) \leftarrow F_{test}(c_2 \cup l)$;
    $T_{mini}.append((c_1 \cup l, test(c_1 \cup l)))$;
    $T_{mini}.append((c_2 \cup l, test(c_2 \cup l)))$;
    if $test(c_1 \cup l)$ $fails$ **then**
      $Q \leftarrow Q \cup \{(c_1, l)\}$;
    if $test(c_2 \cup l)$ $fails$ **then**
      $Q \leftarrow Q \cup \{(c_2, l)\}$;
    if $test(c_1 \cup l)$ $passes \wedge test(c_2 \cup l)$ $passes$ **then**
      $Q \leftarrow Q \cup \{(c_1, c_2 \cup l), (c_2, c_1 \cup l)\}$;
    $t_{mini} \leftarrow t_{mini} \cup \{test(c_1 \cup l), test(c_2 \cup l)\}$;
end
return T_{mini};

As detailed in Alg. 1, after converting the test input i into scene graph c_0, dynamic failure location divides it into two subsets, c_1 and c_2. These subsets are then tested by converted into texts to determine if they meet evaluator requirements. If a subset fails, it undergoes further subdivision. If both pass, one is retained as a baseline while the other is further subdivided. All results are stored in the test node t for future reflection. Additional examples and specific task prompts of F_{test} and F_{split}, and other details are available in the supplementary material.

The splitting and merging of scene graphs are implemented through deterministic code operations on JSON-formatted structures, ensuring reliable manipulation independent of LLM variability. We include the pse code in the Supplementary Materials. We also evaluated the quality of LLM-based bidirectional conversions on internal test sets containing 50 samples per task. Text-to-scene-graph conversion achieved 85.7% triplet accuracy (measuring the correctness of extracted entities, attributes, and relations). Scene-graph-to-text conversion achieved 0.49 BLEU-4 score, demonstrating high semantic fidelity in the reverse transformation.

Self-reflection: Leveraging the logical reasoning and analysis ability of LLM [35], [43], DyEval tries to analyze and summarize failure patterns and reasons for tested model m. The reflection, denoted as $r_{d,w}$, is generated by the LLM using the current test node $t_{d,w}$ from the test tree T:

$$\text{Reflection}_{d,w} = \text{Reflect}(t_{d,w}), \tag{5}$$

where $Reflect$ is a process where the LLM uses all available information from the current node t to analyze potential failure patterns. This reflective process allows DyEval to conduct a

thorough analysis of failed test data, offering a deeper understanding of model performance and identifying potential areas for improvement. The specific prompt used is detailed in the supplementary materials.

Minimality guarantees and limitations: Our dynamic failure location algorithm achieves **1-minimality** [44]: the returned failure trigger is minimal in the sense that removing any single component (entity, attribute, or relation node from the scene graph) causes the test to pass. However, we do not guarantee *global minimality*—there may exist other minimal failing configurations of comparable or smaller size.

This follows established test case reduction principles where achieving true global minimality is computationally impractical due to exponential search spaces and complex semantic dependencies. Our divide-and-conquer strategy provides an efficient approximation that balances reduction quality with computational feasibility.

Importantly, distinguishing syntactic minimality from semantic causality remains challenging. The scene graph representation helps by ensuring splits respect semantic boundaries (entities, relations) rather than arbitrary word positions. Additionally, our self-reflection module analyzes patterns across all test records to identify why failures occur, providing causal insight beyond the minimal test case itself. Together, these mechanisms help distinguish between coincidentally minimal inputs and semantically meaningful failure triggers, though human judgment remains essential for final interpretation.

C. Deepening Layer Exploration

For test nodes with a high average pass rate, the next step involves adaptively generating further exploration topics based on the current test context. While labeling text-image pairs is easy for humans, generating new topics is challenging and cost high. Therefore, we offload this creative task to an LLM, as illustrated in Fig. 2 on the right.

Selection principles and criteria: The selection follows two key principles: progressive complexity and diversity. Progressive complexity ensures each deepening layer pushes model boundaries by analyzing previous test records for vulnerabilities. Diversity explores different capability domains to reveal various failure modes. Through carefully designed prompts, the LLM generates increasingly specific test cases targeting model weaknesses.

The capabilities of LLM in language generation, knowledge, and summary analysis are leveraged to generate suggested test topics for the next layer using the current testing node $t_{d,w}$ as context:

$$t_{o_{d+1}} = \text{TopicGen}(t_{d,w}), \tag{6}$$

where $t_{o_{d+1}}$ is a set of n_t test topics for children nodes of $t_{d,w}$. These new topics automatically are created as new nodes in T with parent node $t_{d,w}$ and updated as the new nodes t_{d+1}.

By default, DyEval follows breadth-first exploration based on LLM generation sequence. Evaluators can override automatic selections, specify testing directions based on domain expertise, or request regeneration when topics are insufficient. For instance in Fig. 2, after testing the node on the topic "DOG-human relationships", the LLM suggests a new set of child topics, such as " Interactions between dogs and owners.", "The role of a dog in a family setting.", and "The role of a therapy dog.".

By deepening layer exploration, DyEval facilitates the generation of subsequent test topics and inputs, guided by the evolving test tree T, until reaches a predefined limit of testing loops. This approach broadens the testing scope by suggesting more fine-grained topics related to the current one, enabling evaluators to refine their focus for future tests.

Bias Mitigation in Dynamic Evaluation: A key concern in adaptive testing is the potential for test-generating models to develop bias based on historical feedback, which could lead to narrow exploration of capability boundaries. We addresses this challenge through a multifaceted approach.

DyEval employs balanced breadth-depth exploration that prevents overfitting while maintaining systematic boundary exploration. Although previous test results guide the exploration process, deliberate diversification strategies prevent narrow convergence. Our experimental design incorporates nine initial starting points, with each node automatically generating three distinct topics during depth exploration. This width-expansion approach scales with computational resources, ensuring diverse exploration paths.

Human oversight provides additional bias mitigation. Evaluators can override automatic selections and specify testing directions, ensuring alignment with evaluation objectives and preventing systematic drift toward particular test case types.

Reproducibility: While DyEval improves replicability over purely human-driven adaptive testing by using LLM-generated prompts with documented templates, complete reproducibility remains challenging due to the subjective nature of visual quality assessment. We mitigate this through rigorous annotation protocols (Kendall's $\tau = 0.7746$ for inter-annotator agreement) and will release our complete implementation to enable independent validation and reproduction.

Addressing Circular Reasoning: A potential concern with LLM-driven testing is circular reasoning—where the same model generates and evaluates tests. Our framework avoids this through strict architectural separation: LLMs generate candidate inputs and propose topics, the text-to-image model produces images, and human evaluators make final pass/fail judgments based on visual quality. Test outcomes derive from human perception rather than LLM opinions, ensuring that LLMs propose hypotheses while humans validate them against actual model outputs. This separation, combined with empirical counterfactual testing in our dynamic failure location module (Section IV-B(2)), ensures that failure attributions are grounded in concrete evidence rather than LLM speculation. Additional bias mitigation strategies, including diversification and human oversight mechanisms, are discussed in Section IV-C.

V. Experiments

In this section, we start by introducing experimental settings in Section V-A, and then conduct both qualitative and quantitative experiments on DyEval to answer the following questions: (1) Can DyEval find bugs effectively in the models being tested, especially compared with static methods (Section V-B)? (2) What are the differences and similarities in failures among various text-to-image models (Section V-C)?

A. Experimental Settings

Hyper-parameters of DyEval: We evaluated four major open-source text-to-image models: Stable Diffusion v1-5

(SD1-5) [33], Stable Diffusion v2-1 (SD2-1) [33], Stable Diffusion XL (SDXL) [30], and Stable Diffusion v3 (SD3) [7]. DyEval is fundamentally built on human evaluation, ensuring alignment with human visual perception. Our study involved 23 participants, all proficient in English, without color blindness, and trained for visual quality assessment, prompt adherence evaluation, and failure identification, comprising individuals with diverse backgrounds: 15 with completed university education and 8 current third- or fourth-year undergraduate students (age distribution: 2 over 50, 2 between 30-50, 19 between 20-30; gender: 15 male, 8 female; expertise: 20 in computer science or related fields, 3 in humanities). All participants were proficient in English and had normal color vision.

The training program was structured as a progressive pipeline. Initially, participants received case-based instruction focusing on visual quality, prompt adherence, and failure identification. This was followed by individual sessions with protocol designers to ensure conceptual alignment. The process culminated in a supervised qualification stage, where designers directly monitored each participant's inaugural evaluation node; progression was strictly contingent upon the flawless execution of all prescribed steps. Furthermore, all participants were compensated at a rate exceeding the local minimum hourly wage.

In our participant evaluation process, we randomly reintroduce previous text-image pairs to test annotation consistency. The inter-rater reliability analysis yielded Kendall's tau coefficient of $\tau = 0.7746$ ($p < 0.05$), demonstrating that DyEval's human-based results consistently reflect human visual perception. This rigorous multi-stage training process, combined with our diverse yet technically qualified annotator pool, ensures that DyEval's human assessments are both reliable and aligned with genuine human visual perception.

The parameters used were: number of test topics per iteration $n_t = 3$, number of test inputs per iteration $n_i = 5$, number of images generated per test input $n_x = 4$, and max depth $d_{\max} = 3$. We set the topic stop extension pass rate ρ to 0 to analyze text-to-image models fully. The LLM we used is GPT-3.5-turbo which is developed by OpenAI and used widely in agent and LLM applications [35], [43]. Detailed prompts in the supplementary material.

Assessment criteria: Here, we set the assessment criteria as text-image alignment and visual quality following [14]. A $test(x, i)$ passes if the generated image aligns with the text input i without obvious defects, missing elements, misplacements, or extraneous content that was incongruent with the text input. Conversely, a $test(x, i)$ fails if these criteria are not met. Note that, DyEval is a dynamic interactive testing protocol that can be integrated with any evaluation criteria and assessment methods.

Evaluation metrics: We define the number of unique inputs identified as bugs as **#Bugs**, the **Average Pass Rate (APR)**, and the **Average Fail Rate (AFR)**. These are formalized as follows:

$$\#\text{Bugs} = \sum_{i \in \text{InputsInTree}} \mathbb{I}(\text{TestGroup}(i) \text{ is a bug}), \tag{7}$$

$$\text{APR} = \frac{N_{\text{passes}}}{N_{\text{total\_evals}}}, \tag{8}$$

$$\text{AFR} = 1 - \text{APR}, \tag{9}$$

where N_{passes} is the total number of individual (x, i) evaluations that passed across the entire test tree T, and $N_{\text{total\_evals}}$ is the total number of individual (x, i) evaluations conducted (excluding those within contextual reflection modules). $\mathbb{I}(\cdot)$ is the indicator function, which is 1 if the condition is true, and 0 otherwise. The average rate APR/AFR provides a general performance overview, while the number of poorly performing cases (#Bugs) highlights specific weaknesses. This dual approach allows for a balanced evaluation, ensuring that models are not only generally effective but also robust against challenging scenarios.

Initial topic settings: Building on prior visual assessment frameworks [1], [18], [21], [22], [42] and image scene graph works [20], we encompass a comprehensive assessment from four perspectives: Object, Relation, Attribute, and the global attribute Context, which is crucial for understanding the broader implications of image generation. Here Context refers to the broader setting or environment that influences how objects, relations, and attributes are perceived. It could include cultural, historical, or situational factors that provide additional meaning to the image. Each perspective is further detailed as follows:

- **Object**
  - *Material Objects:* These are tangible entities that exist in the physical world, such as animals, vehicles, and furniture.
  - *Abstract Objects:* These refer to conceptual entities that do not have a physical form, such as emotions, ideas, and philosophies.
- **Relation**
  - *Explicit Relations:* These are directly observable relationships within images, such as spatial arrangements and object interactions.
  - *Implicit Relations:* These require contextual understanding and knowledge, such as social relationships or implied connections.
- **Attribute**
  - *Static Attributes:* These are inherent characteristics of objects that remain constant, such as shape, color, and texture.
  - *Dynamic Attributes:* These describe characteristics that can change over time, such as movement, growth, and transformation.
- **Context**
  - *Style:* This refers to the overall artistic or visual style that influences the perception of the image, such as impressionism or realism.
  - *Culture:* This involves cultural elements that provide background context, such as traditional symbols or practices.
  - *Knowledge:* This encompasses the broader understanding of historical or situational contexts that inform the image's interpretation.

These categories allow for a nuanced evaluation of text-to-image models, highlighting areas where current benchmarks may fall short, particularly in handling abstract concepts, implicit relationships, and cultural nuances.

B. Usefulness of DyEval

In this experiment, we explored nine initial topics: material objects, abstract objects, explicit relations, implicit relations, static attributes, dynamic attributes, style, culture, and knowledge. Participants used DyEval to generate three subtopics for each topic, continuing until they reached a maximum depth of three, and we could collect a maximum of 13 test topics per test tree (1 test node at depth 1, 1×3 test nodes at depth 2, and

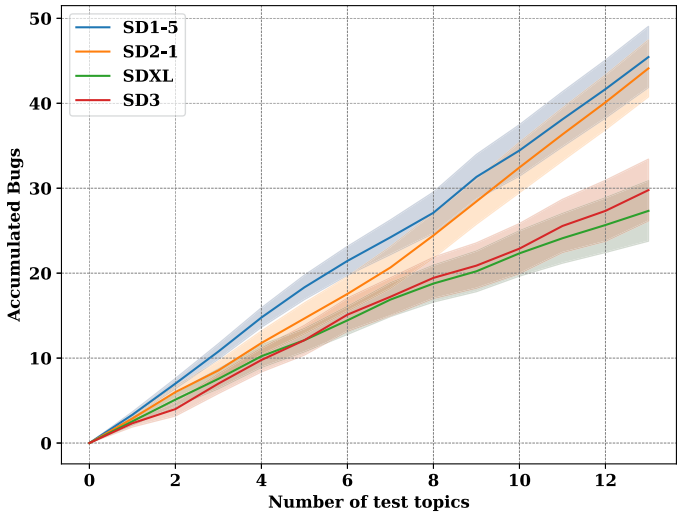


Fig. 4. Average number of bugs accumulated within an initial topic throughout thirteen test nodes (1 + 1×3 + 3×3) during the testing process of DyEval of SD1-5, SD2-1, SDXL, SD3. The standard error is over nine initial topics. The index of the test node of the test tree is obtained according to the breadth-first search. DyEval can constantly find bugs in the model under test, and weaker models are likelier to find bugs (SD1-5, SD2-1 consistently higher than SDXL, SD3). The shading represents the variance.

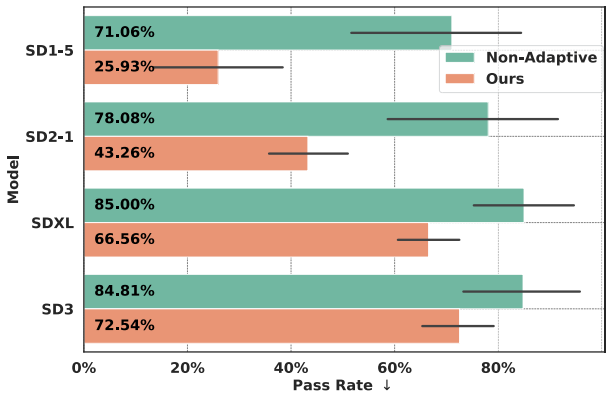


Fig. 5. Comparison experiments between DyEval and non-adaptive testing (LLM directly generates the same amount of text input). The horizontal line represents the variance. Compared to the non-adaptive fashions, DyEval can find more generation failures in assessing the same number of text-image pairs, with greater distinguishability across models and better stability across test aspects.

3×3 test nodes at depth 3). We then cleaned the data to delete user errors, such as duplicates or omissions, and conducted our analysis on this cleaned data.

Accumulated number of bugs: Fig. 4 shows the average accumulated number of bugs found by DyEval during testing, which reveals that DyEval can consistently find bugs in the models. As testing progresses, the variance in #Bugs across different topics increases, as shown by a growing standard deviation. When comparing the final #Bugs and the speed of bug finding, the ranking is clear: SD1-5> SD2-1 > SD3, SDXL. It indicates that SD3 and SDXL have higher generation performance than SD1-5 and SD2-1, and are less prone to trigger bugs. Figs. 5 and 7 further confirm that model performance follows the trend SD3, SDXL > SD2-1 > SD1-5, aligning with current expectations in the field.

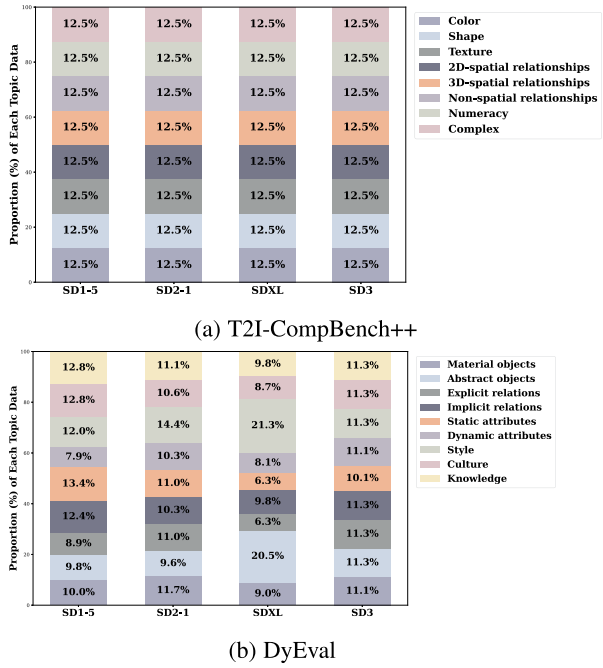


Fig. 6. Comparison of static (T2I-CompBench++) and dynamic (DyEval) test input distributions across models.

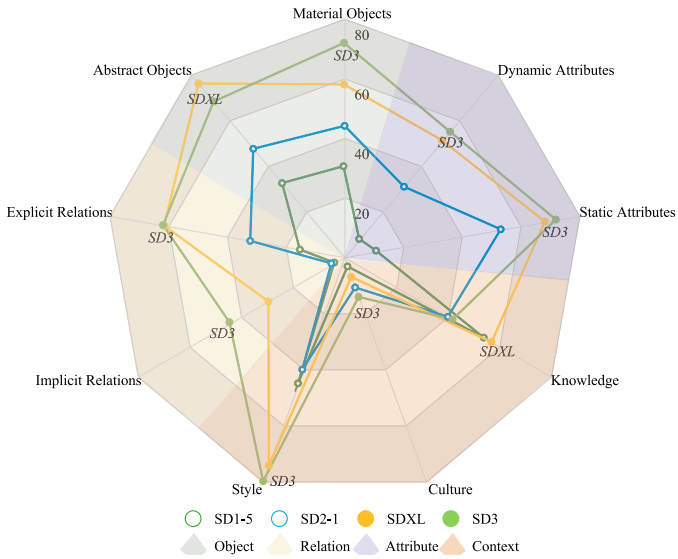


Fig. 7. Average test pass rate APR of SD1-5, SD2-1, SDXL, and SD3 models in nine dimensions. Dimensions of the same base color indicate belonging to the same category. The overall trend of image generation capabilities is SD3>= SDXL> SD2-1> SD1-5; culture and implicit relations are the weakest part of all models. We label the best-performing models in different dimensions with the its name.

Finding 1. Models with superior performance are more challenging to find their bugs: This finding underscores the reliability and usefulness of DyEval, highlighting the inherent challenge of identifying bugs in high-performing models. This insight is crucial for developers aiming to deploy robust text-to-image models in real-world applications, where undetected bugs could lead to significant issues. Notably, while SD3 exhibits a higher

cumulative #Bugs, its overall pass rate is better according to Fig. 7. This indicates that SD3 maintains more consistent performance across different test inputs, whereas SDXL shows more extreme variations in its results—achieving higher success rates on favorable cases but performing poorly on challenging ones. Here, bugs are defined as test inputs with an average pass rate below 0.75, and SD3 has many inputs with pass rates between 0.5 and 0.75.

Comparison with non-adaptive testing: We evaluate the efficacy of the DyEval framework against a non-adaptive testing framework, where the LLM generates the same quantity of text inputs directly (65 be exact, 13 nodes$\times$ 5 test inputs per node, excluding prompts in the contextual reflection module for our method). This evaluation was conducted across four distinct models to assess the failure-finding capabilities of each method. We calculated the average pass rate APR for each model across four relatively objective initial topics: "material objects", "explicit relations", "static attributes" and "style". We avoid selecting "culture", "implicit relationships", or other subjective initial topics due to their complexity and lower human assessment agreement.

As shown in Fig. 5, our experimental results show that when assessing an equal number of text-image pairs, DyEval detects more bugs compared to conventional static evaluation approaches (max up to 2.56 times in SD1-5). Moreover, DyEval finds more differentiated test cases, as the APR difference among SD1-5, SD2-1, SDXL, and SD3 is larger under the DyEval test. This indicates a superior testing efficiency of the DyEval framework in finding model vulnerabilities across different text-to-image models and testing aspects. Finally, DyEval shows a smaller variance across test themes for more stable testing than static testing.

To better demonstrate the differences between our approach and other text-to-image model testing methods, we used the static benchmark T2I-CompBench++ [17] as our baseline, which is also suitable for black-box models and generated through collaboration between LLM and humans, as a static method for comparative analysis. As shown in Fig. 6, T2I-CompBench++ uses the same test set for all text-to-image models, while DyEval continuously adjusts during testing, with the number of test inputs adapting dynamically based on the model and topic being tested.

C. Similarity and Difference Among Different Models

APR of different models across topics: Fig. 7 shows the average pass rate APR of SD1-5, SD2-1, SDXL, and SD3 across nine initial topics. Our experiments reveal consistent patterns in generation performance across different models, with similar relative strengths and weaknesses in specific areas.

In object generation, models demonstrate stronger capabilities with abstract objects compared to material ones. This is because abstract objects rely on conceptual and emotive elements, which are less restricted by physical characteristics and evaluators often assess them based on subjective interpretation and emotional resonance. In contrast, material objects are evaluated against more stringent criteria, emphasizing detail and authenticity grounded in real-world experiences. This disparity in evaluation standards highlights the inherent challenges models face when tasked with generating material objects.

Regarding attributes, models excel at generating static characteristics such as color, shape, and texture, benefiting from abundant training data and their unchanging nature. However, they struggle with dynamic attributes involving temporal changes or movement, revealing limitations in representing time-dependent visual characteristics within single-frame generations.

Furthermore, the models exhibit varying degrees of proficiency in relation generation. They struggle with implicit relations, which points to gaps in their textual comprehension and knowledge integration, particularly in scenarios involving human interactions. Explicit relations are generally handled more effectively, suggesting that models benefit from clear and direct contextual cues.

When it comes to context generation, models are adept at capturing style through visual elements such as lines, colors, and shapes. Knowledge-based content, which necessitates background information, is enhanced by explicit labels and descriptions present in the training data. Nonetheless, cultural content remains a significant challenge due to its complexity, requiring a nuanced understanding of symbols, customs, and traditions across diverse cultural contexts.

Finding 2. Different versions of text-to-image models share consistent performance patterns, with notable weaknesses in culture and implicit relationship handling: These observations underscore the capabilities and limitations of the models, highlighting the need for further research to enhance their understanding and generative performance, particularly in areas rich in cultural nuances and implicit relational contexts.

Part-of-Speech (PoS) Analysis: We analyze test input data from Section V-B excluding data generated by the contextual reflection module, group them by tested model, and divide them into fail and pass sets. After removing stop words and infrequent PoS categories, we calculate the average failure rate AFR for each PoS across models. Fig. 8 displays the PoS rankings by AFR. Please refer to the supplementary material for detailed PoS meanings and distribution across models.

For performance comparison, SD3 and SDXL generally outperform SD2-1 and SD1-5 across various PoS categories. The AFR ranking is SD1-5 $>$ SD2-1 $>$ SDXL $>$ SD3 from highest to lowest. SDXL and SD3 show significant improvement in clause and verb comprehension compared to earlier models, as evidenced by lower AFR in the WRB (Wh-adverb) and VB (Verb) series. SD3 and SDXL exhibit different strengths. SD3 shows lower failure rates in CD, RB, WDT, and CC categories, suggesting stronger text understanding and reasoning abilities. SDXL performs better with NNP, VBP, and WRB, which are often related to human-centric concepts. This aligns with the known weakness of SD3 in body generation.

For numerical and chronological challenges, the CD (cardinal digit) category consistently ranks high in failure rates across all four models. Most failures involve pure numbers or decade terms, indicating significant weaknesses in counting and chronological knowledge.

Finding 3. The failure rate of PRP$(possessive pronoun) ranks high in all three models: PRP$show high failure rates across models. Words like "its", "his", and "their" rely heavily on context and lack specific referents, challenging the models' textual understanding capabilities. Similar issues arise with possessive endings (POS) and certain noun phrases, such as "dog's owner" and "A strong bond of security". The reflections of DyEval also mention: "The model seems to struggle when there is insufficient context provided in the text prompt."

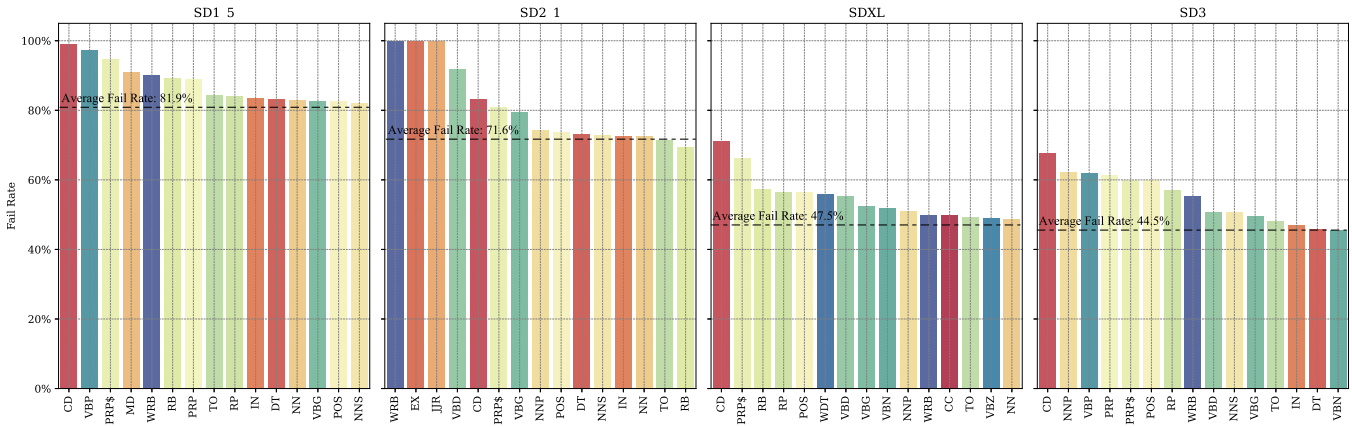


Fig. 8. Top 15 error-prone parts-of-speech from different models, ordered by fail rate. The red dashed line represents the average fail rate for each part of speech in the model under test.

TABLE I
ANALYSIS OF LOW-LEVEL VS. HIGH-LEVEL FEATURE PERFORMANCE ACROSS DIFFERENT TEXT-TO-IMAGE MODELS. "LOW LEVEL AVG" AND "HIGH LEVEL AVG" REPRESENT AVERAGE PASS RATES (%) FOR LOW-LEVEL AND HIGH-LEVEL FEATURES, RESPECTIVELY. "TOP 10" SHOWS THE 10 WORDS WITH THE HIGHEST PASS RATES IN EACH CATEGORY.

Model	Low Level Avg	High Level Avg	Low Level Top 10	High Level Top 10
SD1-5	22.3	17.0	brushstrokes(100.0), colors(45.5), vibrant(43.5), patterns(38.9), carvings(37.5), colorful(33.3), white(29.4), intricate(26.8), black(26.7), blue(21.4)	nature(100.0), architectural(100.0), beauty(100.0), style(100.0), abstract(100.0), cultural(100.0), artwork(83.3), create(77.8), battle(60.0), produce(50.0)
SD2-1	43.9	29.5	brushstrokes(100.0), white(100.0), colors(66.7), vibrant(53.1), color(50.0), lines(50.0), patterns(41.7), intricate(33.3), green(29.2), red(28.0)	visual(100.0), india(100.0), bold(100.0), surface(100.0), painting(100.0), use(100.0), clouds(100.0), used(100.0), movement(100.0), piece(100.0)
SDXL	64.0	54.0	brushstrokes(100.0), swirling(100.0), bold(100.0), geometric(100.0), colors(86.7), color(86.7), white(72.4), patterns(72.2), vibrant(58.8), green(58.6)	intertwining(100.0), elements(100.0), leaves(100.0), forms(100.0), symphony(100.0), incorporating(100.0), pablo(100.0), using(100.0), inspired(100.0), sky(100.0)
SD3	79.5	53.1	patterns(100.0), red(100.0), blue(100.0), green(100.0), color(100.0), texture(100.0), soft(100.0), delicate(100.0), shadows(100.0), metal(100.0)	small(100.0), scattered(100.0), atmosphere(100.0), leaves(100.0), scene(100.0), visual(100.0), serene(100.0), casting(100.0), over(100.0), surface(100.0)

Model performance on low-Level and high-Level features: To further examine failure test input characteristics, we define low-level features as basic visual attributes (color, texture, spatial properties) and high-level semantic aspects as complex conceptual understanding requiring contextual interpretation (object relationships, human actions, cultural concepts). We analyzed test inputs across SD1-5, SD2-1, SDXL, and SD3, statistically ranking words by pass rates and categorizing them through manual filtering into low-level and high-level aspects.

Low-level aspect shows remarkable improvement that SD1-5 achieved 22.3% average pass rate, while SD3 reached 79.5% (Table I). SD3 demonstrates perfect performance on basic colors (red, blue, green, color all at 100%) and excels in texture descriptions (patterns, texture) and visual qualities (soft, delicate, shadows).

High-level semantic features improved more gradually, from 17.0% (SD1-5) to 54.0% (SDXL). Each model exhibits distinct semantic preferences: SD1-5 excels in abstract artistic concepts (nature, beauty, style, abstract all at 100%), while SDXL performs better on compound expressions (intertwining, elements, incorporating all at 100%).

Failure analysis reveals persistent limitations in Table II. SD3 completely fails on certain high-level semantics: traditional, dancers, crops, and group achieve 0% pass rates, indicating systematic difficulties with human actions, group relationships, and cultural concepts. Earlier models showed severe color processing deficiencies, with SD1-5 achieving 0% for red, yellow, and orange, which were resolved in subsequent versions.

Through comparative analysis of success and failure cases, we can precisely characterize the reliable capability boundaries of each model.

Structure analysis: We conducted a comprehensive structural analysis of all test inputs across SD1-5, SD2-1, SDXL, and SD3 models. Using SpaCy's English model, we extracted six linguistic features for each test input: average attributes per sentence (adjectives/adverbs/numbers), predicates per sentence (verbs/auxiliaries/prepositions), objects per sentence (nouns/proper nouns), subordinate clauses, word length,

TABLE II
LOWEST-PERFORMING FEATURES ACROSS DIFFERENT TEXT-TO-IMAGE MODELS. SHOWS THE 10 WORDS (BOTH LOW-LEVEL AND HIGH-LEVEL) WITH THE **LOWEST** PASS RATES (%) IN EACH CATEGORY.

Model	Low Level Bottom 10	High Level Bottom 10
SD1-5	red(0.0), yellow(0.0), orange(0.0), green(11.5), color(15.2), shades(19.0), blue(21.4), black(26.7), intricate(26.8), white(29.4)	dresses(0.0), dancers(0.0), dog(0.0), gowns(0.0), ball(0.0), teacher(0.0), student(0.0), women(0.0), person(0.0), centimeters(0.0)
SD2-1	sleek(0.0), colorful(25.0), red(28.0), green(29.2), intricate(33.3), patterns(41.7), color(50.0), lines(50.0), vibrant(53.1), colors(66.7)	dancers(0.0), group(0.0), playing(0.0), elements(0.0), triangular(0.0), dancing(0.0), over(0.0), overlapping(0.0), hanging(0.0), featuring(0.0)
SDXL	red(40.0), yellow(42.3), black(44.4), blue(44.8), intricate(50.0), green(58.6), vibrant(58.8), patterns(72.2), white(72.4), colors(86.7)	dancers(0.0), group(0.0), owner(0.0), each(0.0), flying(0.0), symbolizing(0.0), other(0.0), dancing(0.0), overlapping(0.0), painting(0.0)
SD3	intricate(43.8), swirling(50.0), vibrant(61.5), colorful(61.9), hues(64.3), wooden(65.0), colors(66.7), patterns(100.0), red(100.0), blue(100.0)	traditional(0.0), dancers(0.0), crops(0.0), group(0.0), hailstones(0.0), playing(0.0), modern(0.0), street(0.0), hands(0.0), being(0.0)

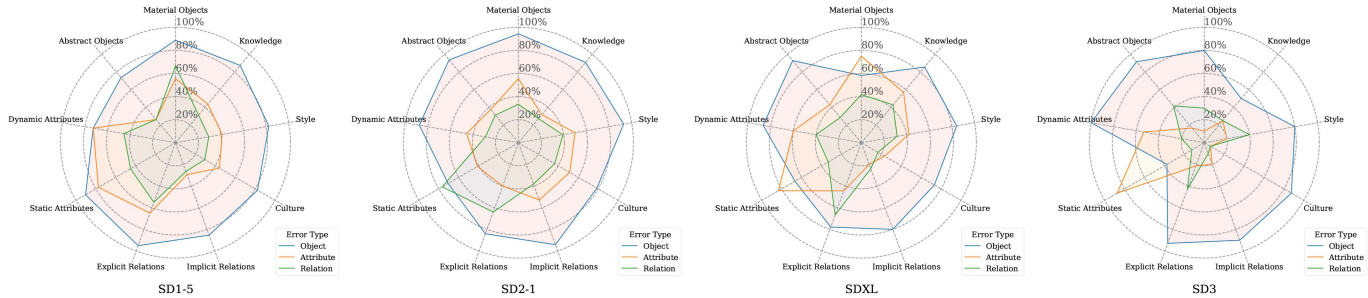


Fig. 9. Frequency of object, attribute, and relation errors in different models.

Fig. 10. An error analysis example mentioned in the paper. Given the test input "A dog and its owner taking a leisurely walk in the park on a bright morning." to test relation generation for SD1-5. While the model correctly generates a leash connecting two subjects, it fails to properly generate the dog and man themselves.

and sentence length. Test inputs were categorized as pass or fail based on model performance outcomes.

The analysis reveals distinct structural sensitivity patterns across different models (Fig. 11). SD2-1 and SDXL demonstrate pronounced vulnerability to structural complexity: failed test inputs in these models contain significantly more predicates (SD2-1: +0.95, SDXL: +0.82) and objects (SD2-1: +1.69, SDXL: +1.14) per sentence, coupled with substantially longer sentences (SD2-1: +3.69, SDXL: +2.71 words) and increased subordinate clause complexity (+0.65 and +0.42, respectively). This pattern suggests these models struggle when processing syntactically dense inputs with multiple semantic elements.

In contrast, SD1-5 exhibits an inverse sensitivity profile where failed test inputs contain fewer attributes (-0.49) and shorter words (-0.39 characters), indicating this model requires more descriptive, attribute-rich inputs to achieve optimal performance. SD3 displays minimal structural differences between successful and failed cases, demonstrating superior robustness to test input complexity variations.

These findings establish clear model-specific guidelines for test inputs optimization: SD2-1 and SDXL perform best with structurally simple test inputs containing fewer objects and predicates, while SD1-5 benefits from detailed, descriptive inputs with rich attributive content. SD3 emerges as the most structurally agnostic model, maintaining consistent performance across diverse test input complexities and representing the most reliable choice for applications requiring varied test input structures.

The influence of initial test input complexity: We conducted systematic evaluation using four topic categories: "Material Objects", "Explicit Relations", "Static Attributes", and "Style" across models SD1-5. For each complexity level (20, 40, and 60 token initial inputs), we generated 5 test cases with width parameter set to 1, exploring generation trees until reaching model token limits. As shown in Table III, results demonstrate a clear correlation between initial text complexity and bug detection effectiveness. As initial complexity increases from 20 to 60 tokens, detected failures rise from 3.25 to 5.00 per layer, while maximum exploration depth decreases from 5 to 2 layers. This indicates that complex initial conditions trigger more frequent bug detection but lead to earlier token exhaustion. The controlled complexity progression enables systematic

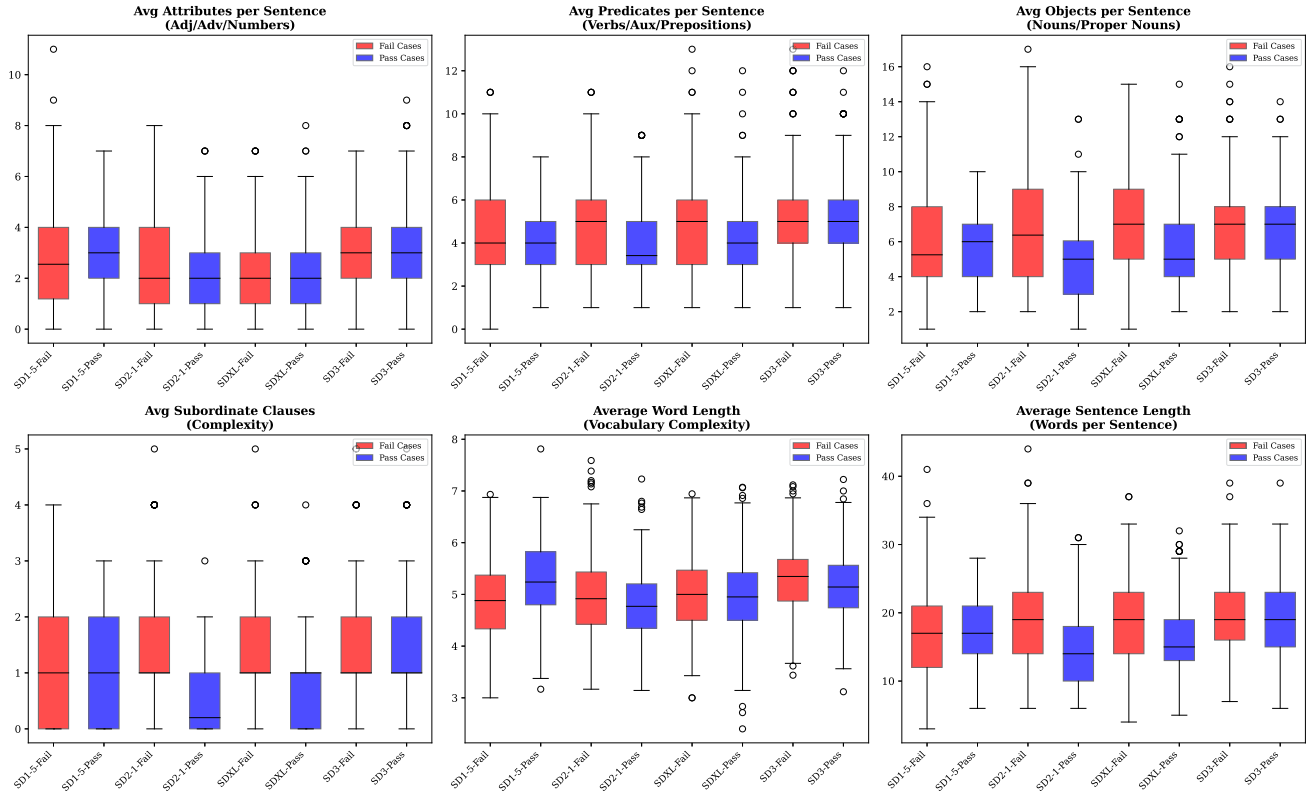


Fig. 11. Structure analysis of pass and failed test inputs.

TABLE III
IMPACT OF INITIAL TEXT COMPLEXITY. ANALYSIS OF DYEVAL OUTCOMES ACROSS VARYING INITIAL TEXT LENGTHS. #INTER REFLECTION/PER LAYER DENOTES THE AVERAGE NUMBER OF REFLECTION MODULE ACTIVATIONS PER GENERATION LAYER. #BUGS/PER LAYER REPRESENTS THE AVERAGE NUMBER OF IDENTIFIED MODEL FAILURES PER LAYER. MAXIMUM DEPTH INDICATES THE DEEPEST GENERATION LEVEL REACHED BEFORE TOKEN LIMIT TERMINATION.

Initial text length	#Inter Reflection/per layer	#Bugs/per layer	Maximum depth
20	3.25	3.25	5
40	4.20	4.20	3
60	5.00	5.00	2

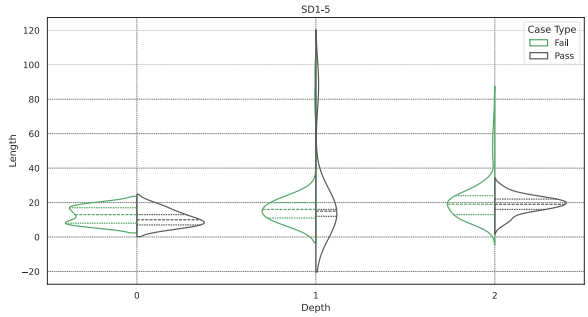
evaluation from basic capabilities to edge cases, revealing that higher initial complexity enhances failure detection sensitivity at the cost of reduced exploration breadth across different model architectures.

Diversity of test topics: We calculated Self-BLEU, a diversity metric in language generation tasks, across all topics, which yielded a 4-gram score of 0.05. This indicates that on average, only 2-4 words per sentence are repeated across different sentences. Considering that function words (such as "a", "the") and topic-related words, this low Self-BLEU score demonstrates high diversity in the generated test cases, indirectly reflecting low overfitting during testing.

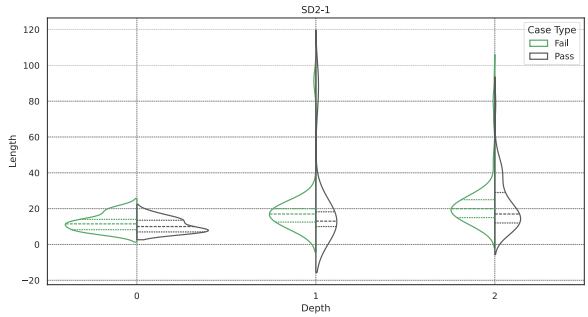
Different types of failures: Based on our analysis and observation of the model failure data and inspired by multimodal hallucination [25], we categorize generation failures into three types: object error, relation error, and attribute error. object error means the generated images contain objects that are poorly generated, missing, or additionally generated objects that are not mentioned in the input text and do not conform to human common sense. Object error is that objects in generated images show poor visual quality, miss, or are not mentioned in the input text, and are harmonious with the overall image content. Relation and attribute errors follow similar definitions.

We conducted hierarchical sampling of 10% from failed text-image pairs, based on the topic and tested model. We then manually annotated specific error types, calculated their frequency and grouped them by model and topic. Two people annotate the same text-image pair and if the results are inconsistent, a third person is introduced. As illustrated in Fig. 9, the advancement in model capabilities has led to a shift in bottleneck issues from objects to attributes to relations. SD1-5 and SD2-1 predominantly fail in generating objects across most topics, while SDXL and SD3 show stronger object depiction capabilities, shifting difficulties to attribute generation. SD3 exhibits fewer attribute errors than SDXL.

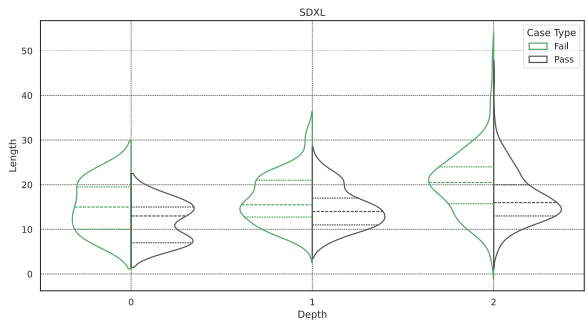
Interestingly, many test inputs in relation topics show higher rates for object error within images than for the relations themselves, also surpassing the rate of object error in the material object topic. For example in Fig. 10, a test input for an implicit relation, "A dog and its owner taking a leisurely walk in the park on a bright morning", may show the man holding the dog's leash (indicating their relationship), yet the dog and



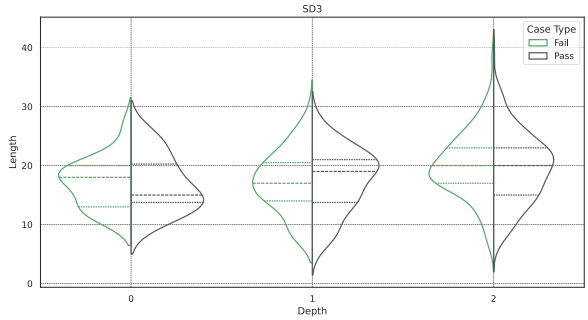
(a)



(b)



(c)

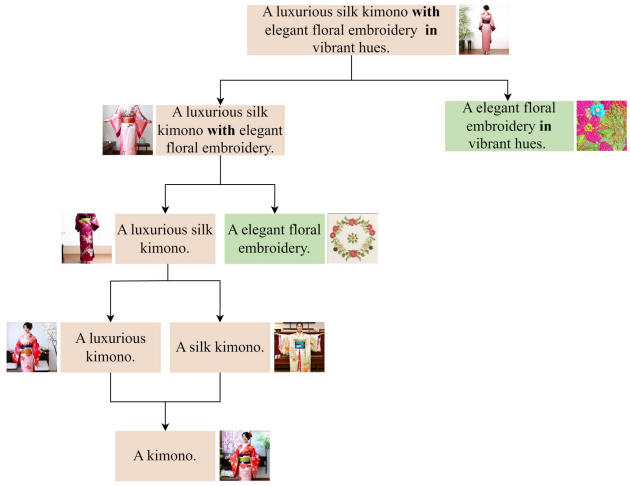


(d)

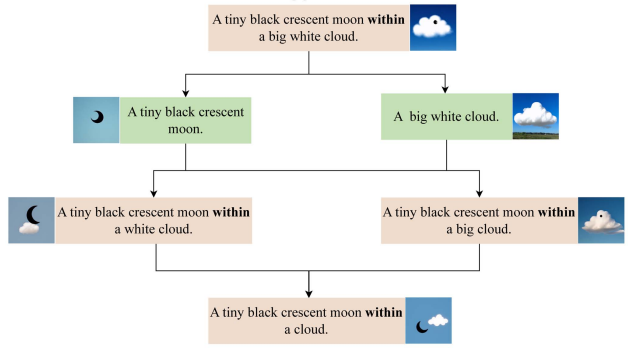
Fig. 12. Violin plot on the length of fail test inputs and pass test inputs in different depths across nine test records for different models. The three dashed lines in each subplot represent, from top to bottom, the upper quartile, median, and lower quartile, respectively.

man themselves are imperfectly generated. This suggests the potential for two-stage generation approaches. Flawed objects could be redrawn to enhance overall generation quality.

Length of test inputs: We analyzed the lengths of all test inputs collected under the nine initial topics for the four models (excluding the contextual reflection module) and presented them in violin plots by depth, as shown in Fig. 12. We can find test



(a) Failure triggers: "A kimono".



(b) Failure trigger: combinatorial failure that "A tiny black crescent moon" and "within a cloud" can not generate successfully in one test input.

Fig. 13. Two examples of dynamic failure location. **Bold** are the relation nodes in their scene graph form. green means test input passes, and brown means test input fails.

input length increases with the depth of DyEval, and failed test inputs are slightly longer than passed test inputs, suggesting that DyEval does not simply find model bugs by generating longer sentences. Test inputs for SDXL and SD3 are slightly longer than those for SD1-5 and SD2-1, with the latter two models showing similar sentence length distributions.

D. Intermediate Process Cases of DyEval

In this section, we show that the contextual reflection module includes an interpretable dynamic failure location and bug analysis capability, which can further enhance the model in the future.

Cases of dynamic failure location: Here we use two cases to show the dynamic failure location can find interpretable failure triggers of tested models in specific test inputs. A case of SD3 as shown in Fig. 13(a), firstly "A luxurious silk kimono with elegant floral embroidery in vibrant hues." fails, and we split it and enter the next iteration. The algorithm is executed until it reaches the smallest cell "A kimono." and fails. In that way, we can find the failure trigger in this test input is "kimono", and this also corresponds to the error-prone generation of SD3 for human body correlation in the previous section.

Fig. 13(b) shows another case of SD3, firstly the model generates "A tiny black crescent moon within a big white cloud."

TABLE IV
CASE STUDY OF REFLECTIONS IN DYEVAL, SHOWING THE INFLUENCE OF DYNAMIC FAILURE LOCATION AND USEFULNESS OF CONTEXTUAL REFLECTION MODULE. HERE 0 MEANS FAIL AND 1 MEANS PASS IN "TEST INPUTS". THE TEXTUAL EXPRESSION IS ABBREVIATED HERE, THE FULL VERSION IS IN THE SUPPLEMENTARY MATERIAL.

Type	Context
Test topic	Dog-human relationships
Test inputs	• A child hugging a dog tightly with a big smile on their face. 1 • A dog lying next to its owner on the couch, both fast asleep. 0 • A dog licking its owner's face to show affection and love. 0 • A dog cuddling up to its owner on a cold winter night, keeping them warm. 0 • A child hugging a dog tightly with a big smile on their face. 1 • A dog sitting patiently next to its owner as they read a book. 1 • A dog jumping up to greet its owner with excitement as they come home. 1 • A dog lying next to its owner on the couch, both fast asleep. 0 • A dog bringing its owner a toy, eagerly waiting for them to play. 1 • A dog licking its owner's face to show affection and love. 0 • A dog following its owner around the house, never wanting to be apart. 1 • A dog barking loudly to protect its owner from a potential threat. 1 • A dog cuddling up to its owner on a cold winter night, keeping them warm. 0
Reflections w/o dynamic failure location	• Lack of Context: Model may fail due to insufficient context. • Misinterpretation of Actions: Difficulty in understanding specific actions. • Lack of Emotional Understanding: Struggles with representing emotions. • Limited Understanding of Relationships: Challenges in capturing nuances of dog-human interactions.
Reflections w/ dynamic failure location	• **Specific Actions**: Model struggles with complex interactions (e.g., "dog licking owner's face"). • **Object Focus**: Better performance with object-oriented inputs (e.g., "A dog's tail"). • **Contextual Relationships**: Difficulty in representing complex scenarios (e.g., "dog cuddling owner on cold night"). • **Incomplete Descriptions**: Surprisingly good performance with general concepts. • **Owner Focus**: Consistent struggles when input focuses solely on the dog's owner.

failed. Then we split it into two sentences, and find they all pass. According to our hypothesis, the limits of the capabilities of the tested models must be in their intersection sets. Thirdly, we split "A big white cloud." to "A big cloud." and "A white cloud.", and merge them with "A tiny black crescent moon." respectively, and they are all failed. Following the algorithm, we continue to split "A big cloud." and "A white cloud." to "A cloud" by deleting attributes and merge with "A tiny black crescent moon.", and it still fails. "A cloud" reaches the smallest cell and the process ends. This is a combinatorial error, "A tiny black crescent moon" and "within a cloud" can not generate successfully in one test input, which can also be found in dynamic failure locations.

We provide the full process with details of the scene graph spliting and merging in the supplementary material.

Cases of self-reflections: As shown in Table IV, LLM can indeed do some initial reflective summaries for the user assessments, and with the addition of dynamic failure location LLM's reflections are richer, longer, and more accurate in identifying mistakes. The full version is in the supplementary material due to space constraints here.

We further conducted a survey among 10 domain experts specializing in text-to-image generation systems to evaluate the effectiveness of reflection contents in DyEval. Each expert was presented with one randomly selected reflection content from DyEval test nodes (an example can be found in the Reflections with dynamic failure location in Table IV). All experts (100%) unanimously confirm that the reflections provide valuable and insightful content for model improvement and accurately correspond to the test records. Although some experts note the presence of redundant information, the majority (80%) emphasize that the structured analysis offers actionable feedback for enhancing model performance.

Reliability validation of LLM's failure analysis: To ensure the reliability of our LLM-based failure analysis, we conducted a comprehensive validation study with domain experts. We recruited 9 professional researchers specializing in text-to-image generation to evaluate the accuracy of LLM failure analysis outputs. We randomly sampled 30 LLM analysis outputs from our collected failure cases. Each evaluator assessed 10 samples, with every analysis independently evaluated by three different experts to ensure reliability. All evaluators received standardized training on evaluation criteria, including failure type classification accuracy and analysis quality.

Evaluators provided binary correct/incorrect classifications for each LLM analysis output. The validation results show that our LLM-based failure analysis achieves 78% accuracy, indicating that most analytical insights align with expert judgment. Inter-rater reliability, measured by Cohen's kappa coefficient, reached 0.59, indicating substantial agreement among evaluators. While not perfect, these results demonstrate that the LLM provides sufficiently reliable insights for identifying failure patterns and guiding improvements in text-to-image generation systems.

E. Multi-LLM Validation and Bias Mitigation.

We conducted multi-LLM validation experiments to identify and mitigate three types of bias inherent in single-LLM approaches: (1) input preference bias, the tendency to generate inputs following model-specific patterns; (2) exploration bias, the preference for certain topic directions while neglecting others;

TABLE V
MULTI-LLM VALIDATION RESULTS ACROSS DYEVAL'S THREE MODULES ON SD2-1. MULTI-LLM ENSEMBLE MITIGATES INPUT PREFERENCE, EXPLORATION, AND ATTRIBUTION BIASES.

Module & Metric	Single-LLM	Multi-LLM	Improvement
Input Generation (Input Preference Bias)			
Self-BLEU-4(↓)	0.28	0.21	↓0.07
AFR(↑)	65%	83%	↑18%
Topic Generation (Exploration Bias)			
Self-BLEU-4(↓)	0.32	0.23	↓0.09
AFR(↑)	65%	84%	↑19%
Reflection Analysis (Attribution Bias)			
Accuracy(↑)	78%	89%	↑11%

and (3) attribution bias, the identification of spurious patterns not grounded in actual failures.

We employed an ensemble of GPT-4, GPT-3.5 (original setting), and Claude-3.5 Sonnet across DyEval's three modules. The tested text-to-image model is SD2-1. We used 10 topics for input generation, 10 contexts for topic expansion, and 30 failure cases for reflection. For multi-LLM generation, each model independently produces outputs, which are then filtered and merged, with GPT-3.5 selecting the top 5 inputs, top 3 topics, and top 10 reflection patterns based on quality criteria. Bias mitigation was quantified using Self-BLEU-4 (diversity) and Average Fail Rate (AFR, bug discovery effectiveness).

Table V summarizes our multi-LLM validation results across three modules.

- *Input Generation:* Multi-LLM ensemble reduced Self-BLEU-4 from 0.28 to 0.21, mitigating input preference bias, with AFR improving from 65% to 83%.
- *Topic Generation:* Multi-LLM approach reduced Self-BLEU-4 from 0.32 to 0.23, addressing exploration bias, with AFR improving from 65% to 84%.
- *Reflection Analysis:* Multi-model consensus improved pattern accuracy from 78% to 89%, effectively filtering attribution bias.

F. Evaluation on Different Architectures

Auto-regressive-based model: To demonstrate the generalizability of our evaluation framework across different T2I architectures, we evaluate SimplerAR-1.5B-RL [36] as a representative auto-regressive approach. Using the same evaluation topics as the diffusion models, we observe that SimplerAR exhibits the weakest performance in culture and implicit relations (Fig. 14), consistent with diffusion-based and flow-based models. However, it performs relatively better in style generation among the evaluated dimensions.

Commercial model: We further evaluate Nano Banana Pro [13] to assess our framework's applicability to commercial systems. As shown in Fig. 14, while Nano Banana Pro achieves the best overall performance across all tested models, it still exhibits weaknesses in culture and implicit relations—a pattern consistent across diffusion, flow, and AR architectures. This recurring limitation suggests that training data composition, rather than architectural design, may be the primary factor constraining performance in these challenging dimensions.

VI. CONCLUSION AND DISCUSSION

We present DyEval, an interactive visual assessment framework that addresses the limitations of static datasets in evaluating rapidly evolving text-to-image models. Through extensive experiments, we demonstrate that our LLM-powered approach significantly outperforms traditional evaluation methods by dynamically generating diverse test cases and providing interpretable insights into model behaviors. The visual analytics components of DyEval enable users to effectively explore and understand complex failure patterns, particularly in challenging scenarios involving cultural nuances, implicit relations, quantifiers, and pronouns.

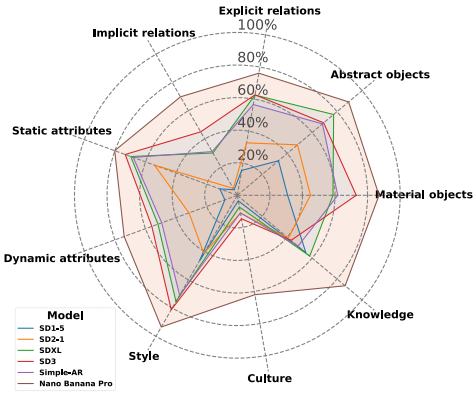


Fig. 14. Average test pass rate APR of SD1-5, SD2-1, SDXL, SD3, SimpleAR, and Nano Banana models in nine dimensions. Dimensions of the same base color indicate belonging to the same category. The overall trend of image generation capabilities is Nano Banana Pro> SD3>= SDXL>=SimpleAR> SD2-1> SD1-5; culture and implicit relations are the weakest part of all models.

Our framework makes several key contributions to the visual analytics and computer graphics communities: it introduces a novel approach to adaptive model assessment that can evolve alongside advancing generation technologies; what's more, it demonstrates the effectiveness of combining LLM capabilities with interactive visualization for comprehensive model evaluation, and finally it provides a flexible foundation for analyzing various aspects of visual generation models.

Scope and Use Cases: DyEval is designed as a precision tool for *boundary discovery* rather than a simple leaderboard-style evaluation. It is most effective in scenarios requiring domain-specific stress testing—such as evaluating spatial reasoning or cultural nuances where static datasets lack coverage—and *model debugging*, where developers need to pinpoint "failure triggers" (Section IV-B(2)) to understand the root causes of systematic errors.

Limitations: While DyEval offers deep insights into model behavior, its effectiveness and scalability are subject to several constraints. First, the framework's *human-in-the-loop design* requires active evaluator participation for visual quality assessment. While adhering to the principle that human judgment remains the gold standard, this requirement may limit throughput in ultra-large-scale evaluations compared to fully automated pipelines. Second, *computational overhead* poses practical challenges: the iterative process of prompt generation, image synthesis, and contextual reflection incurs higher API costs and latency than static inference approaches. Third, the framework exhibits *LLM dependency*, as the effectiveness of exploration is partially bounded by the underlying LLM's reasoning capability, which may introduce biases in test case generation. Moreover, LLM-generated prompts present replicability challenges due to potential model updates. Fourth, our dynamic failure location algorithm achieves 1-minimality but cannot guarantee global minimality. When multiple concepts individually pass testing,

but their combination triggers failure through emergent interaction effects that cannot be attributed to any decomposable subset, our method may not precisely isolate the root cause. Distinguishing syntactic reduction from semantic causality also requires human interpretation.

Future work: To address these limitations and enhance the framework's capabilities, we plan several extensions. To improve scalability, we will investigate *parallel tree exploration* strategies to increase throughput and integrate Vision-Language Models (VLMs) as cost-effective alternatives for preliminary test-case analysis and generation, potentially reducing both LLM bias and computational costs. To strengthen evaluation reliability, we aim to develop more robust protocols, including cost-effective methods for generating multiple samples per prompt and statistical significance tests to quantify evaluation confidence intervals. We also plan to develop professional datasets for instruction tuning and explore multi-model ensemble approaches to further mitigate individual LLM biases. Beyond the current focus on text-image alignment and visual quality, the framework's extensible design enables systematic expansion to encompass broader evaluation criteria, including creativity, style consistency, safety, and diversity. As text-to-image generation metrics become more advanced, we will incorporate them into DyEval to reduce manual evaluation effort while maintaining assessment quality. Finally, we envision extending the framework to evaluate other visual generation tasks, particularly text-to-video generation, and developing automated bug repair mechanisms to enable self-improving generation models, and exploring complementary approaches, such as contrastive testing or causal analysis methods to address the limitation of the divide-and-conquer-based failure location.

References

[1] E. M. Bakr, P. Sun, X. Shen, F.F. Khan, L.E. Li, and M. Elhoseiny, "HRS-Bench: Holistic, reliable and scalable benchmark for text-to-image models," in *Proc. IEEE/CVF Int. Conf. Comput. Vis.*, 2023, pp. 20041–20053.

[2] J. Betker et al., "Improving image generation with better captions," *Comput. Sci.*, pp. 7454–7464, 2023. [Online]. Available: https://cdn.openai.com/papers/dall-e-3.pdf

[3] A. Borji, "Qualitative failures of image generation models and their application in detecting deepfakes," *Image Vis. Comput.*, vol. 137, p. 104771, 2023.

[4] Y. Chen, "X-IQE: Explainable image quality evaluation for text-to-image generation with visual large language models," 2023, *arXiv:2305.10843*.

[5] J. Cho et al., "Davidsonian scene graph: Improving reliability in fine-grained evaluation for text-image generation," in *Proc. 12th Int. Conf. Learn. Representations*, 2024, pp. 54710–54730.

[6] C. Du, Y. Li, Z. Qiu, and C. Xu, "Stable diffusion is unstable," in *Proc. 37th Conf. Neural Inf. Process. Syst.*, 2023, pp. 58648–58669. [Online]. Available: https://openreview.net/forum?id=tesBViWnbx

[7] P. Esser et al., "Scaling rectified flow transformers for high-resolution image synthesis," in *Proc. 41st Int. Conf. Mach. Learn.*2024, pp. 12606–12633.

[8] S. Eyuboglu et al., "Domino: Discovering systematic errors with cross-modal embeddings," in *Proc. Int. Conf. Learn. Representations*, 2022, pp. 295–305. [Online]. Available: https://openreview.net/forum?id=FPCMqjI0jXN

[9] Y. Feng et al., "PromptMagician: Interactive prompt engineering for text-to-image creation," *IEEE Trans. Vis. Comput. Graph.*, vol. 30, no. 1, pp. 295–305, Jan. 2024, doi: 10.1109/TVCG.2023.3327168.

[10] D. Ganguli et al., "Red teaming language models to reduce harms: Methods, scaling behaviors, and lessons learned," 2022, *arXiv:2209.07858*.

[11] I. Gao, G. Ilharco, S. Lundberg, and M.T. Ribeiro, "Adaptive testing of computer vision models," in *Proc. IEEE/CVF Int. Conf. Comput. Vis.*, Oct. 2023, pp. 4003–4014.

[12] X. Gao, Y. Yang, Z. Xie, S. Du, Z. Sun, and Y. Wu, "GUESS: Gradually enriching synthesis for text-driven human motion generation," *IEEE Trans. Vis. Comput. Graph.*, vol. 30, no. 12, pp. 7518–7530, Dec. 2024, doi: 10.1109/TVCG.2024.3352002.

[13] Google, "Nano banana (Gemini 2.5 flash image) technical documentation," 2025. Accessed: Jan. 14, 2026. [Online]. Available: https://ai.google.dev/gemini-api/docs/image-generation

[14] S. Hartwig et al., "A survey on quality metrics for text-to-image generation," *IEEE Trans. Vis. Comput. Graph.*, vol. 31, no. 10, pp. 9464–9483, Oct. 2025.

[15] J. Hessel, A. Holtzman, M. Forbes, R. Le Bras, and Y. Choi, "ClipScore: A reference-free evaluation metric for image captioning," in *Proc. Conf. Empirical Methods Natural Lang. Process.*, 2021, pp. 7514–7528.

[16] Y. Hu et al., "TIFA: Accurate and interpretable text-to-image faithfulness evaluation with question answering," in *Proc. IEEE/CVF Int. Conf. Comput. Vis.*, 2023, pp. 20406–20417.

[17] K. Huang, C. Duan, K. Sun, E. Xie, Z. Li, and X. Liu, "T2I-CompBench++: An enhanced and comprehensive benchmark for compositional text-to-image generation," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 47, no. 5, pp. 3563–3579, May 2025.

[18] K. Huang, K. Sun, E. Xie, Z. Li, and X. Liu, "T2i-compbench: A comprehensive benchmark for open-world compositional text-to-image generation," in *Proc. Adv. Neural Inf. Process. Syst.*, 2024, vol. 36, pp. 78723–78747.

[19] D. Jiang et al., "GenAI-Arena: An open evaluation platform for generative models," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 37, 2024, pp. 79889–79908.

[20] J. Johnson et al., "Image retrieval using scene graphs," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2015, pp. 3668–3678.

[21] Y. Kirstain, A. Polyak, U. Singer, S. Matiana, J. Penna, and O. Levy, "Pick-a-Pic: An open dataset of user preferences for text-to-image generation," in *Proc. 37th Conf. Neural Inf. Process. Syst.*, 2023, pp. 36652–36663. [Online]. Available: https://openreview.net/forum?id=G5RwHpBUv0

[22] T. Lee et al., "Holistic evaluation of text-to-image models," in *Proc. 37th Conf. Neural Inf. Process. Syst. Datasets Benchmarks Track*, 2023, pp. 69981–70011. [Online]. Available: https://openreview.net/forum?id=qY9LR74O3Z

[23] C. Li et al., "AGIQA-3 K: An open database for AI-generated image quality assessment," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 34, no. 8, pp. 6833–6846, Aug. 2024, doi: 10.1109/TCSVT.2023.3319020.

[24] T.Y. Lin et al., "Microsoft COCO: Common objects in context," in *Proc. Comput. Vis.*, 2014, pp. 740–755.

[25] F. Liu, K. Lin, L. Li, J. Wang, Y. Yacoob, and L. Wang, "Mitigating hallucination in large multi-modal models via robust instruction tuning," in *Proc. 12th Int. Conf. Learn. Representations*, 2023, pp. 56626–56670.

[26] N. Liu, S. Li, Y. Du, A. Torralba, and J. B. Tenenbaum, "Compositional visual generation with composable diffusion models," in *Proc. Eur. Conf. Comput. Vis.*, 2022, pp. 423–439.

[27] Y. Lu, X. Yang, X. Li, X.E. Wang, and W.Y. Wang, "Llmscore: Unveiling the power of large language models in text-to-image synthesis evaluation," in *Proc. Adv. Neural Inf. Process. Syst.*, 2023, vol. 36, pp. 23075–23093.

[28] I. Magar and R. Schwartz, "Data contamination: From memorization to exploitation," in *Proc. 60th Annu. Meeting Assoc. Comput. Linguistics*, vol. 2, 2022, pp. 157–165.

[29] Y. Oren, N. Meister, N.S. Chatterji, F. Ladhak, and T. Hashimoto, "Proving test set contamination for black-box language models," in *Proc. 12th Int. Conf. Learn. Representations*, 2024, pp. 56589–56607. [Online]. Available: https://openreview.net/forum?id=KS8mIvetg2

[30] D. Podell et al., "SDXL: Improving latent diffusion models for high-resolution image synthesis," in *Proc. 12th Int. Conf. Learn. Representations*, 2024, pp. 22275–22287. [Online]. Available: https://openreview.net/forum?id=di52zR8xgf

[31] M.T. Ribeiro and S. Lundberg, "Adaptive testing and debugging of NLP models," in *Proc. 60th Annu. Meeting Assoc. Comput. Linguistics*, 2022, pp. 3253–3267.

[32] M.T. Ribeiro, T. Wu, C. Guestrin, and S. Singh, "Beyond accuracy: Behavioral testing of NLP models with CheckList," in *Proc. 58th Annu. Meeting Assoc. Comput. Linguistics*, D. Jurafsky, J. Chai, N. Schluter, and J. Tetreault, Eds., Jul. 2020, pp. 4902–4912, doi: 10.18653/v1/2020.acl-main.442. [Online]. Available: https://aclanthology.org/2020.acl-main.442

[33] R. Rombach, A. Blattmann, D. Lorenz, P. Esser, and B. Ommer, "High-resolution image synthesis with latent diffusion models," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2022, pp. 10684–10695.

[34] C. Saharia et al., "Photorealistic text-to-image diffusion models with deep language understanding," in *Proc. Adv. Neural Inf. Process. Syst.*, 2022, vol. 35, pp. 36479–36494.

[35] N. Shinn, F. Cassano, A. Gopinath, K. Narasimhan, and S. Yao, "Reflexion: Language agents with verbal reinforcement learning," in *Proc. Adv. Neural Inf. Process. Syst.*, 2023, vol. 36, pp. 8634–8652.
[36] J. Wang et al., "SimpleAR: Pushing the frontier of autoregressive visual generation through pretraining, SFT, and RL, 2025, *arXiv:2504.11455*.
[37] Z.J. Wang, E. Montoya, D. Munechika, H. Yang, B. Hoover, and D. H. Chau, "DiffusionDB: A large-scale prompt gallery dataset for text-to-image generative models," in *Proc. 61st Annu. Meeting Assoc. Comput. Linguistics*, 2023, pp. 893–911.
[38] O. Wiles, I. Albuquerque, and S. Gowal, "Discovering bugs in vision models using off-the-shelf image generation and captioning," in *Proc. NeurIPS ML Saf. Workshop*, 2022, pp. 1–20.
[39] Q. Wu et al., "Harnessing the spatial-temporal attention of diffusion models for high-fidelity text-to-image synthesis," in *Proc. IEEE/CVF Int. Conf. Comput. Vis.*, 2023, pp. 7766–7776.
[40] X. Wu et al., "Human preference score V2: A solid benchmark for evaluating human preferences of text-to-image synthesis," 2023, *arXiv:2306.09341*.
[41] C. Xu, Y. Xu, H. Zhang, X. Xu, and S. He, "DreamAnime: Learning style-identity textual disentanglement for anime and beyond," *IEEE Trans. Vis. Comput. Graph.*, vol. 31, no. 8, pp. 4198–4209, Aug. 2025, doi: 10.1109/TVCG.2024.3397712.
[42] J. Xu et al., "ImageReward: Learning and evaluating human preferences for text-to-image generation," in *Proc. Adv. Neural Inf. Process. Syst.*, 2023, vol. 36, pp. 15903–15935.
[43] S. Yao et al., "ReAct: Synergizing reasoning and acting in language models," in *Proc. 11th Int. Conf. Learn. Representations*, 2022, pp. 30084–30116.
[44] A. Zeller, "Yesterday, my program worked. today, it does not. why?," *ACM SIGSOFT Softw. Eng. Notes*, vol. 24, no. 6, pp. 253–267, 1999.
[45] Z. Zhang, J. Chen, H. Fu, J. Zhao, S.Y. Chen, and L. Gao, "Text2face: Text-based face generation with geometry and appearance control," *IEEE Trans. Vis. Comput. Graph.*, vol. 30, no. 9, pp. 6481–6492, Sep. 2024, doi: 10.1109/TVCG.2023.3349050.
[46] Z. Zhou et al., "A comprehensive evaluation of arbitrary image style transfer methods," *IEEE Trans. Vis. Comput. Graph.*, vol. 31, no. 9, pp. 5668–5686, Sep. 2025, doi: 10.1109/TVCG.2024.3466964.

Xiaoyue Mi is currently working toward the PhD degree with the Institute of Computing Technology, Chinese Academy of Sciences, Beijing, China. Her research interests include adversarial defense, continual learning, and AIGC evaluation.

Fan Tang received the BSc degree in computer science from North China Electric Power University, Beijing, China, in 2013, and the PhD degree from the Institute of Automation, Chinese Academy of Sciences, Beijing, in 2019. He is currently an associate professor with the Institute of Computing Technology, Chinese Academy of Sciences. His research interests include computer graphics, computer vision, and machine learning.

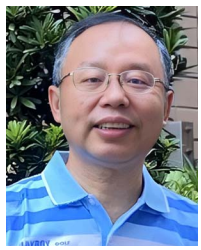
Juan Cao received the PhD degree from the Institute of Computing Technology, Chinese Academy of Sciences, in 2008. She is currently a Professor with the Institute of Computing Technology, Chinese Academy of Sciences. She has more than 90 publications in international journals and conferences, including *IEEE Transactions on Knowledge and Data Engineering*, *IEEE Transactions on Image Processing*, KDD, and CVPR. Her research interests include multimedia content analysis and fake multimedia detection.

Qiang Sheng received the PhD degree from the Chinese Academy of Sciences, in 2023. He is currently an associate professor with the Institute of Computing Technology, Chinese Academy of Sciences. He has authored or coauthored more than 20 papers in conferences and journals including ACL, *IEEE Transactions on Knowledge and Data Engineering*, SIGIR, WWW, and MM. His research interests include fake news detection, fact-checking, and large language model safety.

Ziyao Huang received the BE degree from Beihang University, in 2017 and the ME degree, in 2022, from the Institute of Computing Technology, Chinese Academy of Sciences, Beijing, China, where he is currently working toward the PhD degree with the Institute of Computing Technology. His research interests include digital human and video generation.

Peng Li received the PhD degree in computer science and technology from Tsinghua University, China. He was the principal researcher and team leader with WeChat AI, Tencent. He was with the Institute of Deep Learning, Baidu Inc. He is currently a research associate professor with the Institute for AI Industry Research (AIR), Tsinghua University, China. His research interests include natural language processing, pre-trained models, and multimodal models. He was the recipient of ACL 2023 Outstanding Paper Award and First Prize of Qian Weichang Chinese Information Processing Science and Technology Award.

Yang Liu (Senior Member, IEEE) received the PhD degree from the Institute of Computing Technology, Chinese Academy of Sciences, Beijing, China, in 2007. He is currently a tenured full professor with the Department of Computer Science and Technology, Tsinghua University, Beijing, China. His research interests include natural language processing and machine translation.



Tong-Yee Lee (Senior Member, IEEE) received the PhD degree in computer engineering from Washington State University, Pullman, in 1995. He is currently the chair professor with the Department of Computer Science and Information Engineering, National Cheng-Kung University, Tainan, Taiwan. He also leads the Computer Graphics Group, Visual System Laboratory, National Cheng-Kung University (http://graphics.csie.ncku.edu.tw). His research interests include computer graphics, nonphotorealistic rendering, medical visualization, virtual reality, and media resizing. He is a senior member of the IEEE Computer Society and member of ACM. He is on the editorial boards of *IEEE Transactions on Visualization and Computer Graphics* and *IEEE Computer Graphics and Applications*.