Follow the Flow: On Information Flow Across Textual Tokens in Text-to-Image Models

Anonymous ACL submission

Abstract

Text-to-image generation models suffer from alignment problems, where generated images fail to accurately capture the objects and relations in the text prompt. Prior work has focused on improving alignment by refining the diffusion process, ignoring the role of the text encoder, which guides the diffusion. In this work, we investigate how semantic information is distributed across token representations in text-to-image prompts, analyzing it at two levels: (1) *in-item representation*—whether individual tokens represent their lexical item (i.e., a word or expression conveying a single concept), and (2) *cross-item interaction*—whether information flows between tokens of different lexical items. We use patching techniques to uncover encoding patterns, and find that information is usually concentrated in only one or two of the item's tokens; for example, in the item "San Francisco's Golden Gate Bridge", the token "Gate" sufficiently captures the entire expression while the other tokens could effectively be discarded. Lexical items also tend to remain isolated; for instance, in the prompt "a green dog", the token "dog" encodes no visual information about "green". However, in some cases, items do influence each other's representation, often leading to misinterpretations—e.g., in the prompt "a pool by a table", the token "pool" represents a "pool table" after contextualization. Our findings highlight the critical role of token-level encoding in image generation, and demonstrate that simple interventions at the encoding stage can substantially improve alignment and generation quality.

1 Introduction

Text-to-image (T2I) models typically consist of two main components: a text encoder and a diffusion model (Ho et al., 2020; Song and Ermon, 2019). The former processes the user's prompt, transforming it into a representation that guides the latter in generating the image. Though widely used, T2I models often exhibit prompt-image misalignment, where generated images fail to capture key concepts from the user's prompt (Chefer et al., 2023a; Rassin et al., 2022; Huang et al., 2023a). Prior work has attempted to address these issues by modifying the diffusion stage, and particularly the cross-attention mechanism (Rassin et al., 2023; Chefer et al., 2023a; Dahary et al., 2024), under the implicit assumption that each textual token reliably encodes the item it is intended to convey. This raises two fundamental questions regarding *textual encoding*: (1) is the meaning of a lexical item evenly distributed across its tokens, or concentrated in just one or two? and (2) does each token exclusively encode its lexical item, or can it also absorb information from surrounding items?

In this work, we examine those questions and study how visual information is distributed across tokens after the textual encoding stage. We focus our analysis on *lexical items*—words or phrases that convey a single concept, such as "pelican" or "golden gate bridge". We trace how item information is distributed both *within* the tokens of a single item (*in-item*), and *across* tokens of different items (*cross-item*).[1] To do so, we use a causal intervention framework (Toker et al., 2025) that systematically assess what information is encoded in each contextual token representation at the encoder's output (Section 2).[2] We then evaluate this framework on prompts drawn from widely used T2I benchmarks (Section 3).

For *in-item* representation (Section 4), we find that a lexical item's meaning is typically concentrated in one or two *representative tokens*. Surprisingly, ablating the non-representative tokens reduces generation errors by 21%. We further show that representative tokens can be identified

[1] See Fig. 1 for examples of the different cases.

[2] Throughout this work, we study token representations at the output of the text encoder. For brevity, we sometimes omit the word "representation".

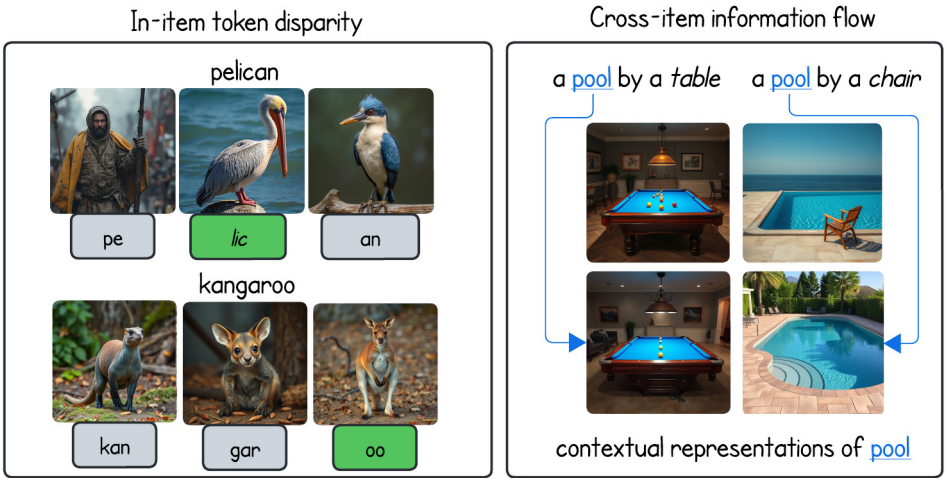

Figure 1: **Our main findings**. **Left:** Information within a lexical item is unevenly distributed across its tokens' contextualized representations. In this example, one token carries the meaning of the entire item (e.g., **lic** represents a pelican, while **pe** and **an** do not). **Right:** Contextual items may distort a token's encoding, leading to misaligned interpretations in the generated image. Top: generations from the full prompts "a **pool** by a *table*" (misaligned) and "a **pool** by a *chair*" (aligned). Bottom: generations from the contextualized token **pool** alone. With *table*, **pool** encodes a pool table; with *chair*, it retains the intended swimming pool meaning.

efficiently without image generation, opening the door to pruning non-representative tokens directly within T2I pipelines.

For *cross-item* interactions (Section 5), we assess whether information flows between different lexical items in a prompt. We observe such cross-item flow in 11% of cases. Interestingly, the flow of information does not always follow sentence syntax, which can result in incorrect item resolution of items within specific contexts—especially with polysemous words. For instance, in the prompt "a pool by a table", "pool" can wrongly suggest "table" refers to a billiard table. To address such item-level misalignment, we introduce a simple patching technique that both corrects these errors and allows explicit control over the interpretation of ambiguous items.

Overall, our analysis shows that token-level encoding is typically concentrated , largely item-isolated , yet vulnerable to *semantic leakage* in cases such as polysemous words, which can lead to misinterpretations. Across our experiments, we also demonstrate simple interventions for improving alignment, highlighting the importance of further investigating the text encoder's role in T2I.

2 Methodology

We introduce a method for intervening on the text encoder by generating images from arbitrary subsets of token representations while masking the others.[3] We later use this method to analyze two phenomena: (1) how information is distributed across tokens within a single lexical item (Section 4), and (2) how different lexical items influence one another (Section 5). Our approach builds on the text intervention framework proposed by Toker et al. (2025), adapting it to enable controlled generation from chosen subsets of tokens.

Given a prompt with N tokens $t_1, t_2, \ldots, t_N$, our goal is to isolate and interpret the information encoded by a subset of these tokens. Let $S \subset \{1, \ldots, N\}$ be the indexes of a given subset of the tokens, where $0 < |S| < N$. We begin by encoding the full prompt using the text encoder E, yielding the final hidden states $h_1, \ldots, h_N$. Separately, we encode a sequence consisting entirely of pad tokens to obtain pad embeddings $p_1, \ldots, p_N$. We then construct a *patched prompt* by replacing all hidden states outside S with the corresponding pad embeddings:

$$\tilde{t}_i = \begin{cases} h_i & i \in S, \\ p_i & \text{otherwise} \end{cases} \quad \text{for } i = 1, \ldots, N.$$

The patched sequence $\tilde{t}_1, \ldots, \tilde{t}_N$ is then used to guide the diffusion model. Generating an image from this patched representation allows us to isolate

[3]While tokens interact throughout encoding, their final representations can be examined in isolation. This allows us to visualize how each token—following the encoding step—contributes to the diffusion process.

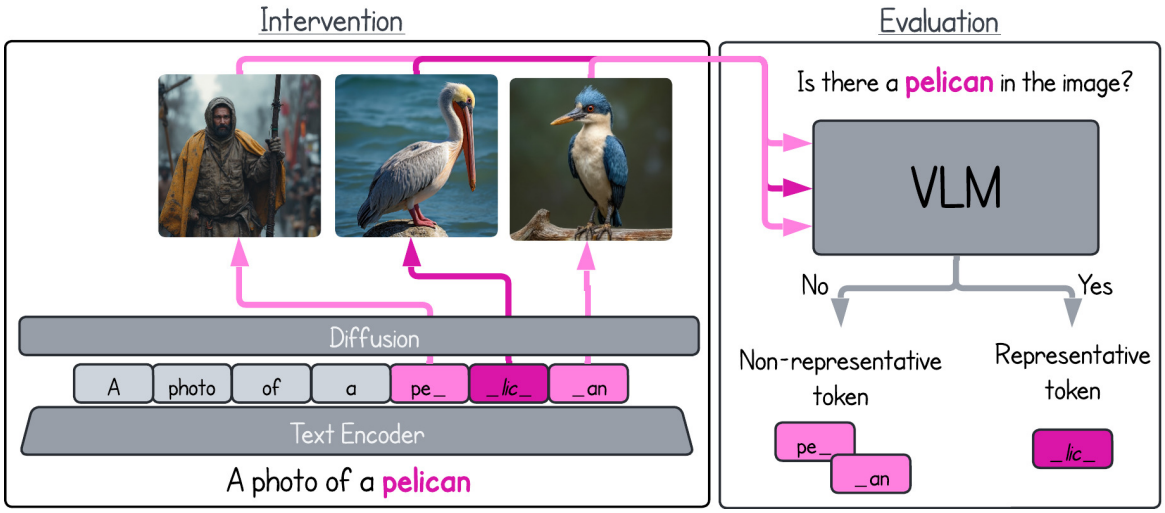

Figure 2: **Evaluating in-item information flow.** Our proposed framework interprets the information flow within a lexical item. We generate images from each token comprising the lexical item (left) and analyze them with a VLM (right). In this example, only the token **lic** represents the concept "pelican", whereas **pe** and **an** do not.

the individual contributions of the selected tokens as interpreted by the diffusion model. To evaluate the information in the generated images, we use a vision-language model (VLM). See Fig. 2 for an illustration of our method. Below we describe our experimental setup, followed by the two phenomena we analyze using our method.

3 Experimental Setup

Models. We experiment with four text-to-image models: FLUX-schnell, FLUX-dev (black-forest labs, 2024), SDXL-Turbo (Sauer et al., 2023), and SANA (Xie et al., 2024a). The FLUX models employ T5-XXL (Raffel et al., 2019) as their main text encoder, SDXL-Turbo uses CLIP (Radford et al., 2021), and SANA uses Gemma (Team et al., 2024). While our core insights and results are consistent across all models, certain differences arise due to encoder-specific artifacts. In particular, the unidirectional information flow in Gemma and CLIP, as well as the presence of a dominant [CLS] token in CLIP, lead to a slightly different behaviors compared to the bidirectional T5 encoder. For clarity and simplicity, we focus on results from FLUX-schnell in the main paper, and discuss these cross-model differences in detail in Section 6.

Data. We use a subset of 1,053 prompts from the DrawBench (Saharia et al., 2022) and PartiPrompts (Yu et al., 2022) datasets, filtered to include 4–20 words prompts and exclude cases with added complexity unrelated to our focus (e.g., misspellings, written text, rare words). For each prompt, we generate five images using different random seeds. To extract the lexical items, we prompt GPT-4o (OpenAI et al., 2024), resulting in 4,864 unique items across prompts. See Section A.1.1 for further technical details. We then use spaCy (Honnibal et al., 2020) to determine the part of speech of each lexical item, and retain only nouns, proper nouns, and adjectives, as these are typically concrete and can be identified in their visual representation. We end up with 3,891 unique lexical items and use them to evaluate both *in-item* representation and *cross-item* interactions.

Evaluation. To evaluate the content of generated images, we employ Qwen2-VL-72B-Instruct (Wang et al., 2024), a model with strong general vision capabilities. We restrict our evaluation to binary (yes/no) questions and use the model to assess prompt-image alignment and the presence of specific items in the image (see Section A.1.2 for additional details). To validate Qwen2-VL's reliability, we ask three human annotation to evaluate 100 randomly sampled cases, evenly split between *in-item* representation (Section 4) and *cross-item* interactions (Section 5) settings. The results indicate substantial agreement, with a Cohen's Kappa values of 0.868 for *in-item* representation and 0.764 for *cross-item* interactions. Next, we compute the majority agreement among the human annotators and compare it to the model predictions. For *cross-item* interactions, the accuracy (F1) is 0.927 (0.933), and for *cross-item* interactions, the accuracy is 0.810 (0.740). These

results suggest that Qwen2-VL predictions align well with human judgment for our tasks.

4 In-Item Representation

In this section, we analyze information flow within lexical items at the token level. We first show semantic information is unevenly distributed across tokens, with typically one or two tokens representing the item (Section 4.1). We then show non-representative tokens are largely redundant and may even harm generation (Section 4.2).

4.1 Information Distribution Across Tokens

We begin by exploring how information is distributed across the tokens within a lexical item. Is it spread evenly across tokens, or rather concentrated in specific? This question is relevant, as many T2I applications (Chefer et al., 2023a; Rassin et al., 2023; Dahary et al., 2024) treat all tokens of a lexical item in a given prompt equally. Uncovering an asymmetrical distribution of information could improve the effectiveness of such applications by focusing on the most informative tokens.

Given a prompt and a lexical item, we feed the prompt to the text encoder, obtaining contextualized token representations. We then identify the tokens comprising the item and apply our intervention method (Section 2) to generate an image conditioned on each token's representation, studying how much of the item's meaning is retained in that token's contextual representation, as probed from the final encoder layer. Finally, we use Qwen2-VL to assess whether the image represents the overall lexical item it is part of. We define a *representative token* as one whose isolated representation results in an image containing its corresponding lexical item. Tokens that do not meet this criterion are considered *non-representative*. We repeat this analysis for each lexical item and for each prompt in our dataset. For example, given the prompt "a *spaceship* that looks like the *Sydney Opera House*," we conduct the experiment for each of "spaceship's" tokens ('space', 'ship') and "Sydney Opera House" tokens ('Sydney', 'Opera' and 'House'). Our results (Fig. 3, *bottom row*) show that in 89% of the cases, there is (at least) one *representative token*. Interestingly, in cases where no representative tokens exist, the item is usually absent from the full-prompt image as well, partly reflecting gaps in the encoding process (see Section A.5 for detailed analysis and examples).

We next focus on instances where at least one

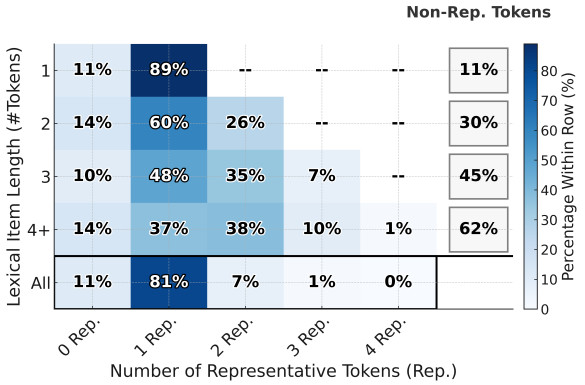

Figure 3: Distribution of representative tokens per item length. "Rep." denotes the number of representative tokens; the bottom row aggregates over all lexical items. In most cases, one or two tokens sufficient to represent the entire item. As item length increases, the number of non-representative tokens grows accordingly.

token represents the lexical item, and examine the number of *representative* and *non-representative tokens* across items of different lengths. Our results (Fig. 3, *top rows*) show that typically one or two tokens represent the concept, while the remaining tokens are *non-representative*. Further, as the token length of the lexical item becomes longer, the number of *non-representative* tokens increases (Fig. 3, *rightmost column*). Inspecting all lexical items in our data composed of two or more tokens, these *non-representative tokens* account for 52% of their tokens. Therefore, we next examine the effect of removing *non-representative tokens* altogether.

4.2 Textual Tokens During Diffusion: Are They All Necessary?

We now examine whether *non-representative* tokenshave a prominent effect on image generation. To answer this question, we apply our intervention method to each prompt, this time generating an image after masking all the *non-representative tokens*, while keeping *all representative tokens*. We use Qwen-VL to measure whether this image aligns with the prompt, and compare it to an image generated from the full prompt without intervention.

***Non-representative tokens* are redundant.** Our results (Table 1) show that removing *non-representative tokens* generally preserves quality (see Fig. 4, left-hand side for illustration). When the original generation is aligned with the prompt, the generated image after non-representative token removal remains aligned in 98% of cases, suggesting these tokens are largely redundant. Sur-

Tokens Removed	# Prompts	Acc.		Relative %			Accuracy Δ (pp)
		Before	After	Unaffected	Degraded	Improved	
1	144	81.25	83.33	98.29	0.85	14.83	+2.08
2	98	82.65	88.78	100.00	0.00	35.27	+6.13
3	45	93.33	93.33	97.62	2.38	33.28	+0.00
4	24	87.50	91.67	100.00	0.00	33.36	+4.17
5+	28	78.57	85.71	100.00	0.00	33.32	+7.14
Overall	339	83.48	87.02	98.90	0.71	25.00	+3.54

Table 1: Effect of removing non-representative tokens on prompt-level accuracy. The table reports: (i) accuracy before and after token removal, (ii) the percentage of originally successful prompts that remained successful (Unaffected) or became failures (Degraded), and (iii) the percentage of originally failed prompts that were corrected (Improved). Overall, removing non-representative tokens rarely harms generation and often improves it.

Figure 4: Examples illustrating the effect of removing *non-representative tokens*. **Top row**: Images generated after removing *non-representative tokens* (Representative tokens are shown in **bold**; non-representative tokens are in gray.). **Bottom row**: Images generated from the full prompt. **Left**: In most cases, removal results in no noticeable effect on the generation. **Right**: In some cases, removal improves alignment with the prompt.

prisingly, in cases where the original image fails to align with the prompt, we observe a 21% improvement in alignment after removing non-representative tokens (see Fig. 4, right-hand side). We attribute this improvement to the model relying exclusively on the remaining representative tokens, which encode the correct semantics of the item.[4]

4.3 Efficient Redundant Token Identification

While identifying and removing redundant tokens before the diffusion process improves image-text alignment, this identification process is computationally expensive, as it requires generating an image from each subtoken of a lexical item.

To make this process efficient, we introduce a lightweight classifier that predicts token redundancy directly from text encoder representations, *without generating any images*. This classifier operates immediately after textual encoding, allowing redundant tokens to be masked before the prompt is passed to the diffusion model. It enables fast, on-the-fly filtering and straightforward integration into existing T2I pipelines. Using the same dataset of token-level annotations described (after the classification using the original patching technique), we train a simple k-nearest neighbors (k-NN) classifier ($k = 5$, Euclidean distance) to predict redundancy from token embeddings, with an 80/20 train-validation split. As shown in Table 2, the classifier achieves **92% precision** and 82% accuracy, ensuring that representative tokens are rarely misclassified and inadvertently removed.

By filtering out redundant tokens before generation, the diffusion model receives cleaner, more

[4]We measure improvement rates with Qwen2-VL before and after masking non-representative tokens. See Section A.1.2 for further details.

Metric	Accuracy	Precision	Recall	F1-score
Score	0.82	0.92	0.74	0.82

Table 2: Performance of lightweight k-NN classifier ($k = 5$) for predicting token redundancy.

Category	Count	Percentage
# pairs	15,950	100.00%
No Information Flow	14,251	89.35%
Information Flow	1,699	10.65%
– Source before Reference	835	49.15%
– Reference before Source	823	48.44%

Table 3: Distribution of information flow between lexical item pairs. Subcategories under "With Information Flow" indicate the order of source and reference.

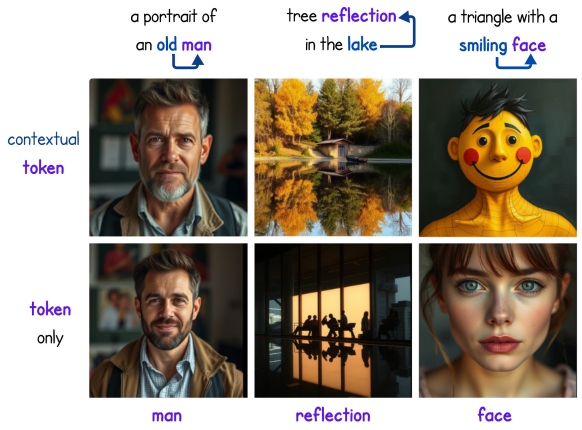

Figure 5: **Examples of information flow between items.** Top: Images generated from a **lexical item** encoded alongside **another item** that alters its representation. Bottom: Images generated from the uncontextualized representation of the same lexical item.

focused inputs, improving prompt-image alignment and reducing noise. Using only representative tokens yields a *21% reduction in generation failures* (Table 1, Figure 4). This transition moves our approach beyond analysis, establishing it as a practical component that can be applied to efficiently enhance the overall alignment of T2I pipelines.

5 Cross-Item Interactions

In Section 4, we have demonstrated that a single token can effectively encapsulate the semantics of an entire lexical item. This raises a broader question: What defines the segmentation boundaries of a lexical item? Can its tokens also encode information about adjacent items, or is their meaning primarily confined within the item itself?

To this end, we isolate each lexical item in a prompt and assess whether it encodes information about *other* items in the prompt. For each item, we generate images from its contextualized representation (encoded within the full prompt), and its uncontextualized representation (using the item's text by itself as a prompt). For example, for the prompt "tree *reflection* in the lake," we generate two images: one using the single word *reflection*, without any context, and another where the same item is contextualized as part of the full prompt. We then use Qwen2-VL to evaluate whether any other item from the prompt—such as *tree* or *lake*—appears in the contextualized image, but not in the uncontextualized one, which would indicate flow introduced by the surrounding context. See Section A.4 for implementation details.

Our results (Table 3) show that in general, lexical items do not encode information from other items in the prompt: 89% of the tokens do not lead to images containing other items. Interestingly, in the remaining 11%, information flow can also emerge between items with no direct syntactic relation in the prompt—e.g., information flowing from *lake* into the representation of *reflection*.[5]

5.1 Can Information Flow Leads to Misinterpretation of Items?

Our analysis reveals a recurring pattern of semantic influence between items, where the interpretation of one item can be affected by another that is syntactically distant. While such influence is crucial, as language is inherently contextual, some times this influence becomes misleading or distorting. For example, in "a standing *zebra* to the right of a city *bus station*," where the association between *bus station* and *zebra* causes the model to depict a *zebra crossing* instead of the animal (see Fig. 6). We refer to this *clearly unambiguous*, unintended and misleading form of cross-item influence as *semantic leakage* (Rassin et al., 2022; Gonen et al., 2024). We note that this definition is behaviorally defined—a leakage only occurs if the model is wrong in a clearly unambiguous case.

We note that such distorting forms of information flow typically arise in cases involving polysemous words. For example, the item *bats* is interpreted as *baseball bats* in the prompt "bats fly around a baseball stadium", even though the intended meaning is obviously the flying animal. We focus on cases in which the context provides a *definitive cue* for resolving ambiguity of the pol-

[5] See Fig. 5 for qualitative examples

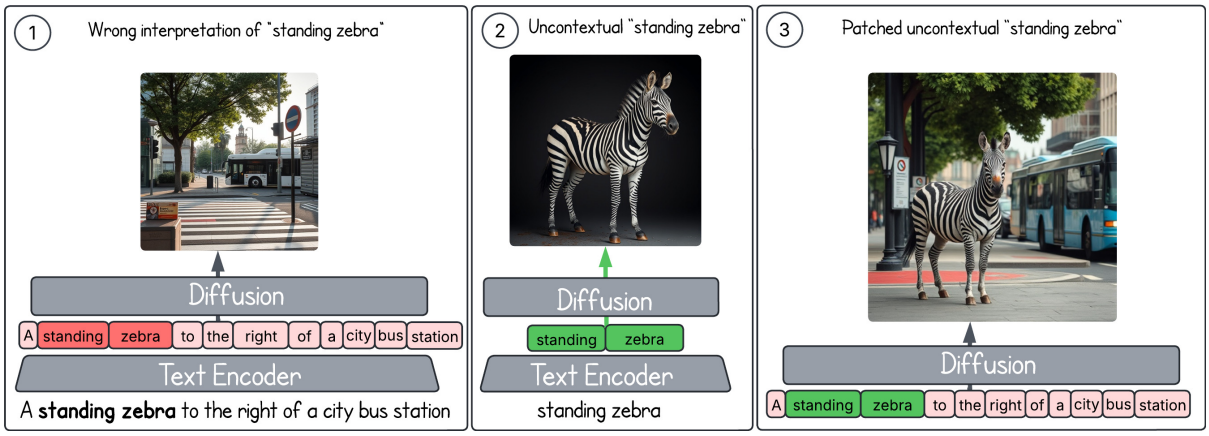

Figure 6: **Verifying and mitigating textual semantic leakage by replacing the contextually leaked concept representation.** (1) Regular generation produces an image showing a crosswalk to the right of a bus station. (2) Generation from the prompt "standing zebra", without any context, results in the expected interpretation of the zebra as an animal. (3) Mitigating the leakage by generation using the original prompt, but with the leaked concept "standing zebra" replaced by it's uncontextualized representation.

ysemous word, but the model still fails to resolve the item correctly. We hypothesize that these cases are caused by *semantic leakage* during the textual encoding, where unintended associations override contextually appropriate meaning.

To test this hypothesis, we use the dataset from Rassin et al. (2022), which includes prompts known to induce misalignment due to implicit lexical associations (e.g., *bat* and *baseball stadium*). Unlike standard T2I benchmarks such as Huang et al. (2023b), which focus on spatial or visual challenges, these prompts highlight failures rooted in the encoding process. Since the original set includes only 30 relevant examples, we expand it with GPT-4o to generate additional candidates. We then keep only prompts whose intended interpretation of each polysemous word was judged by human annotators to be *fully unambiguous*, and compare that intended meaning to the one expressed in the generated image. Prompts where the model's output contradicts the intended sense yield a final curated set of 110 examples (see Section A.2 and A.1.5 more details and the full list).

On those prompts, we compare two generations: (1) from the full prompt, (2) from a "cleaned" version containing only the item and its modifiers. For example, given the prompt "*bats* fly around a baseball stadium", we generate images of *bats* once encoded in the phrase "*bats* fly" and once encoded in the full context. Each image is then evaluated for whether it depicts the intended interpretation of the item—*an animal bat* (See Fig. 6 for details). In 93% of cases, the item is incorrectly resolved in the contextualized item level, indicating that the error originates at the encoding stage, whereas the cleaned version preserves the intended meaning.

5.2 Mitigation of Semantic Leakage

Having established that misalignment may arise from *semantic leakage* during encoding, we now turn to a method for mitigating it. We apply a patching technique (Fig. 6, step 3), in which the wrongly resolved item are replaced with its "cleaned" representation. Our method resembles Feng et al. (2023), but differs in that we replace the target tokens completely, and not just the value projection. We use RAG-Diffusion (Tan et al., 2024) as a baseline. This method enhances generation by ensuring items are correctly placed, which it achieves by creating bounding boxes for each item and limiting the diffusion attention to these specific areas.

Table 4 reports the leakage rates under each method. Our patching technique consistently achieves the strongest reduction, lowering misinterpretations to 20% on Flux-Dev and 14% on Flux-Schnell, substantially outperforming the baseline. These results demonstrate that patching provides an effective and practical solution for mitigating *semantic leakage* of this kind.

While our analysis focused on prompts with clear intended meanings, similar effects arise in ambiguous (e.g., "*bats* in a stadium") or biased contexts (e.g., gender-related prompts). We further discuss these cases in appendices A.6 and A.7, showing how patching naturally extends to user-guided control and bias mitigation.

Model	Initial Leakage	↓ RAG-Diffusion	↓ Patching (Ours)
Flux-Dev	79%	39%	**20%**
Flux-Schnell	94%	43%	**14%**

Table 4: Mitigation of semantic leakage across models. Values show percentage of prompts exhibiting leakage (lower is better).

6 Discussion

Developing textually challenging benchmarks. Current T2I datasets focus primarily on visual or spatial complexity (Huang et al., 2023a; Ghosh et al., 2023; Saharia et al., 2022; Yu et al., 2022). Yet our findings show that even slight linguistic ambiguity—particularly with polysemous or compositional phrases—can cause encoding failures. This highlights the need for evaluation benchmarks that probe textual difficulty more directly, which may in turn drive improvements in encoder design.

Generalization across T2I architectures. Our analysis across three encoder types—T5 (FLUX models), Gemma (SANA), and CLIP (SDXL-Turbo)—reveals two key differences that shape semantic representation and interpretability. First, T5 is bidirectional, allowing representative tokens to appear at various positions within a lexical item, while Gemma and CLIP are unidirectional, consistently placing representative tokens at the final token. This regularity in unidirectional models simplifies practical tasks such as redundant token masking. Second, T5 and Gemma distribute meaning across multiple tokens, with over 89% of lexical items containing at least one representative token, whereas CLIP centralizes most semantic information in its [CLS] token, leaving other tokens weak and limiting token-level interpretability. These distinctions, discussed further in Appendices A.3.2 and A.3.3, highlight how encoder architecture impacts both analysis and downstream applications.

7 Related Work

Interpretability in T2I models. Recent work has explored how T2I models encode and align concepts. Chefer et al. (2023b) studied CLIP's latent space, while Toker et al. (2024) interpreted representations across the textual encoding process using the diffusion model as a lens. Others examined text-to-image alignment via attention maps (Tang et al., 2023). Another direction is to apply SAEs (Cunningham et al., 2023) to interpret intermediate representations in T2I models (Cywiński and Deja, 2025). We on the other hand, focus on token-level information at the encoder's final layer, which directly conditions generation process.

Token representation and flow in LLMs. Studies have shown that token information in LLMs is not uniformly distributed. Kaplan et al. (2025) found that subwords fuse into word-level meaning, while Feucht et al. (2024) documented how later tokens can erase earlier ones. Attention-based flow (Vig and Belinkov, 2019; Clark et al., 2019) can be misleading (Pruthi et al., 2020; Jain and Wallace, 2019), prompting alternatives like Attention Rollout (Abnar and Zuidema, 2020). Methods like Patchscopes (Ghandeharioun et al., 2024) and logit lens (nostalgebraist, 2020) decode internal representations, but do not test if downstream components use them.

Probing and causal methods. Probing methods (Adi et al., 2016; Liu et al., 2019; Zhang and Bowman, 2018; Brunner et al., 2019) reveal information presence but are not fully reliable, as probes can exploit spurious correlations (Belinkov, 2022). Instead, we use causal interventions to test whether token information is used during image generation.

Challenges in T2I models. *Semantic leakage*—when context distorts word meaning—has been observed in both LLMs (Gonen et al., 2024) and T2I (Rassin et al., 2022; Dahary et al., 2024). Another common issue is *neglect*, where key items are omitted (Chefer et al., 2023a; Chang et al., 2024). While prior work focuses on solutions during the diffusion process, we show that some of these failures often originate in the text encoder.

8 Conclusion

We present a causal masking method for probing the semantic content encoded in individual tokens at the output of the text encoder in T2I models. Using this framework, we show that (1) lexical meaning is often concentrated in one or two dominant tokens, and (2) cross-item information flow—observed in about 11% of cases—can distort item interpretation. These insights lead to two practical improvements: masking redundant tokens and patching misaligned ones, both enhancing text–image alignment. Our findings highlight the text encoder as a key source of alignment issues and a promising target for intervention and improvement.

Limitations

Evaluating token-level representations remains challenging. While we rely on strong vision-language models as judges and validate key findings through human evaluation, these are still approximations of true semantic alignment. Our prompt set focuses on object-centric, syntactically simple cases, which may limit generalization to prompts involving misspellings, rare words, or abstract concepts. Further work is needed to explore how information flow behaves under more linguistically complex conditions.

References

Samira Abnar and Willem Zuidema. 2020. Quantifying attention flow in transformers. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 4190–4197, Online. Association for Computational Linguistics.

Yossi Adi, Einat Kermany, Yonatan Belinkov, Ofer Lavi, and Yoav Goldberg. 2016. Fine-grained analysis of sentence embeddings using auxiliary prediction tasks. *arXiv preprint arXiv:1608.04207*.

Yonatan Belinkov. 2022. Probing classifiers: Promises, shortcomings, and advances. *Computational Linguistics*, 48(1):207–219.

black-forest labs. 2024. Flux. `https://github.com/black-forest-labs/flux`.

Gino Brunner, Yang Liu, Damian Pascual, Oliver Richter, Massimiliano Ciaramita, and Roger Wattenhofer. 2019. On identifiability in transformers. *arXiv preprint arXiv:1908.04211*.

Zhiyuan Chang, Mingyang Li, Junjie Wang, Yi Liu, Qing Wang, and Yang Liu. 2024. Repairing catastrophic-neglect in text-to-image diffusion models via attention-guided feature enhancement. In *Findings of the Association for Computational Linguistics: EMNLP 2024*, pages 11379–11390, Miami, Florida, USA. Association for Computational Linguistics.

Hila Chefer, Yuval Alaluf, Yael Vinker, Lior Wolf, and Daniel Cohen-Or. 2023a. Attend-and-excite: Attention-based semantic guidance for text-to-image diffusion models. *ACM transactions on Graphics (TOG)*, 42(4):1–10.

Hila Chefer, Oran Lang, Mor Geva, Volodymyr Polosukhin, Assaf Shocher, Michal Irani, Inbar Mosseri, and Lior Wolf. 2023b. The hidden language of diffusion models. *arXiv preprint arXiv:2306.00966*.

Kevin Clark, Urvashi Khandelwal, Omer Levy, and Christopher D. Manning. 2019. What does BERT look at? an analysis of BERT's attention. In *Proceedings of the 2019 ACL Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP*, pages 276–286, Florence, Italy. Association for Computational Linguistics.

Hoagy Cunningham, Aidan Ewart, Logan Riggs, Robert Huben, and Lee Sharkey. 2023. Sparse autoencoders find highly interpretable features in language models. *arXiv preprint arXiv:2309.08600*.

Bartosz Cywiński and Kamil Deja. 2025. Saeuron: Interpretable concept unlearning in diffusion models with sparse autoencoders. *arXiv preprint arXiv:2501.18052*.

Omer Dahary, Or Patashnik, Kfir Aberman, and Daniel Cohen-Or. 2024. Be yourself: Bounded attention for multi-subject text-to-image generation. In *European Conference on Computer Vision*, pages 432–448. Springer.

Weixi Feng, Xuehai He, Tsu-Jui Fu, Varun Jampani, Arjun Akula, Pradyumna Narayana, Sugato Basu, Xin Eric Wang, and William Yang Wang. 2023. Training-free structured diffusion guidance for compositional text-to-image synthesis. *Preprint*, arXiv:2212.05032.

Sheridan Feucht, David Atkinson, Byron Wallace, and David Bau. 2024. Token erasure as a footprint of implicit vocabulary items in llms. *Preprint*, arXiv:2406.20086.

Asma Ghandeharioun, Avi Caciularu, Adam Pearce, Lucas Dixon, and Mor Geva. 2024. Patchscopes: A unifying framework for inspecting hidden representations of language models. *ArXiv*, abs/2401.06102.

Dhruba Ghosh, Hanna Hajishirzi, and Ludwig Schmidt. 2023. Geneval: An object-focused framework for evaluating text-to-image alignment. *Preprint*, arXiv:2310.11513.

Hila Gonen, Terra Blevins, Alisa Liu, Luke Zettlemoyer, and Noah A Smith. 2024. Does liking yellow imply driving a school bus? semantic leakage in language models. *arXiv preprint arXiv:2408.06518*.

Jonathan Ho, Ajay Jain, and Pieter Abbeel. 2020. Denoising diffusion probabilistic models. *Advances in neural information processing systems*, 33:6840–6851.

Matthew Honnibal, Ines Montani, Sofie Van Landeghem, and Adriane Boyd. 2020. spaCy: Industrial-strength natural language processing in python.

Kaiyi Huang, Chengqi Duan, Kaiyue Sun, Enze Xie, Zhenguo Li, and Xihui Liu. 2023a. T2i-compbench++: An enhanced and comprehensive benchmark for compositional text-to-image generation. *IEEE transactions on pattern analysis and machine intelligence*, PP.

Kaiyi Huang, Kaiyue Sun, Enze Xie, Zhenguo Li, and Xihui Liu. 2023b. T2i-compbench: A comprehensive benchmark for open-world compositional text-to-image generation. *Advances in Neural Information Processing Systems*, 36:78723–78747.

Ziwei Huang, Wanggui He, Quanyu Long, Yandi Wang, Haoyuan Li, Zhelun Yu, Fangxun Shu, Weilong Dai, Hao Jiang, Fei Wu, and Leilei Gan. 2025. T2I-FactualBench: Benchmarking the factuality of text-to-image models with knowledge-intensive concepts. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 27501–27524, Vienna, Austria. Association for Computational Linguistics.

Sarthak Jain and Byron C. Wallace. 2019. Attention is not Explanation. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 3543–3556, Minneapolis, Minnesota. Association for Computational Linguistics.

Guy Kaplan, Matanel Oren, Yuval Reif, and Roy Schwartz. 2025. From tokens to words: On the inner lexicon of llms. *Preprint*, arXiv:2410.05864.

Nelson F. Liu, Matt Gardner, Yonatan Belinkov, Matthew E. Peters, and Noah A. Smith. 2019. Linguistic knowledge and transferability of contextual representations. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 1073–1094, Minneapolis, Minnesota. Association for Computational Linguistics.

nostalgebraist. 2020. Interpreting GPT: The logit lens. lesswrong, 2020.

OpenAI, Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, Red Avila, Igor Babuschkin, Suchir Balaji, Valerie Balcom, Paul Baltescu, Haiming Bao, Mohammad Bavarian, Jeff Belgum, and 262 others. 2024. Gpt-4 technical report. *Preprint*, arXiv:2303.08774.

Danish Pruthi, Mansi Gupta, Bhuwan Dhingra, Graham Neubig, and Zachary C. Lipton. 2020. Learning to deceive with attention-based explanations. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 4782–4793, Online. Association for Computational Linguistics.

Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, and 1 others. 2021. Learning transferable visual models from natural language supervision. In *International conference on machine learning*, pages 8748–8763. PmLR.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. 2019. Exploring the limits of transfer learning with a unified text-to-text transformer. *Preprint*, arXiv:1910.10683.

Royi Rassin, Eran Hirsch, Daniel Glickman, Shauli Ravfogel, Yoav Goldberg, and Gal Chechik. 2023. Linguistic binding in diffusion models: Enhancing attribute correspondence through attention map alignment. *Advances in Neural Information Processing Systems*, 36:3536–3559.

Royi Rassin, Shauli Ravfogel, and Yoav Goldberg. 2022. DALLE-2 is seeing double: Flaws in word-to-concept mapping in Text2Image models. In *Proceedings of the Fifth BlackboxNLP Workshop on Analyzing and Interpreting Neural Networks for NLP*, pages 335–345, Abu Dhabi, United Arab Emirates (Hybrid). Association for Computational Linguistics.

Chitwan Saharia, William Chan, Saurabh Saxena, Lala Li, Jay Whang, Emily L Denton, Kamyar Ghasemipour, Raphael Gontijo Lopes, Burcu Karagol Ayan, Tim Salimans, and 1 others. 2022. Photorealistic text-to-image diffusion models with deep language understanding. *Advances in Neural Information Processing Systems*, 35:36479–36494.

Axel Sauer, Dominik Lorenz, Andreas Blattmann, and Robin Rombach. 2023. Adversarial diffusion distillation. *Preprint*, arXiv:2311.17042.

Yang Song and Stefano Ermon. 2019. Generative modeling by estimating gradients of the data distribution. *Advances in neural information processing systems*, 32.

Xianfeng Tan, Yuhan Li, Wenxiang Shang, Yubo Wu, Jian Wang, Xuanhong Chen, Yi Zhang, Ran Lin, and Bingbing Ni. 2024. Ragdiffusion: Faithful cloth generation via external knowledge assimilation. *Preprint*, arXiv:2411.19528.

Raphael Tang, Linqing Liu, Akshat Pandey, Zhiying Jiang, Gefei Yang, Karun Kumar, Pontus Stenetorp, Jimmy Lin, and Ferhan Ture. 2023. What the DAAM: Interpreting stable diffusion using cross attention. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 5644–5659, Toronto, Canada. Association for Computational Linguistics.

Gemma Team, Thomas Mesnard, Cassidy Hardin, Robert Dadashi, Surya Bhupatiraju, Shreya Pathak, Laurent Sifre, Morgane Rivière, Mihir Sanjay Kale, Juliette Love, and 1 others. 2024. Gemma: Open models based on gemini research and technology. *arXiv preprint arXiv:2403.08295*.

Michael Toker, Ido Galil, Hadas Orgad, Rinon Gal, Yoad Tewel, Gal Chechik, and Yonatan Belinkov. 2025. Padding tone: A mechanistic analysis of padding tokens in t2i models. *Preprint*, arXiv:2501.06751.

Michael Toker, Hadas Orgad, Mor Ventura, Dana Arad, and Yonatan Belinkov. 2024. Diffusion lens: Interpreting text encoders in text-to-image pipelines. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 9713–9728, Bangkok, Thailand. Association for Computational Linguistics.

Jesse Vig and Yonatan Belinkov. 2019. Analyzing the structure of attention in a transformer language model. In *Proceedings of the 2019 ACL Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP*, pages 63–76, Florence, Italy. Association for Computational Linguistics.

Peng Wang, Shuai Bai, Sinan Tan, Shijie Wang, Zhihao Fan, Jinze Bai, Keqin Chen, Xuejing Liu, Jialin Wang, Wenbin Ge, Yang Fan, Kai Dang, Mengfei Du, Xuancheng Ren, Rui Men, Dayiheng Liu, Chang Zhou, Jingren Zhou, and Junyang Lin. 2024. Qwen2-vl: Enhancing vision-language model's perception of the world at any resolution. *arXiv preprint arXiv:2409.12191*.

Enze Xie, Junsong Chen, Junyu Chen, Han Cai, Haotian Tang, Yujun Lin, Zhekai Zhang, Muyang Li, Ligeng Zhu, Yao Lu, and Song Han. 2024a. Sana: Efficient high-resolution image synthesis with linear diffusion transformer. *Preprint*, arXiv:2410.10629.

Enze Xie, Junsong Chen, Junyu Chen, Han Cai, Haotian Tang, Yujun Lin, Zhekai Zhang, Muyang Li, Ligeng Zhu, Yao Lu, and Song Han. 2024b. Sana: Efficient high-resolution image synthesis with linear diffusion transformers. *Preprint*, arXiv:2410.10629.

Jiahui Yu, Yuanzhong Xu, Jing Yu Koh, Thang Luong, Gunjan Baid, Zirui Wang, Vijay Vasudevan, Alexander Ku, and 1 others. 2022. Scaling autoregressive models for content-rich text-to-image generation. *arXiv preprint arXiv:2206.10789*, 2(3):5.

Kelly Zhang and Samuel Bowman. 2018. Language modeling teaches you more than translation does: Lessons learned through auxiliary syntactic task analysis. In *Proceedings of the 2018 EMNLP Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP*, pages 359–361.

A Appendix

A.1 Technical Details

A.1.1 Lexical Item Classification

We define a *lexical item* as either a single word or a compound expression of multiple words that, in context, conveys a unified semantic meaning. A compound expression is treated as a single item when its words form a fixed lexical unit with cohesive semantics rather than merely exhibiting a modifier–head relationship. For example, while "broken mirror" describes a mirror's state, expressions like "hot air balloon" or "teddy bear" denote entities with distinct identities. Similarly, although phrases such as "identical twins" or "baseball bat" might be interpreted as separate concepts, conventional usage supports their treatment as unified entities. We employ the reasoning model `03-mini-high` as a classifier to tag multi-word lexical items in both the target prompts and the dataset. The model returns a list of identified multi-word expressions, while the remaining untagged words are treated as individual lexical items.

A.1.2 Evaluation Visual Generations.

We evaluate whether an image matches a textual description using Qwen2-VL-72B-Instruct (Wang et al., 2024). The following prompt is used:

> "In Yes, No and maybe. Does every image match one of those descriptions: (description string)? Answer Yes if all images match or relate to at least one description, Maybe if only some match, otherwise No."

Here, the *textual description* can be either a single lexical item or a complete textual prompt.

A.1.3 Evaluating Generated Textual Descriptions.

We employ GPT-4o (OpenAI et al., 2024) to evaluate the textual interpretations produced by Patchscopes. We use the following prompt:

> "In Yes, No and Maybe. Does every image match the description: {Patchscopes_description} ? Answer Yes if all images match or relate to the description, Maybe if only some match, otherwise No."

A.1.4 Evaluating Relations Between Items.

We enhance our leakage validation by distinguishing between cases where two lexical items exhibiting semantic leakage are perceptually bound together—for example, "old" and "man" in the prompt "a portrait of an old man"—and cases where they are not as "cone hat" and "eating" in the prompt "A person searing a cone hat is eating" (see Fig. 5). To achieve this, we use a large language model (LLM) as a judge. Specifically, we use GPT-4o and employ the following prompt:

> "In Yes or No: in this prompt: {input_prompt}, are {item_1} and {item_2} perceptually bound together?"

We then filter out all cases where the lexical items are perceptually bound together and find that only 6.5% instances exhibit unintentional leakage.

A.1.5 Intended Item Evaluation

For each lexical-item, we manually create two interpretations - one in the intended interpretation from the prompt, and another is a possible wrong interpretation of the word in other contexts. For example, given the prompt "A standing zebra to the right of a city bus station", the current interpretations would be "the animal zebra", while the incorrect interpretations would be "A zebra crossing". We then ask a VLM to evaluate the generated image, and asses if the first or the latter interpretations exists in the images.

To evaluate the model's capacity for contextual disambiguation, we manually define two interpretations for each lexical item in each prompt. The first is the intended semantic meaning derived from the prompt's context, and the second is a alternative interpretation, that is wrong in this context. For instance, in the prompt "A standing zebra to the right of a city bus station", the intended meaning of "zebra is the animal", whereas the wrong interpretation is a "zebra crossing". Subsequently, a VLM analyzes the generated image to determine if it depicts the intended interpretation or the wrong interpretation as we define it.

A.1.6 Resources

Our computational experiments involved inference with four distinct text-to-image models: Flux-dev, Flux-schnell, sdxl-turbo, and Sana. The parameter sizes for these models are approximately 12 billion for Flux-dev and Flux-schnell, 3.1 billion for sdxl-turbo, and a range of 0.6 to 4.8 billion for the Sana models, with our experiments utilizing a 1.6 billion parameter version. The total computational budget for these experiments is estimated to be approximately 480 GPU hours. The computing infrastructure consisted of a cluster of eight NVIDIA A100 GPUs. This configuration provided the necessary computational power for the large number of inference tasks performed.

A.1.7 Use of AI Assistants.

We utilized AI assistants to support this research. For coding the experiments, we used Microsoft's Copilot and Anthropic's Claude 3; all generated code was manually reviewed and validated by us to ensure it aligned with our requirements. For the paper, Google's Gemini models (Pro and Flash) were used to improve the writing and clarity. We have carefully reviewed all content to ensure it accurately reflects our intentions.

A.2 Data

DrawBench (Saharia et al., 2022): We include all categories except for "misspelling", "rare words", and "text". Overall we extract 134 prompts from DrawBench.

In total, we obtain 1,056 prompts.

Extended dataset of leakage prompts. Our augmentation process incorporates two components. First, we generate variations of existing prompts from (Rassin et al., 2022) (e.g., modifying 'a gentleman with a bow in the forest' to 'a man wearing a bow in the jungle'). Second, we introduce novel prompts with potential semantic leakage. For these prompts, we apply a one-lexical item change test by generating an image from a similar prompt that substitutes the affected or leaked item with an alternative term (e.g., replacing 'bishops' with 'cardinals' or 'checkers' in 'chess in '2 bishops playing chess'). This test ensures that minimal lexical modifications do not alter the intended semantic meaning while producing a different image due to semantic leakage from another item in the prompt (see the first two columns in Fig. 7 for few visual examples). Together, these methods enrich the dataset and provide a robust framework for analyzing semantic leakage. The full list of prompts is available in an anonymous Git repository.[6]

A.3 Additional models

A.3.1 FLUX-Dev

In addition to our primary experiments with FLUX, we repeated all analyses using the Flux-dev variant. The redundant versus representative token experiments yielded similar trends, with 55% of tokens identified as representative and 45% as non-representative—values closely matching those observed with FLUX. Likewise, our inter-item flow experiments confirmed that information flow occurred in 11% of cases (and 3.1% miss intended leakage), reinforcing the overall patterns reported in the main text. Notably, while the aggregate trends are consistent across models, the specific lexical items resolved can differ between FLUX-schnell and Flux-dev, indicating potential a slightly different inner-lexicon (Kaplan et al., 2025). These findings underscore the robustness of our approach while highlighting model-dependent nuances in token representation and information flow dynamics.

[6] https://anonymous.4open.science/r/TokenRole-3E19

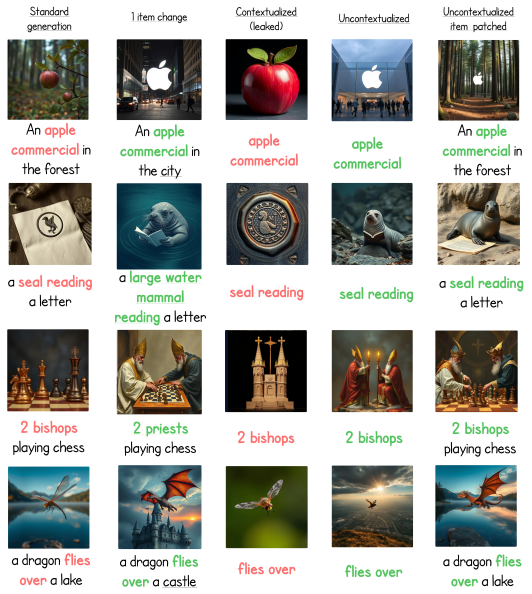

Figure 7: Examples from our semantic leakage method. **Left**: standard generation of leakage contained prompt. **Second**: generation using a one-lexical item change test as part of the dataset creation (a minimal substitution to verify that a slight lexical change yields a different image). **Third**: image from the contextual representation (misinterpreted item). **Forth**: image from the uncontextualized representation (correct interpretation). **Right**: final generation after patching the correct, uncontextualized representation into the prompt.

A.3.2 SDXL-Turbo

Figure 8: Images generated from individual subtokens in SDXL-Turbo. We find that, in many cases, the representation of an item is not clearly reflected in any of its subtokens—for example, in the case of the token "skateboard." Another interesting observation is that the last token of a lexical item often carries its representation, as seen in the "square" token of "times square." We also observe that the EOT token incorporates information from the full prompt.

Our analysis reveals that SDXL-Turbo, which uses the CLIP text encoder, behaves markedly differently from FLUX, which relies on the encoder in the encoder-decoder T5-XXL. In SDXL-Turbo, the text encoder is a causal language model, meaning each token's encoding is influenced only by its preceding tokens during the encoding process.

We repeated our *in-item* representation experiments using SDXL-Turbo. Our first observation is that most generated images are either abstract or unrelated to the intended lexical items. According to our analysis, 55% of lexical items in SDXL-Turbo lack any representative token (compared to just 11% in FLUX). Moreover, when a representative token is present in CLIP, it is typically the final token of the lexical item (see Fig. 8). This is aligned with the unidirectional encoding of the model.

Another phenomenon we observe—consistent with CLIP's training objective—is the unusually dominant role of the end-of-sequence (EOS) token. Images generated from the EOS token often encapsulate nearly the full semantic content of the prompt. In our evaluation, 62% of EOS-generated images matched the prompt (compared to 73% when using the full prompt). We believe this also causes our intervention method to be less effective, since when we interpret a single token, we patch all other tokens, including the EOT token—which usually contains a lot of information—with tokens derived from an empty prompt (see Fig. 8).

A.3.3 Sana (Gemma-based encoder)

We also evaluate our methodology on the Sana model (Xie et al., 2024b), which uses a Gemma-based autoregressive language model as the text encoder. These results help validate the generality of our findings across architectures with differing encoding strategies.

For *in-item information flow*, we observe that Sana produces fewer multi-token lexical items due to its larger vocabulary relative to T5 and CLIP. In cases where items do consist of multiple tokens, we find that only the *last* token typically acts as a representative token—a behavior aligned with the unidirectional nature of autoregressive encoders.

In our *cross-item information flow* analysis, we find that 17.45% of item pairs in Sana exhibit contextual information flow, compared to 10.45% in FLUX. This increase likely stems from the one-sided (forward-only) nature of Sana's encoding. For example, in the prompt *"a black baseball hat with a flame decal on it"*, we find contextual influence such as "black baseball" affecting "hat" and "flame" affecting "decal", but not the reverse. See Fig. 9 for illustrations.

Overall, while Sana shows a slightly higher rate of contextual influence than FLUX, it preserves many of the key structural insights found in our primary analysis—particularly the sparsity and location of representative tokens. Unlike CLIP, which

Figure 9: Token-level image generation using Sana on the prompt *"a black baseball hat with a flame decal on it"*. Representative information often resides in the last token of each item (e.g., "hat", "decal"), consistent with Sana's causal encoding. Contextual influence is one-directional, with earlier tokens shaping later ones.

encodes substantial information in the EOS token, Sana lacks such artifacts, suggesting that EOS-related effects are not fundamental to autoregressive models more generally.

A.4 Inter-Item Information Flow Framework

To assess whether one lexical item encodes information about others in the same prompt, we conduct the following experiment.

Given a prompt, we isolate each lexical item one at a time. First, we encode the full prompt using the text encoder. Then, for a given lexical item, we apply our patching method (Section 2) to mask the representations of all other tokens—leaving only the contextualized representation of the target item intact. Formally, for a lexical item with token indices $S \subset \{1, \ldots, N\}$, we construct a patched sequence $\tilde{t}_1, \ldots, \tilde{t}_N$ in which only the tokens in S retain their original contextualized representations. We then generate an image from this modified sequence, capturing what information is encoded in the selected item's contextualized form.

We repeat this process for the same item in an uncontextualized setting: we encode and generate an image using only that lexical item in isolation, without the rest of the prompt. This allows us to distinguish between information inherently present in the item's encoding and information introduced by context.

To measure influence between items, we use Qwen2-VL to check whether a second item y appears in the image generated from a first item x, both in the contextualized and uncontextualized versions. Influence is considered *True* if y appears in the image generated from contextualized x, but not from uncontextualized x. This comparison ensures that the observed presence of y is attributable to contextual information flow during encoding, rather than coincidental co-occurrence or inherent correlation.

This setup enables us to detect breaches of lexical segmentation—i.e., cases where one lexical item encodes visual information belonging to another—quantifying interdependence across items within the prompt.

See Fig. 11 for an illustration of the procedure.

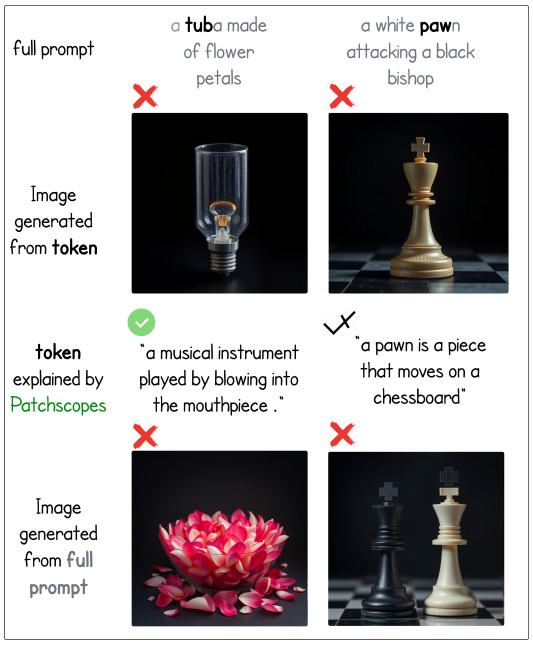

Figure 10: Comparing Patchscopes to image generation with token-level patching in cases of *item negligence*. We assess whether a token's concept is preserved by comparing images generated from its contextual representation, Patchscopes' textual interpretation, and the full prompt image. **Left:** "tub" (from "tuba") is correctly described by Patchscopes but fails to ground visually. **Right:** "paw" (from "pawn") has missing details by both interpretations, suggesting a gap in the encoder's conceptual knowledge.

A.5 Analysis of Negligence: Understanding Omitted Items

We analyze cases where a lexical item is completely absent from the generated image, a phenomenon we refer to as *item negligence* (Chang et al., 2024; Chefer et al., 2023a). As shown in Fig. 3, when a lexical item has no representative tokens, it is also missing from an image generated using all of its

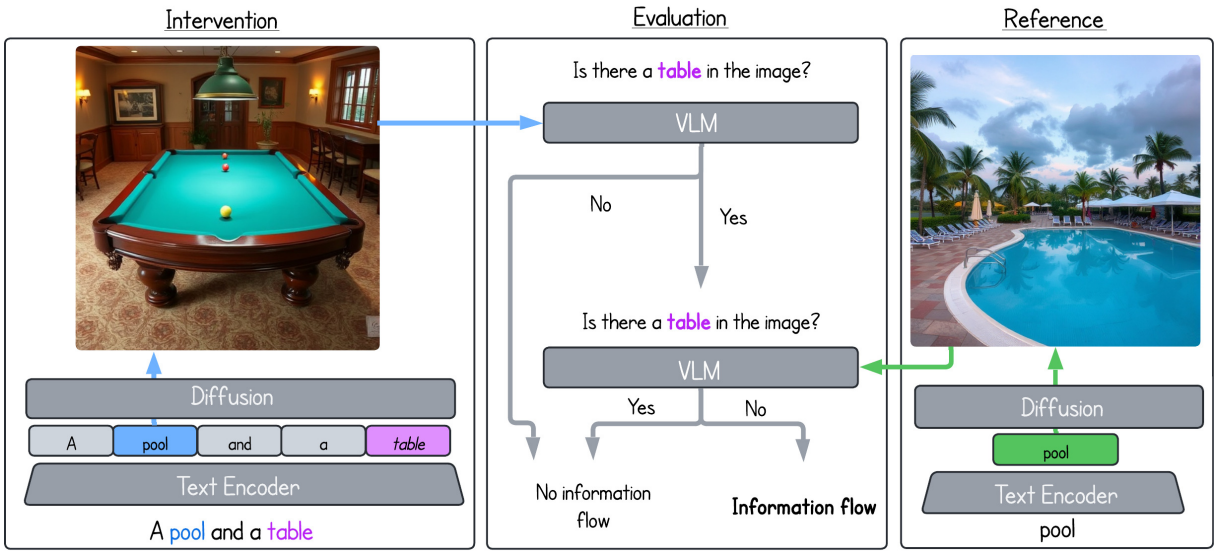

Figure 11: **Evaluating inter-item information flow:** Our proposed framework to interpret the information flow between lexical items in the prompt. For each lexical item, we generate an image from its contextual representations (left), and from it's uncontextualized representation (right), and analyze the generated images using a VLM (middle). In this example, we interpret the item "pool" and assess whether it is influenced by the item "table". To do so, we ask a VLM whether the image generated from the token "pool" contains a "table". To ensure this is the result of information flow, and not a natural correlation between a pool and tables, we generate the item "pool" without context (right-hand side), and verify that "table" is not present in this image. If this is the case, we conclude that there was information flow from "table" to "pool".

tokens in 88% of cases. This strong correlation suggests a direct connection between the absence of representative tokens and the failure to visually render certain concepts.

Hypotheses To understand why this occurs, we consider two potential explanations. First, the text encoder may fail to capture the concept adequately, resulting in a weak or missing internal representation. Second, the concept may be well-encoded in text, but the diffusion model itself may lack familiarity with the concept's visual appearance, preventing it from rendering the item correctly.

Methodology To distinguish between these two possibilities, we analyze the text encoder independently of the diffusion model using Patchscopes (Ghandeharioun et al., 2024). Patchscopes decodes token-level representations into natural language descriptions through the full encoder-decoder architecture of T5. For each neglected item, we patch its encoding into the template:

```
describe <item>
```

We then evaluate whether the resulting description accurately represents the intended concept using GPT-4o (OpenAI et al., 2024). Details of the GPT-4o evaluation prompt are provided in Appendix A.1.3. This procedure is applied to all neglected lexical items identified in Section 4.1.

Results Our analysis shows that in 67% of the neglected cases, Patchscopes returns accurate descriptions. For instance, as illustrated in Fig. 10, the token sequence for "tuba" yields the description: *"a musical instrument played by blowing into the mouthpiece."*

These results suggest that for the majority of neglected items, the text encoder correctly encodes the concept, and the failure likely arises within the diffusion model's visual grounding process. In the remaining 33% of cases, the Patchscopes outputs are incomplete or incorrect, indicating that the text encoder itself lacks a robust representation of the item.

Interpretation Taken together, these findings point to two distinct sources of item-level negligence: (1) representational gaps in the text encoder, where the concept is poorly encoded or missing entirely; and (2) visual grounding failures in the diffusion model, where the concept is well-encoded but cannot be visually rendered.

The first failure mode aligns with prior work showing that strengthening textual descriptions can

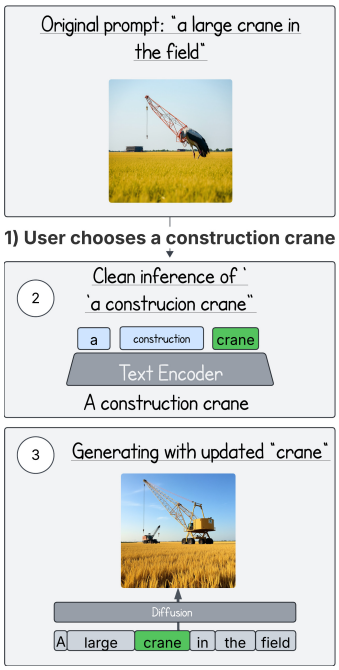

Figure 12: **Resolving polysemous words using patching.** Example: in the prompt *"a crane in the field"*, the word **crane** may refer either to a construction machine or to a bird. (1) The user specifies the intended meaning. (2) The prompt is encoded with this interpretation. (3) We patch the encoded representation and generate an image, producing a result consistent with the user's choice.

improve factual generation (Huang et al., 2025). The second highlights a limitation of the diffusion model's training distribution and its capacity to map textual meaning to visual form.

Additional qualitative examples of these two cases are provided in Figure 10, where we show how different concepts behave under our Patchscopes-based evaluation.

A.6 Resolving Polysemous Words via Patching

Polysemous words such as "bat," "bow," or "crane" can correspond to multiple meanings, and current T2I models often default to the most common interpretation or fail to disambiguate. In some cases, users may even prefer a surprising reading (e.g., *flying baseball bats on top of a baseball stadium*), or the context alone may not suffice (e.g., *a gentleman wearing a bow in the forest*).

To address this, we extend our patching technique to allow explicit user control over interpretation. Given a prompt and a user-specified meaning, we replace the ambiguous token with a clean, context-independent representation of the intended concept, while leaving the surrounding context unchanged.

We demonstrate this approach over several ambiguous words, each paired with two distinct interpretations. Ambiguous prompts were generated using GPT-4o, which also produced disambiguated representations of each meaning. For every prompt, we generated images and used a vision-language model (VLM) to assess whether the output matched the user-specified interpretation. Our method achieved a 95% success rate in aligning images with the chosen interpretation.

As shown in Figure 12, this patching-based approach enables precise and reliable resolution of lexical ambiguity, offering improved control for both faithful image generation and creative use cases.

A.7 Mitigating Bias via Encoder-Side Patching

Biases can manifest as *meaning shifts* already at the textual encoding stage, similar to the misinterpretations discussed in §5.1. In particular, gendered cues in the prompt may steer the encoded representation of a target item toward an unintended sense, producing biased or inconsistent generations. For example, in the prompts "a **businesswoman** on a *runway*" and "a **businessman** on a *runway*", the representation of *runway* changes: in the first case it encodes a *fashion runway*, while in the second it encodes an *airport runway*. This indicates that gender context can alter the semantic encoding of otherwise unambiguous items.

We address this bias using the same *patching* technique introduced for correcting semantic leakage (see Fig. 13 for an illustration of the process). We replace the biased, contextualized representation of the affected token (e.g., *runway*) with its clean, uncontextualized representation, while keeping the rest of the prompt fixed. This minimal intervention neutralizes the gender-induced drift and restores the intended interpretation—here, generating a **businesswoman** on an *airport runway*. This approach provides a lightweight, encoder-level mitigation for bias-driven meaning shifts, complementing diffusion-stage fairness methods.

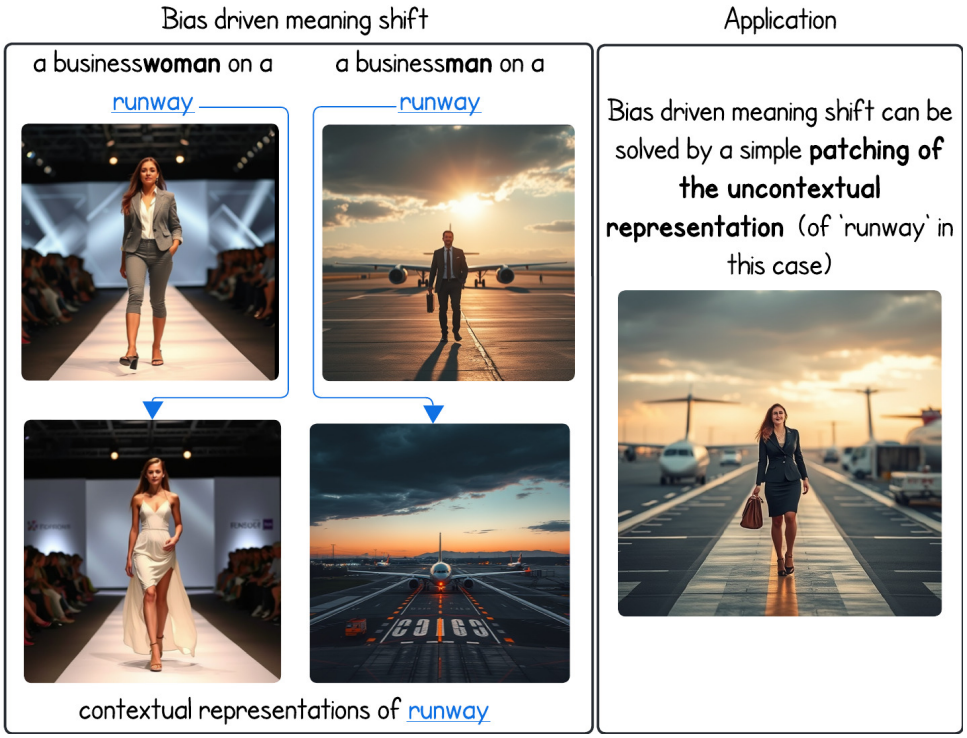

Figure 13: **Bias-driven meaning shift and mitigation.** *Left:* The top row show generations for "a business**woman** on a *runway*" (fashion runway) and "a business**man** on a *runway*" (airport runway). The bottom row shows generations from the contextualized token *runway* alone, revealing how gender context distorts its encoding. *Right:* Patching the *runway* token with its uncontextualized representation mitigates this bias, yielding an unbiased image of a businesswoman on an airport runway.