A. Appendix

This appendix provides additional details to complement the main text. We elaborate on the method, including the prompt template (A.1.1), the automatic evaluation protocol (A.1.2), and the design of the prioritization model (A.1.3). We also present details of experiments introduced in the main paper, covering implementation details (A.2.1), additional visualizations, case studies (A.2.2), bias analysis (A.2.3). Social impact is discussed in A.4.

Our supplementary materials also include (1) code for reproduce error slices presented in the paper, and (2) the full entity–attribute corpus used in exploration. Full implementation and error slice discovery results of four T2I models will be released upon acceptance.

A.1. Method Details

A.1.1. Prompt Template

Each node is instantiated with a natural language prompt using predefined templates with three modular components:

- **Base description**, which combines entity name with quantity and descriptive attributes, e.g., *"three small red birds"*. Multiple attributes are concatenated with commas and conjunctions.
- **Action description**, which converts action-related attributes into short clauses, e.g., *"flying upward at accelerating speed"*, supporting both active and passive forms.
- **Background description**, which expresses environmental attributes (e.g., weather, lighting, time) as full sentences, e.g., *"The background is cloudy. The time is night."*

These components are concatenated to form the full prompt, for example: *"An image of three small red birds flying upward at accelerating speed. The background is cloudy. The time is night."*

An LLM (Qwen2.5-14B-Instruct) is used only for minor grammatical corrections, such as:

- Pluralization (*"2 cat"* $\rightarrow$ *"two cats"*)
- Article insertion (*"red apple"* $\rightarrow$ *"a red apple"*)
- Verb tense consistency (*"is fly"* $\rightarrow$ *"is flying"*)
- Smoother conjunctions for multiple attributes

This two-step design ensures consistent and fluent prompts while avoiding uncontrolled expansion.

A.1.2. Multi-Choice Questions for Automatic Evaluation

To evaluate generated images, we design a multi-choice question framework where both the entity and its attributes are verified by a multimodal large language model (MLLM). For each attribute or entity, up to five visually similar alternatives are sampled from the same subcategory, and two universal options ("Others" and "Can not answer") are appended. The first question always concerns the entity, and subsequent questions target attributes such as quantity, color, or background, each presented in a structured multi-choice format.

The MLLM is instructed to select the closest answers based on the image, while following strict rules: choosing "Can not answer" if the object is distorted or ambiguous, and choosing "Others" if none of the listed options match the visual evidence. In addition to entity recognition, this framework also enforces attribute-level consistency, with outputs represented as both a short natural language description and explicit multi-choice decisions. The generation success rate of a node is then defined as the proportion of answers that align with the intended entity and attribute labels. The evaluation of this method is presented in A.2.3.

A.1.3. Model for Search Prioritization

The accuracy predictor is designed as a lightweight transformer decoder that refines entity embeddings with attribute information. Each layer consists of self-attention on the entity embedding, cross-attention with attribute embeddings, and a feed-forward network, all connected through residual links and pre-layer normalization. Entities and attributes are first encoded with a pretrained text encoder, and missing attributes at shallower depths are padded with zeros to ensure consistent input structure. The final entity representation is passed through a two-layer feed-forward head with a sigmoid activation, producing a scalar in [0,1] that estimates the node's success rate. The model is trained with an L1 loss with the observed outcomes. During error slice discovery, the predictor is updated online with evaluation results and used to prioritize nodes predicted to have high failure likelihood, enabling efficient exploration under limited budgets.

A.1.4. Validation of the Assumption in Pruning

Pruning in FAILUREATLAS is applied only when a parent node already fails. If a parent succeeds, all descendants are explored; thus failures that occur only at more specific prompts are not pruned. Once a parent fails, it already constitutes a valid minimal failure-inducing concept under our definition. Empirically, on 1,000 sampled minimal failure slices from SDXL Turbo, over 98% of their child nodes also fail, indicating that no-prune exploration mainly revisits already failed regions rather than uncovering new minimal failures. Under a fixed evaluation budget, pruning removes redundant evaluations without changing the taxonomy of minimal failure slices.

A.2. Experiments Details

A.2.1. Implementation Details

We apply FAILUREATLAS to a range of T2I models, including SD1.5, SDXL Turbo, SD3.5 Large Turbo, and Flux.1-dex. For all models, we adopt the hyper-parameters used in their official implementations (e.g., inference steps, guidance scale, and precision settings), shown in Table 1.

Table 1. Hyper-parameters used for different generative models. We adopt the default setting from official implementation.

Model	Inference Steps	Guidance Scale	Precision
SD1.5	50	7.5	FP16
SDXL Turbo	1	0.0	FP16
SD3.5 Large Turbo	4	0.0	BF16
Flux.1-dex	50	3.5	BF16

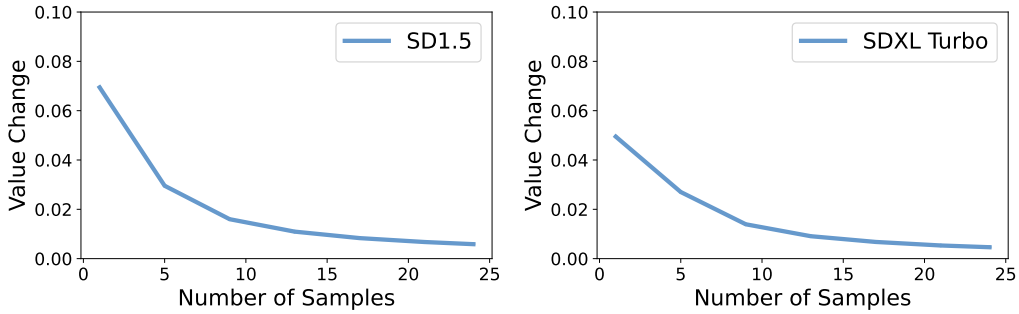


Figure 1. We analyze how the average success rate changes with the number of repeated generations. The success rate stabilizes with 25 generations, with additional samples leading to an average change of less than 0.0058 for SD1.5 and 0.0046 for SDXL Turbo.

A.2.2. Visualization and Case Study

We present six additional error slices for each model. For SD1.5, we observe that the model struggles with producing certain rare but plausible combinations, such as dusty chopstick or sitting zebra. It also often fails to render the racket or shuttlecock in badminton, frequently producing distorted images instead.

For SDXL Turbo, we find that it often ignores or simplifies control signals in favor of more common scenes. For instance, when prompted to generate a paper helicopter, it instead produces a paper airplane. Similarly, when asked for an oval-shaped cake, it outputs a circular cake.

For SD3.5 Large Turbo, we continue to observe failures in rendering certain entities, such as rubber bands. It also exhibits error patterns similar to earlier models. For example, prompting for a pink tomato often yields a red tomato, although other unusual colors such as purple can succeed. The model sometimes struggles with complex attribute controls, including lighting and artistic style, and frequently fails when asked to render a specific number of tools, such as scissors.

For Flux.1-Dev, we find failures in challenging scenarios such as bent keys or keys scattered horizontally. The model also struggles with rarer attribute specifications—for example, requesting a square camera rarely induces a shape change. Likewise, assigning the color gray to a camera often fails, even though other colors are consistently successful.

A.2.3. Bias Analysis

To analyze the robustness of our evaluation, we examine two potential sources of bias. The first concerns the effect of repeated generations on the stability of node-level success rates. We sample 100K nodes for both SD1.5 and SDXL Turbo, and record the average change in success rate as the number of generations increases. As shown in Figure 1, the curve flattens after around 15 generations, and further sampling beyond 25 generations leads to an average change of less than 0.58% for SD1.5 and 0.46% for SDXL Turbo. This indicates that our choice of generating 25 images per node provides a sufficiently stable estimate of performance.

The second potential bias arises from the evaluation procedure itself. To quantify this, we conduct a human alignment study: for each of the 758 entities and 437 attributes, we generate 5 images and compare model predictions with human judgments. Results are summarized in Figure 2, grouped at the category level. We observe high consistency, with an average alignment rate of 96% for entities and 86% for attributes. The lowest-alignment attributes are Temperature, Hardness, and Visual Density, which are known to be visually ambiguous. These results confirm that both repeated sampling and evaluation are stable, lending confidence to the reliability of our reported findings.

A.3. Future Study and Application

We outline several promising directions for future work.

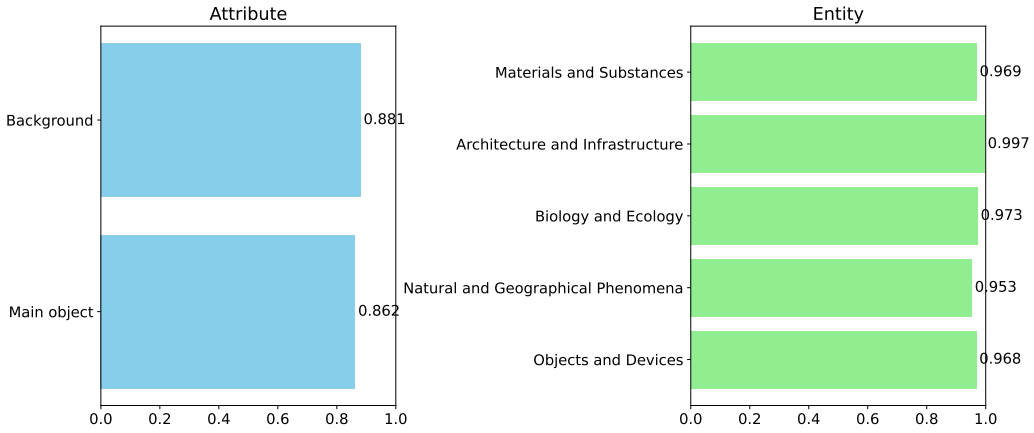


Figure 2. Human alignment test for the automatic evaluation method. Left: Results on attributes, grouped by categories. Right: Results on entities, grouped by categories.

A.3.1. In-depth Exploration of Failure Patterns

In this paper, our primary goal is to introduce the proposed framework. Accordingly, although we present several representative failure cases of T2I models, our experiments mainly focus on demonstrating the effectiveness of the framework itself. In future work, a more systematic and in-depth investigation of model failure patterns can be conducted. Since our framework can generate a large number of failure cases, it enables detailed analyses of failure characteristics, including both the differences and the commonalities across models.

A.3.2. Extension to Other Domains

Our framework offers a systematic and efficient approach for exploring model failures and can be applied to various domains as long as (1) the model inputs can be easily constructed (e.g., text), and (2) the model outputs can be automatically evaluated. For target domains that satisfies these conditions, one can construct a corresponding corpus as input space, and apply our framework for automated error discovery. Notably, the corpus does not need to be limited to entities and attributes; it can also encode different input types, such as "editing background" or "editing main object" in image-editing tasks. We hope that this framework can offer a new perspective for model development across diverse fields.

A.3.3. Data Lens

Our method provides model developers with a clear and structured view of a model's weaknesses. As described in the paper, training data can be analyzed using the corpus as the bridge. By combining insights from both data analysis and model weaknesses, developers can better understand what types of data are currently lacking and which cases remain challenging for the model. This understanding can guide the construction of more effective and targeted training datasets.

A.4. Social Impact

Our study focuses on systematic error discovery in text-to-image models through active exploration. It does not involve human subjects or private data. All models evaluated are publicly released systems, and all data used—either for training analysis or evaluation—are derived from publicly available datasets. While our work highlights potential weaknesses in generative models, it is intended solely for diagnostic and research purposes, with no intent to exploit or amplify model failures. We see this direction as a step toward more robust and interpretable generative AI systems.

SD1.5

Prompt: An image of lettuce. The background is gold.
Success Rate: 52%

Prompt: An image of dusty chopsticks.
Success Rate: 54%

Prompt: An image of an onion. The background is a tornado.
Success Rate: 54%

Prompt: An image of badminton.
Success Rate: 56%

Prompt: An image of a wolf made of stone, jumping.
Success Rate: 66%

Prompt: An image of a zebra sitting.
Success Rate: 50%

Figure 3. Example error slices of SD1.5. The prompt specifies entities in blue and attributes in red.

SDXL Turbo

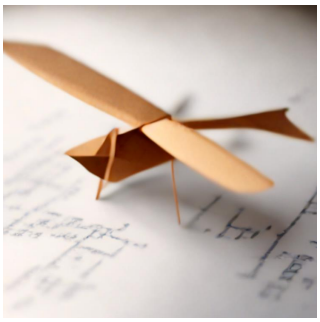
Prompt: An image of five pencils.
Success Rate: 50%

Prompt: An image of an oval cake.
Success Rate: 50%

Prompt: An image of three square shields.
Success Rate: 37%

Prompt: An image of a person walking. The background is pink.
Success Rate: 26%

Prompt: A bottom view of daisy.
Success Rate: 50%



Prompt: An image of a paper helicopter.
Success Rate: 32%

Figure 4. Example error slices of SDXL Turbo. The prompt specifies entities in blue and attributes in red.

SD3.5 Large Turbo

Prompt: An image of a rubber band.
Success Rate: 24%

Prompt: An image of a five scissors.
Success Rate: 50%

Prompt: An image of a sheep. The overall image has an abstract art style.
Success Rate: 50%

Prompt: An image of a pink tomato.
Success Rate: 50%

Prompt: An image of an orange candle holder. The time is sunset.
Success Rate: 59%

Prompt: An image of a watch in backlight.
Success Rate: 54%

Figure 5. Example error slices of SD3.5 Large Turbo. The prompt specifies entities in blue and attributes in red.

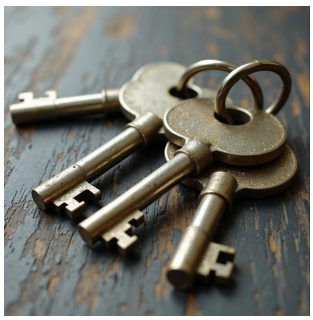
Flux.1 dev

Prompt: An image of a square camera.
Success Rate: 52%

Prompt: An image of a white camera. The background is beige.
Success Rate: 66%

Prompt: An image of a gray camera.
Success Rate: 56%

Prompt: An image of keys scattered horizontally.
Success Rate: 68%



Prompt: An image of bent keys.
Success Rate: 46%

Prompt: An image of brown keys.
Success Rate: 64%

Figure 6. Example error slices of Flux.1-dev. The prompt specifies entities in blue and attributes in red.