Region-Level Data Attribution for Text-to-Image Generative Models

Trong Bang Nguyen[1] Phi Le Nguyen[1*] Simon Lucey[2] Minh Hoai[2]
[1] Institute for AI Innovation and Societal Impact, Hanoi University of Science and Technology
[2] Australian Institute of Machine Learning, The University of Adelaide

Abstract

Data attribution in text-to-image generative models is a crucial yet underexplored problem, particularly at the regional level, where identifying the most influential training regions for generated content can enhance transparency, copyright protection, and error diagnosis. Existing data attribution methods either operate at the whole-image level or lack scalability for large-scale generative models.

In this work, we propose a novel framework for region-level data attribution. At its core is the Attribution Region (AR) detector, which localizes influential regions in training images used by the text-to-image generative model. To support this research, we construct a large-scale synthetic dataset with ground-truth region-level attributions, enabling both training and evaluation of our method. Empirical results show that our method outperforms existing attribution techniques in accurately tracing generated content back to training data. Additionally, we demonstrate practical applications, including identifying artifacts in generated images and suggesting improved replacements for generated content. Our dataset and framework will be released to advance further research in region-level data attribution for generative models. Code is available at: https://github.com/AIoT-Lab-BKAI/AR-Detector.

1. Introduction

Einstein once remarked, "The secret to creativity is knowing how to hide your sources," and modern text-to-image generative models embody this irony remarkably well. These models exhibit impressive "creativity," generating novel images with coherent compositions of objects, attributes, and styles, yet their training on vast datasets makes it difficult to trace the origins of individual elements. This lack of traceability, however, raises legal and practical concerns for both model operators and consumers of generated images. To help address this issue, we study the task of tracing the origins of individual objects in generated images—an essential capability for detecting copyright and ownership violations and for correcting errors by identifying their sources and addressing them at the root.

Given an object in a generated image, our goal is to determine which regions of training samples have the most significant influence on its generation and to what extent. This is a challenging task, as it requires attributing influence to specific regions within training images rather than treating each image as a whole. Theoretically, data attribution involves analyzing model outputs when trained with and without a particular data sample whose influence we seek to determine. Traditional approaches in this area include methods such as influence functions [12, 24], which approximate loss changes, and Shapley-value-based techniques [6, 25], which estimate attribution by training the model on multiple subsets of the dataset. However, these approaches are not scalable for modern generative models, which comprise billions of images and parameters, nor do they support region-level attribution within each image. While more recent techniques for studying large generative models have emerged [26, 27, 29], the field is still in its early stages, and existing methods consider only entire training images, without attributing influence to specific regions within them.

In this paper, we propose a method for performing region-level data attribution in text-to-image generative models. At the core of our method is a framework for training an Attribution Region (AR) detector tailored to a given text-to-image generative model G. The AR detector localizes the region R_X within a training sample X that most significantly contributes to the formation of a given region R_Q within an image Q generated by G from the text prompt p. Additionally, the AR detector computes an attribution score that quantifies the extent to which R_X influences R_Q. By using the AR detector to scan and analyze multiple regions within training images, we can identify the regions that exert the greatest influence on the generated output.

This AR detector may appear similar to an object detector, but unlike traditional object detection, it must be tailored to a specific generation process and detect regions that have been absorbed and transformed through a complex, poorly understood generative mechanism. Addition-

*Corresponding Author

ally, unlike conventional detectors that detect objects from predefined categories, the AR detector must also process additional textual and visual prompts to determine what needs to be detected. A key challenge lies in obtaining training data and establishing a reliable ground truth connection between a generated object and its influential regions within training images. There is no efficient way to make a deep generative model forget specific objects/regions in the training data, making it difficult to measure the direct attribution of the removed objects/regions.

In this work, we develop the AR detector based on an open-vocabulary detector [16], which is specifically designed to accept various types of prompts. To quantify attribution scores, we enhance the original architecture with two adapter modules. To address the challenge of obtaining training data, we leverage the insight that attribution can be measured not only by removing an element and assessing its negative impact but also by enhancing it and measuring its additive impact. Specifically, we adopt the customization approach [4] where a pretrained text-to-image model is fine-tuned to generate images that explicitly contain specified objects or concepts. This establishes a direct attribution between the generated content and the provided inputs, enabling the creation of training data for the AR detector.

Using the proposed approach, we construct a large-scale synthetic dataset with ground truth annotations to train and evaluate our region-level data attribution method. Empirical experiments on this dataset demonstrate that our method surpasses existing attribution techniques in accurately identifying and assigning attribution scores to training image regions. To further advance research in region-level data attribution, we will release this dataset as a benchmark for future studies. Additionally, we showcase a practical application of our approach, demonstrating how it can diagnose issues in generated images and suggest more suitable replacements at the source, improving alignment with user intent. These contributions lay the groundwork for more interpretable and controllable generative models, enhancing both transparency and practical utility.

2. Related Work

Text-to-image synthesis. The recent progress of diffusion models has led to impressive text-to-image generation capabilities, enabling the creation of high-quality and diverse images [20, 23]. These diffusion-based models have advanced various downstream applications, such as image editing [9, 28] and image inpainting [5, 15]. Additionally, a key challenge in image generation is the ability to quickly learn and synthesize new concepts without disrupting previously learned knowledge. To address this, several "customization" methods have been proposed [4, 13, 22]. In this paper, we demonstrate the effectiveness of our region-level data attribution framework for Stable Diffusion [21], though our approach is applicable to other models as well.

Data attribution. To identify the attribution of a training subset to a model's output for a given sample, early methods first define an objective function and then evaluate how its output changes after removing the subset from the training process. Koh and Liang [12] proposed influence functions to analyze classifier behavior and suggested using inverse-Hessian products [1] to reduce the computational cost of approximating influence scores. Pruthi et al. [19] further reduced this cost by omitting the Hessian matrix and instead relying on gradients saved during model training. However, these approaches are not scalable or practical for deep models with a large number of parameters trained on extensive datasets.

Alternative training-based approaches have been explored to address this issue. Ilyas et al. [8] proposed learning a linear data-model that directly predicts influence scores based on a binary indicator vector for a given subset. TRAK [18], a method that leverages Taylor expansion to linearize a one-step Newton approximation for influence scores, achieves a state-of-the-art balance between accuracy and computational efficiency when measuring attribution for discriminative models.

Several recent works have explored the problem of data attribution for diffusion models. Some studies extend TRAK [14, 29] to estimate the influence function's value at different denoising steps. Wang et al. [27], building on the mirrored hypothesis [11], employs an unlearning method to approximate the influence of a test sample on training samples and identify the most affected one. Other approaches, such as ProMark [3] and Attribution-by-Customization [26], define data attribution intuitively by analyzing the visual representation of the entire image.

In contrast to these methods, our approach leverages an object detector to extract the target region in training images that contributes the most to the generated region. We demonstrate that by extending attribution measurement to the regional level, a broader range of real-world applications can be addressed.

3. Attribution Region Detectors

The core of our region-level attribution framework is the Attribution Region (AR) detector, which identifies the most influential region in a training image that contributes to the formation of a region of interest in a generated image. The problem can be formally defined as follows.

Given a region of interest R_Q, specified by a bounding box in a generated image Q and an associated textual prompt p, the AR detector $\mathcal{F}$ aims to identify the most influential region $\hat{R}_X$ in a training image X and compute its attribution score S_X:

$$(\hat{R}_X, S_X) = \mathcal{F}(X, p, Q, R_Q), \tag{1}$$

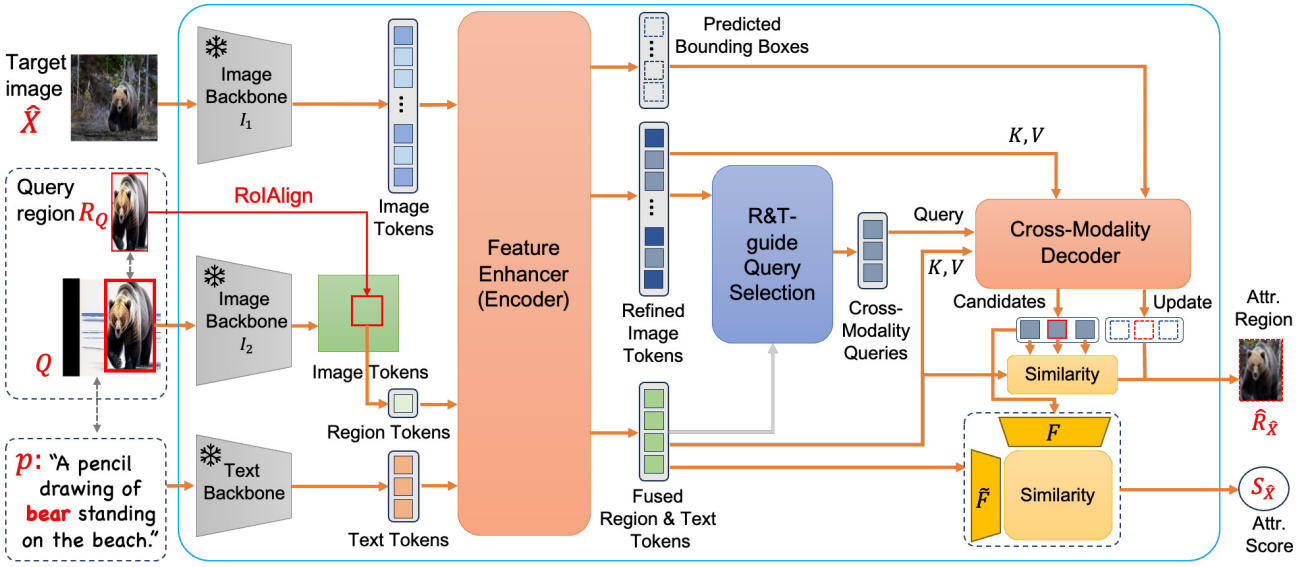


Figure 1. **The architecture of our method consists of two phases**. In Phase 1, we omit F and $\tilde{F}$ and train the Encoder and Decoder using only positive samples. In Phase 2, the Encoder and Decoder are frozen, while the two adapters, F and $\tilde{F}$, are introduced and trained with both positive and negative samples. Backbones I_1 and I_2 are the same in our work but separated in the figure to clarify processing paths.

where $\hat{R}_X$ is a four-dimensional vector representing the bounding box coordinates, and S_X denotes the confidence score indicating its influence. In the following subsections, we first describe the architecture of the AR detector, followed by its training process.

3.1. Model Architecture

Fig. 1 illustrates the architecture of the proposed AR detector. It builds upon CountGD [2], an open-world counting and detection model capable of accepting both text and image regions as prompts, with modifications to the image encoder and the addition of two adapters, as described below.

Image backbones. CountGD is a network designed to count and detect object instances in an image based on a textual prompt and/or a visual query. In CountGD, a single image backbone processes both the target image and the visual query, as the visual query is simply a subregion of the target image. However, in our case, the visual query originates from an image generated by a text-to-image generative model, which differs from the target image on which the detector must operate. To accommodate this difference, we introduce an additional image backbone, I_2, for the query image Q, while retaining I_1 as the backbone for the target image $\hat{X}$, as illustrated in Fig. 1.

To maintain both simplicity and effective utilization of pretrained models, we use a pretrained SwinTransformer-Base for both I_1 and I_2. The query region R_Q is then represented using tokens extracted through a structured three-step process: (1) all feature maps produced by I_2 are upscaled to a consistent resolution, (2) the upscaled feature maps are concatenated and projected into 256 channels via a 1×1 convolution, and (3) RoIAlign is applied to extract region tokens based on the bounding box coordinates of R_Q.

Text backbone. We simply use a pretrained BERT-Base as text backbone.

Token adapters. We incorporate two adapters into our model: F for the target image tokens and $\tilde{F}$ for the query tokens (a fusion of textual and visual tokens). These adapters map the target and query tokens into the same space while enhancing the predicted attribution scores. In our experiments, both F and $\tilde{F}$ are implemented as linear layers.

Output. Our method operates similarly to other transformer-based object detectors, such as GroundingDINO [16], which generates N_c candidate regions from the decoder and refines predictions using a similarity matrix. For our region-level attribution task, we need to output only the single most influential region, $(\hat{R}_{\hat{X}}, S_{\hat{X}})$, which corresponds to the maximum entry in the similarity matrix.

3.2. Training

We now describe the training process for the AR detector. We assume access to a training dataset of the form $\{(Q_i, R_{Q_i}, p_i, X_i, R_{X_i})\}$, where Q_i is the output of a text-to-image generative model given the text prompt p_i, R_{Q_i} is the query region of interest, and R_{X_i} is the ground-truth region that contributes most to the generation of R_{Q_i}. The region R_{X_i} is contained within the training image X_i.

We refer to this dataset as the *positive dataset*, as each X_i contains a region that contributes to the formation of the query region R_{Q_i}. Additionally, we assume the exis-

tence of a *negative dataset* for each data sample in the positive dataset, consisting of multiple images $\{X_{i1}^{-}, X_{i2}^{-}, \ldots\}$, where none of the regions within these images contribute to the generation of R_{Q_i}.

We train the AR detector in two phases, starting from the pretrained backbones of CountGD. In the first phase, we train the model using only positive data, allowing it to localize the ground-truth attribution region within an image. In the second phase, we fine-tune the model with both positive and negative data to effectively learn attribution scores. Throughout both phases, we keep the image and text backbones frozen while optimizing the remaining components of the model.

Phase 1: Training with positive data. In this phase, we remove the two adapters, F and $\tilde{F}$, and train the remaining components of the AR detector similarly to a standard object detector. This is achieved by minimizing a loss function that combines a classification loss, which determines whether a region is an object of interest, and a regression loss, which refines the bounding box localization for detected objects.

Phase 2: Fine-tuning with both positive and negative data. To avoid damaging what has been learned during Phase 1, we introduce the adapters F and $\tilde{F}$ into the score head and optimize only these adapters during this training phase. To enhance the learning of attribution scores, we incorporate negative samples to encourage the model to capture contrastive properties between positive and negative instances. Specifically, in each training epoch, we iterate over all positive samples and select a set of negative samples for each. Given a sample $(R_{Q_i}, Q_i, p_i, X_i, R_{X_i})$ we randomly select k negative samples from $\{X_{i1}^{-}, X_{i2}^{-}, \ldots\}$ (with $k = 4$ in our experiments) and combine them with the corresponding positive sample to form a batch of size $k+1$. This batch is then used to fine-tune F and $\tilde{F}$. For optimization, we employ focal loss as the primary loss function, where ground-truth attribution regions in positive samples are assigned a score of 1. Additionally, we introduce a regularization term to constrain the weights of F and $\tilde{F}$, ensuring stability and preventing overfitting. The overall loss function is:

$$\lambda_{cls}\mathcal{L}_{focal}(\hat{\mathcal{S}}, \mathcal{T}) + \lambda_{reg}\mathcal{L}_{reg}(F, \tilde{F}), \tag{2}$$

where $\hat{\mathcal{S}}$ represents the similarity matrix between N candidate regions and c query tokens, and $\mathcal{T}$ depicts the target matching between candidates and query tokens. Specifically, $\mathcal{T} \in \{0, 1\}^{N \times c}$ for positive samples and $\mathcal{T} \in \{0\}^{N \times c}$ for negative ones. The regularization term is defined as

$$\mathcal{L}_{reg}(F, \tilde{F}) = \frac{1}{2}\left(\|FF^T - \mathbf{I}\|_2 + \|\tilde{F}\tilde{F}^T - \mathbf{I}\|_2\right), \tag{3}$$

where $\mathbf{I}$ is the identity matrix. The coefficients λ_{cls} and λ_{reg} are hyperparameters that balance the classification and regularization terms.

4. Dataset Generation

This section outlines the process of generating training data for AR detectors, which requires ground-truth information for region-level attribution. To achieve this, we employ a customization approach that adapts a text-to-image model to a novel concept/object present in an exemplar image. This method enables the generation of exemplar-generated image pairs, where the generated images contain the target concept or object. Since the generated content originates from the exemplars, we can directly attribute it back to them, establishing the necessary ground-truth associations for training.

4.1. Text-to-image Model Customization

To generate an image where the origin of specific regions can be traced, a customized text-to-image generative model is required. We begin with a diverse set of exemplar images, denoted as $\mathcal{X}$, and a set of predefined concepts (e.g., `cat`, `person`, `car`), denoted as $\mathcal{C}$, for customization. To ensure that the generated concept is primarily influenced by the exemplar rather than pre-existing knowledge in the generative model, it is preferable that $\mathcal{X}$ does not overlap with the dataset used to train the original generator. In this study, we use the validation data of the COCO and LVIS datasets, which are distinct from those used to train Stable Diffusion [21].

Given an exemplar image X_i sampled from $\mathcal{X}$ and a base concept $\mathbf{c}_b$ (e.g., `cat`), we first use the open-set object detector GroundingDINO [16] to detect a bounding box R_{X_i} corresponding to an instance of $\mathbf{c}_b$ in X_i. If no such instance is detected, the image-concept pair is discarded and the process is repeated. We then apply the Segment Anything Model (SAM) [10], a state-of-the-art segmentation model, to refine R_{X_i} into a segmentation mask $\mathcal{M}_i$.

Next, we construct a modified prompt for customization based on the base concept $\mathbf{c}_b$ by introducing a special text token to represent a 'personalized' concept $\mathbf{c}_i$ (e.g., `V* cat`). Finally, we employ Break-a-Scene [4] with the masked object $\mathcal{M}_i$ to align the concept across different denoising steps and optimize the token representation for $\mathbf{c}_i$.

The output of this process is a customized text-to-image generative model G_i capable of generating novel images that include the learned concept exemplified by the masked region $\mathcal{M}_i$ within a new scene defined by a modified text prompt, such as "`A V* cat lying on a sofa`." This customization process takes approximately seven minutes per exemplar with an 80GB A100 GPU.

4.2. Prompt template

The customized model G_i can generate images that contain the learned concept in a scene described by a text prompt. In this work, we construct text prompts using predefined templates with the assistance of ChatGPT.

To prevent concept-context mismatches that may lead to unnatural prompts, such as `"A V* cat is hanging on a hook in the kitchen"`, we implement a preprocessing step before template creation. First, we instruct ChatGPT to categorize all base classes in our dataset into semantic groups (e.g., living creatures, wearables, vehicles). For each group, we generate a diverse set of templates reflecting typical poses or locations, such as `"A <concept> is lying on soft grass in a rural setting."` for living creatures and `"A <concept> is parked near a quiet countryside road."` for the transportation group. Additionally, we extend each concept's appearance variations with templates like `"<medium> of a <concept> sitting on a rock"`, where `<medium>` could be replaced with `"A charcoal sketch"` or `"An ink drawing"`. Overall, we define 10 diverse semantic groups, each containing 10 distinct templates, five emphasizing contextual variations and five focusing on different visual representations of the concept.

This approach ensures meaningful and diverse text prompts, leading to more coherent and realistic image generation. All prompts used to query ChatGPT and the resulting templates are summarized in Supplementary C.

4.3. Image Generation and Dataset Creation

Given the customized generative model G_i for the concept R_{X_i} of an exemplar image X_i, as described in Sec. 4.1, and a text prompt p_i obtained as in Sec. 4.2, we generate an image Q_i that incorporates the learned concept, which can be directly attributed to the region R_{X_i}. Next, we use GroundingDINO [16] to detect the bounding box R_{Q_i} of the learned concept in Q_i. As a result, the process yields the following 5-tuple: $(R_{Q_i}, Q_i, p_i, X_i, R_{X_i})$. We refer to R_{Q_i} as the *query* region and Q_i the query image, where our goal is to identify the source concept primarily responsible for generating this query region.

We define the 5-tuple $(R_{Q_i}, Q_i, p_i, X_i, R_{X_i})$ as a *positive sample*, since the exemplar image X_i contains the region R_{X_i}, which is primarily attributed to the creation of the query region. We put all such positive samples together to create a set of positive samples:

$$\mathcal{AD}^+ = \{(Q_i, R_{Q_i}, p_i, X_i, R_{X_i})\}_i, \tag{4}$$

Next, we construct a *negative set* $\mathcal{AD}_i^- = \{X_{i1}^-, X_{i2}^-, \ldots\}$ for each positive sample. Each image X_{ij}^- is selected from the set of exemplar images $\{X_1, X_2, \ldots\}$ such that it is visually distinct from X_i. To ensure diverse training and testing samples, we prevent duplications within the negative set. The complete set of negative samples is defined as:

$$\mathcal{AD}^- = \bigcup_i \mathcal{AD}_i^-. \tag{5}$$

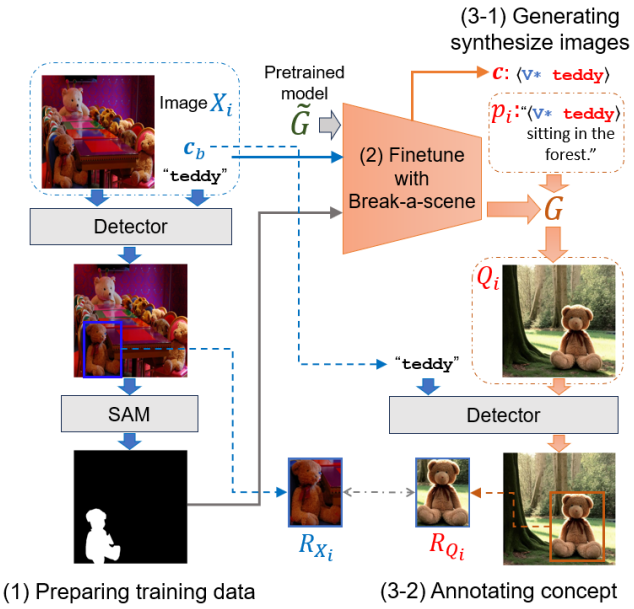


Figure 2. **Illustration of our data generation process.**

Table 1. **Statistic of our curated dataset.** In which, 'M' and 'K' represent for million and thousand unit, respectively.

Dataset	# concepts	# positive samples train/val/test	# negative samples train/val/test
COCO	10	7.5K / 1K / 1K	5.5M / 84K / 85K
LVIS	477	19.8K / 2.7K / 2.6K	25M / 415K / 409K

We partition $\mathcal{AD}^+$ into disjoint training, validation, test sets and achieve the corresponding sets $\mathcal{AD}^-$. The sizes of these sets are provided in Table 1.

5. Experiments

In this section, we begin by outlining the experimental setup, followed by a discussion of the results and the applications of regional data attribution.

5.1. Experiment Settings

Evaluation metrics. We evaluate the performance of our region-level attribution method by measuring its ability to accurately localize attribution regions within training images and assign appropriate scores to them. To this end, we employ a ranking-based metric that requires ground-truth regions in training images to be not only correctly identified but also assigned higher scores than other regions.

Specifically, for each positive test sample $(R_{Q_i}, Q_i, p_i, X_i, R_{X_i})$ and its corresponding negative set $\{X_{ij}^-\}$, we run the AR detector on X_i to obtain $\hat{R}_{X_i}$, the predicted most influential region, along with the attribution score S_{X_i}. We apply the same procedure to the negative set, obtaining attribution scores $\{S_{X_{ij}^-}\}$. Let R be the rank

of S_{X_i} within the sorted list of scores $S_{X_i}, S_{i1}^-, S_{i2}^-, \ldots$ (ordered from largest to smallest). We then compute two evaluation metrics: **Recall@K**, which is 1 if $R \leq K$ and 0 otherwise, and **InverseRank@K**, which is $1/R$ if $R \leq K$ and 0 otherwise. For both metrics, the value is set to 0 if the predicted attribution region $\hat{R}_{X_i}$ and the ground-truth region R_{X_i} have an IoU of less than 0.5, as we consider both correct localization and accurate score assignment.

Comparison Baselines. As no existing methods specifically address the region attribution problem, we compare our approach with two of the strongest baselines we could devise and implement: image-level attribution and few-shot object detection. We adapt these approaches to our regional data attribution task as follows.

- **Image-level attribution methods.** We select Attribution by Customization (**AbC**) [26] and Attribution by Unlearning (**AbU**) [27] as baseline methods. AbC directly quantifies similarity at the image level, while AbU applies an unlearning process to the query image Q and measures its influence based on the resulting loss changes in the training images X_i. To quantitatively evaluate these methods, we use image-level attribution scores returned by AbC and AbU to compute Recall@K and InverseRank@K.
- **Few-shot object detectors.** Our method operates similarly to a one-shot detector, as it takes a single visual prompt for the concept of interest and detects instances of this concept in images. A natural question, then, is how it compares to few-shot detectors. To investigate this, we consider two state-of-the-art few-shot object detection models: **VFA** [7] and **ICPE** [17]. To adapt these methods to our problem, we treat the query region R_{Q_i} as a novel concept and fine-tune a few-shot detector using R_{Q_i} and Q_i. We then apply the detector to X_i and X_{ij}^- and obtain the highest detection scores for the images and compute the Recall@K and InverseRank@K as described above.

Hyperparameters. We use the same hyperparameters for both the model architecture and training process as recommended in GroundingDINO [16] for the COCO and LVIS datasets. In the second training phase, we set $\lambda_{cls} = 1$ and tuning λ_{reg} during training process. Main experimental results are depicted for $\lambda_{reg} = 0.5$.

5.2. Main Results

Quantitative result. Table 2 presents a comprehensive comparison of our method against baseline approaches. In addition to ranking metrics, we report localization accuracy across all positive samples in $\mathcal{AD}^+$ to evaluate the effectiveness of the detector-based methods in localization tasks. Compared to VFA and ICPE, our approach consistently demonstrates a 3–10% improvement in detection accuracy. Regarding ranking metrics, our method outperforms all baselines, achieving a 34% higher Recall@1 and up to 226.2% higher Recall@10 than the second-best approach in COCO-type datasets. Concerning the LVIS-type datasets, although our performance is slightly affected by the increased number of base classes, we still obtain a 111.2% improvement in Recall@10. This significant advantage stems from our model's ability to leverage text-based queries, whereas VFA and ICPE rely solely on image-based queries, limiting their adaptability. Further comparisons with pretrained CountGD and baseline performance discussion are provided in the Supplementary D.

Qualitative result. We further illustrate the effectiveness of our method by comparing it with AbC in the top-10 retrieved results for a query image generated from the text prompt: "`A <V* fork> is lying on an antique wooden desk`". Due to space constraints, additional comparisons with other baselines are provided in the Supplementary B. As shown in Fig. 13, our method consistently retrieves highly relevant regions belonging to the base class "`fork`", whereas AbC is often misled by the overall scene layout, prioritizing similar colors (as seen in the first and the fifth images of the second row) or stylistic resemblance (as observed in the sixth image of the second row). Furthermore, the low attribution scores assigned by AbC across all retrieved images (smaller than 0.3) indicate significant uncertainty in its predictions, underscoring the limitations of image-level attribution methods when applied to region-level tasks. In contrast, our approach provides accurate predictions, correctly identifying the ground-truth image (the first image in the first row) with a high confidence score of 0.7391.

5.3. Ablation Studies

To assess the impact of key components in our framework, we conduct an ablation study examining the role of the two adapters F and $\tilde{F}$ as well as the second phase of our training process. Additionally, we explore the dependency of our method on different backbone architectures by substituting the default text and image backbones with alternative options. Our current model utilizes SwinT-Base as the image backbone and BERT-Base as the text backbone. To investigate their influence, we evaluate three different configurations: (1) **B-B:** SwinT-Base image backbone + BERT-Base text backbone (default), (2) **T-B:** SwinT-Tiny image backbone + BERT-Base text backbone, and (3) **B-L:** SwinT-Base image backbone + BERT-Large text backbone. Since no pretrained CountGD models are available for SwinT-Tiny or BERT-Large, we initialize these variants using pretrained GroundingDINO models instead. The experimental results are presented in Table 3. Among the three configurations, **B-B** consistently outperforms both **T-B** and **B-L**, highlighting the importance of a well-pretrained model for both visual and textual representations. Furthermore, we observe that fine-tuning in the second training phase (de-

Table 2. **Quantitative comparison between our method and other baselines.** The best-performing results are highlighted in **bold-red**. Values in parentheses represent the relative improvement of our method over the second-best approach.

Dataset	Method	Box acc.	InverseRank@5	Recall@1	Recall@5	Recall@10
COCO	AbC	N/A	0.1059	0.0817	0.1643	0.2079
	AbU	N/A	0.0771	0.0594	0.1239	0.1452
	ICPE	0.86	0.1143	0.0793	0.1752	0.2231
	VFA	0.83	0.1026	0.0759	0.1508	0.2115
	Ours	**0.89**	**0.2293** (+100.6%)	**0.1095** (+34.0%)	**0.4620** (+163.7%)	**0.7278** (+226.2%)
LVIS	AbC	N/A	0.0574	0.0507	0.0774	0.1061
	AbU	N/A	0.0494	0.0417	0.0644	0.0813
	ICPE	0.62	0.0812	0.0607	0.1304	0.1665
	VFA	0.64	0.0724	0.0621	0.1179	0.1593
	Ours	**0.72**	**0.1534** (+88.9%)	**0.0968** (+55.9%)	**0.2531** (+94.1%)	**0.3516** (+111.2%)

noted as **ARD** in Table 3) significantly improves performance across all three versions compared to models trained without this phase (**ARD–**). The enhancement is particularly noticeable in Recall@5, where ARD achieves an improvement of up to 6%, increasing from 0.4425 in ARD– (B-B) to 0.4696 in ARD(B-B). These findings underscore the effectiveness of the two adapters and demonstrate that fine-tuning with both positive and negative samples is essential.

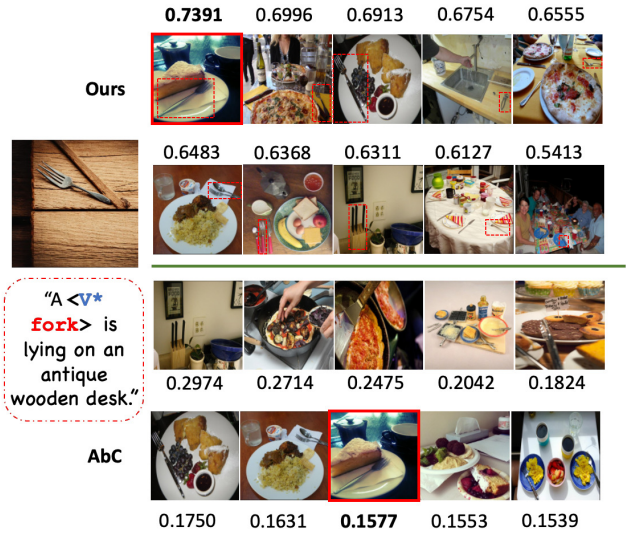


Figure 3. **Qualitative comparison between our method and AbC**. Our method accurately localizes the attributing region and assigns a high score, whereas AbC fails to do so.

Table 3. **Comparison of different variants of our method**. ARD represents our full approach with $\lambda_{reg} = 1.0$, while ARD– is the version without the second training phase.

	Recall@10		Recall@5	
	ARD–	ARD	ARD–	ARD
B-B	0.704	0.709	0.4425	0.4696
T-B	0.508	0.521	0.2701	0.2910
B-L	0.589	0.593	0.3124	0.3231

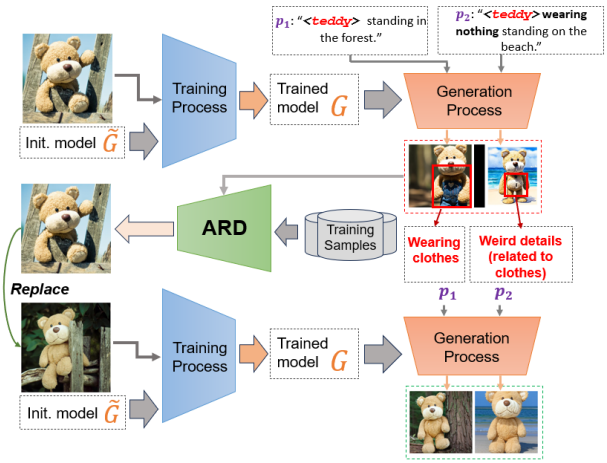


Figure 4. **Application of detail repair.** Our method retrieves concept region from training images to identify error-causing details and refine the training concept.

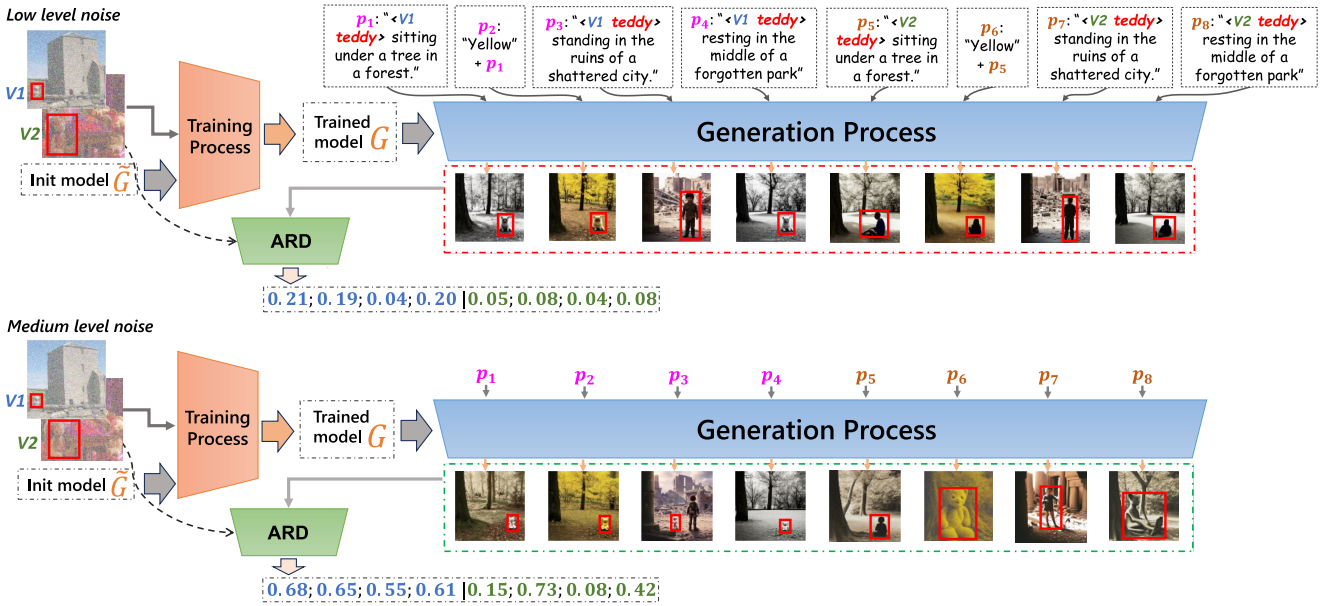


Figure 5. **Applying region-level attribution to investigate model's behavior with noisy training images**.

5.4. Applications

We present several examples to highlight the practical applications of our proposed method. Fig. 4 illustrates how our method can be leveraged to correct errors in images generated by text-to-image models. In this example, we employ the customization process to quickly learn the special case of a teddy bear as well as illustrate our application. But we suggest that our method can be utilized in the general case of this application because we use any common classes (*e.g.*, "`cat`") as text input for our method to retrieve the origin of the generated concept. In the general case, when realizing the unexpected details in the generated concept of teddy (with prompt p_1) and it is not addressed well even after refining the prompt to p_2, we could use an AR detector to identify the corresponding training image and the specific attributing region. This analysis reveals that the object was originally partially covered by a wooden fence, making it difficult for the model to reconstruct hidden details (such as the teddy bear's torso) in the generated image. To resolve this, we modify the training data by removing the occlusion and reusing the updated image to train the model again, resulting in better quality of generated teddy. This application is successfully depicted with customization as the training process, and it raises promising success in the general case with an arbitrary training strategy.

Furthermore, we conduct an experiment to investigate how a text-to-image model behaves when trained on images with varying levels of Gaussian noise (Fig. 5). The results indicate that in the low-noise case, all generated concepts fail to match the base class ("teddy") of the training concepts. Specifically, concepts generated from prompts p_1, p_2, p_4, p_6, p_8 resemble a dog, while others appear more human-like. In contrast, the medium-noise case produces only two mismatched concepts. To analyze these effects, we employ the AR detector to measure attribution scores and observe that the generative model G is more sensitive to low levels of Gaussian noise. Traditional data attribution methods, which assess entire images, are ineffective in this scenario because they treat images with the same text prompt as identical, regardless of noise level. For example, under the same text prompt p_2, the difference between low-noise and medium-noise cases appears only at the regional level, while the overall layout remains unchanged. This demonstrates the effectiveness of our approach in this application.

6. Conclusion

We proposed a framework for region-level attribution, which aimed to localize the most influential region within a training image and quantify its attribution score relative to a region of interest in a synthesized image generated by a text-to-image model. To support this task, we constructed a dedicated dataset for regional attribution and developed our method by adapting the architecture of existing open-vocabulary object detectors. Our approach was the first to perform region-level data attribution, surpassing both image-level attribution methods and few-shot object detection. Additionally, we demonstrated its potential for detailed image correction at the source. The limitations of our method are discussed in Supplementary G.

Acknowledgments. This project was initiated with support from the University of Adelaide's Global Engagement Fund. Subsequent support was provided in part by the Australian Institute for Machine Learning (University of Adelaide), the Centre for Augmented Reasoning (an initiative of the Department of Education, Australian Government), and Hanoi University of Science and Technology grant number T2024-TD-002.

References

[1] Naman Agarwal, Brian Bullins, and Elad Hazan. Second-order stochastic optimization for machine learning in linear time. *J. Machine Learning Research*, 18(116):1–40, 2017. 2

[2] Niki Amini-Naieni, Tengda Han, and Andrew Zisserman. Countgd: Multi-modal open-world counting. In *NeurIPS*, 2024. 3

[3] Vishal Asnani, John Collomosse, Tu Bui, Xiaoming Liu, and Shruti Agarwal. Promark: Proactive diffusion watermarking for causal attribution. In *Proc. CVPR*, 2024. 2

[4] Omri Avrahami, Kfir Aberman, Ohad Fried, Daniel Cohen-Or, and Dani Lischinski. Break-a-scene: Extracting multiple concepts from a single image. In *Proc. ACM SIGGRAPH Asia*, 2023. 2, 4

[5] Ciprian Corneanu, Raghudeep Gadde, and Aleix M Martinez. Latentpaint: Image inpainting in latent space with diffusion models. In *Proc. WACV*, 2024. 2

[6] Amirata Ghorbani and James Zou. Data shapley: Equitable valuation of data for machine learning. In *Proc. ICML*, 2019. 1

[7] Jiaming Han, Yuqiang Ren, Jian Ding, Ke Yan, and Gui-Song Xia. Few-shot object detection via variational feature aggregation. In *Proc. AAAI*, 2023. 6

[8] Andrew Ilyas, Sung Min Park, Logan Engstrom, Guillaume Leclerc, and Aleksander Madry. Datamodels: Predicting predictions from training data. In *Proc. ICML*, 2022. 2

[9] Bahjat Kawar, Shiran Zada, Oran Lang, Omer Tov, Huiwen Chang, Tali Dekel, Inbar Mosseri, and Michal Irani. Imagic: Text-based real image editing with diffusion models. In *Proc. CVPR*, 2023. 2

[10] Alexander Kirillov, Eric Mintun, Nikhila Ravi, Hanzi Mao, Chloe Rolland, Laura Gustafson, Tete Xiao, Spencer Whitehead, Alexander C Berg, Wan-Yen Lo, et al. Segment anything. In *Proc. ICCV*, 2023. 4

[11] Myeongseob Ko, Feiyang Kang, Weiyan Shi, Ming Jin, Zhou Yu, and Ruoxi Jia. The mirrored influence hypothesis: Efficient data influence estimation by harnessing forward passes. In *Proc. CVPR*, 2024. 2

[12] Pang Wei Koh and Percy Liang. Understanding black-box predictions via influence functions. In *Proc. ICML*, 2017. 1, 2

[13] Nupur Kumari, Bingliang Zhang, Richard Zhang, Eli Shechtman, and Jun-Yan Zhu. Multi-concept customization of text-to-image diffusion. In *Proc. CVPR*, 2023. 2

[14] Chris Lin, Mingyu Lu, Chanwoo Kim, and Su-In Lee. An efficient framework for crediting data contributors of diffusion models. In *Proc. ICLR*, 2025. 2

[15] Haipeng Liu, Yang Wang, Biao Qian, Meng Wang, and Yong Rui. Structure matters: Tackling the semantic discrepancy in diffusion models for image inpainting. In *Proc. CVPR*, 2024. 2

[16] Shilong Liu, Zhaoyang Zeng, Tianhe Ren, Feng Li, Hao Zhang, Jie Yang, Qing Jiang, Chunyuan Li, Jianwei Yang, Hang Su, et al. Grounding dino: Marrying dino with grounded pre-training for open-set object detection. In *Proc. ECCV*, 2024. 2, 3, 4, 5, 6

[17] Xiaonan Lu, Wenhui Diao, Yongqiang Mao, Junxi Li, Peijin Wang, Xian Sun, and Kun Fu. Breaking immutable: Information-coupled prototype elaboration for few-shot object detection. In *Proc. AAAI*, 2023. 6

[18] Sung Min Park, Kristian Georgiev, Andrew Ilyas, Guillaume Leclerc, and Aleksander Madry. Trak: Attributing model behavior at scale. In *Proc. ICML*, 2023. 2

[19] Garima Pruthi, Frederick Liu, Satyen Kale, and Mukund Sundararajan. Estimating training data influence by tracing gradient descent. In *NeurIPS*, 2020. 2

[20] Aditya Ramesh, Prafulla Dhariwal, Alex Nichol, Casey Chu, and Mark Chen. Hierarchical text-conditional image generation with clip latents. *arXiv:2204.06125*, 2022. 2

[21] Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. High-resolution image synthesis with latent diffusion models. In *Proc. CVPR*, 2022. 2, 4

[22] Nataniel Ruiz, Yuanzhen Li, Varun Jampani, Yael Pritch, Michael Rubinstein, and Kfir Aberman. Dreambooth: Fine tuning text-to-image diffusion models for subject-driven generation. In *Proc. CVPR*, 2023. 2

[23] Chitwan Saharia, William Chan, Saurabh Saxena, Lala Li, Jay Whang, Emily L Denton, Kamyar Ghasemipour, Raphael Gontijo Lopes, Burcu Karagol Ayan, Tim Salimans, et al. Photorealistic text-to-image diffusion models with deep language understanding. In *NeurIPS*, 2022. 2

[24] Andrea Schioppa, Polina Zablotskaia, David Vilar, and Artem Sokolov. Scaling up influence functions. In *Proc. AAAI*, 2022. 1

[25] Jiachen T Wang and Ruoxi Jia. Data banzhaf: A robust data valuation framework for machine learning. In *Proc. AISTATS*, 2023. 1

[26] Sheng-Yu Wang, Alexei A Efros, Jun-Yan Zhu, and Richard Zhang. Evaluating data attribution for text-to-image models. In *Proc. ICCV*, 2023. 1, 2, 6

[27] Sheng-Yu Wang, Aaron Hertzmann, Alexei Efros, Jun-Yan Zhu, and Richard Zhang. Data attribution for text-to-image models by unlearning synthesized images. In *NeurIPS*, 2024. 1, 2, 6

[28] Binxin Yang, Shuyang Gu, Bo Zhang, Ting Zhang, Xuejin Chen, Xiaoyan Sun, Dong Chen, and Fang Wen. Paint by example: Exemplar-based image editing with diffusion models. In *Proc. CVPR*, 2023. 2

[29] Xiaosen Zheng, Tianyu Pang, Chao Du, Jing Jiang, and Min Lin. Intriguing properties of data attribution on diffusion models. In *Proc. ICLR*, 2024. 1, 2

Region-Level Data Attribution for Text-to-Image Generative Models

Supplementary Material

A. More applications

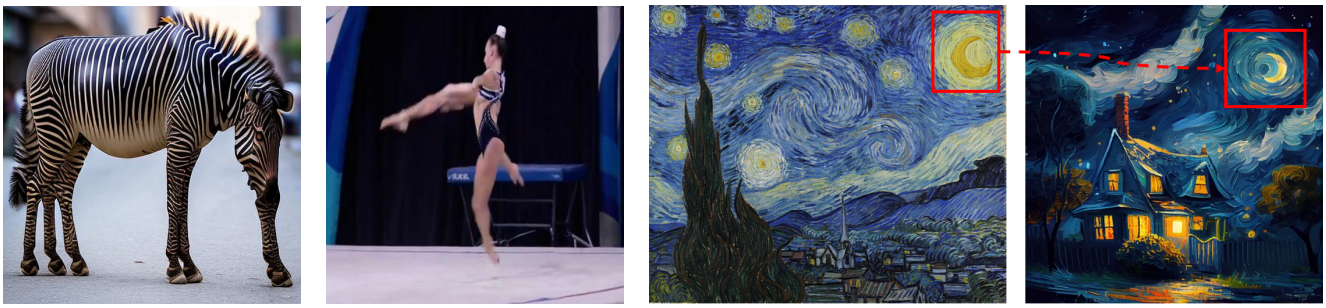
In practice, there are various applications that need the help of region-level understanding (Fig. 6). And in this section, we will define more such applications to depict the necessity of our region-level attribution method.

Position repair. We give here an additional example of repair application (Fig. 7), in which we train the model using a toy exemplar with a fixed position (top view) and regenerate it in the synthesized image. In both cases of p_1 and p_2, the returned positions of that toy seem unnatural due to the unclear view of the given concept in the training process. We employ ARD to find the region that the given concept occupies and fix this region only, resulting in better quality generated concept.

With the image-level attribution methods, we cannot extract exactly which region in the training image needs repair, especially when there is more than 1 object with the similar visual in the training image. Wrong object repair will lead to problems for other concepts while the current concept remains unfixed.

Understanding behaviors of generative model. Next, we present 3 different applications that utilize region-level data attribution to further investigate how the generative models behave in different settings.

- **Behaviors with different training concept's size.** This application is depicted in Fig. 8, in which Fig. 8a illustrates how we crop the concept for evaluating and Fig. 8b shows the quantitative correlation between concept's size (proportion of training image) and respective score. We first collect 10 concepts that their regions occupy less than 5% of their images. Then we perform the process in Fig. 8a separately for each concept, that we crop its image multiple times and resize the cropped to the size of full image. Each time returns a cropped image including the concept such that the area of the concept occupies $K\%$ of the cropped image ($K = 5, 10, 15, ..., 95, 100$). For each cropped image, we use it to train the generator and then synthesize the concept with 10 different text prompts. Next, we use ARD to measure the attribution score of these synthesized concepts to the original concept and take average to get the score regarding the concept at size of $K\%$. We finally take average of all scores of 10 concepts at size of $K\%$ to obtain the estimated Average Score at size of $K\%$. We plot Average Score at all values of K in Fig. 8b, the depicted pattern shows that the quality of the synthesized concept increases linearly with its size in the training image. If we measure the attribution with entire images, this application would never be done because: 1) the entire image including both obstacles and distracted background cannot be used to measure the quality of the concept accurately. 2) Experiments with qualitative results show that with medium and small concepts (less than 40% of the image), score from AbC is low (around 0.2) and can deflect the correlation.
- **Behaviors with changes in text input.** This application is showed in Fig. 9, that we test the generator *G* with different changes in input text prompts. In the first case, we slightly change the prompt with different appearance of the concept (resulting in p_2, p_3, p_4) and by the scores from our method, we can see that the generator *G* synthesizes the concept consistently under different changes of appearance. Meanwhile, when we change the background in text prompt (resulting in p_5, p_6, p_7, p_8), *G* sometimes produces less accurate concept (concept from p_7). All these sensitivities of the synthesized concept's quality can be estimated by our region-level data attribution method (0.57 in case of generated concept from p_7). Results from ARD suggest that generator is more sensitive with context in generating the given concept.
- **Behaviors with different noise levels in the training image.** We depict this application in Fig. 10 for Gaussian noise and Fig. 11 for Salt&Pepper noise. In both cases, we increase the noise level and show how well the generative model *G* can perform with different noise level in training image. This application is similar to what represented in the main paper, but clearer in details.
  In the case of **Gaussian noise** (Fig. 10), all generated concepts belonging to low noise case are mismatched with the base class ("`teddy`") of the training concepts (generated concepts from p_1, p_2, p_4, p_6, p_8 look like a dog and these others are similar to human). While number of mismatched concepts in the case of medium noise level is only 2. Here we utilize ARD to measure the attribution scores and observe that generative model *G* is more sensitive with low noise level of Gaussian. Other data attribution for entire image would not work in this case because it will treat images with the same text prompt in those two cases (low noise and medium noise) as the same. For example, for the same text prompt p_2, the difference between 2 cases only represents in region level, while it remains similar in general layout. Hence, our work can work effectively in this application.
  In the case of **Salt&Pepper noise** (Fig. 11), we perform the experiment similarly and estimate the scores for each case by ARD. Although pictures generated in case of medium noise often show additional object in it, we conclude from the output scores that generator *G* works on



(a) A six-legged zebra. (b) Distorted human limbs. (c) The right image shows the infringement in copyright of the left art.

Figure 6. **Example of synthesized images that meet region-level issue** (*e.g.*, distorted details, concept copyright).

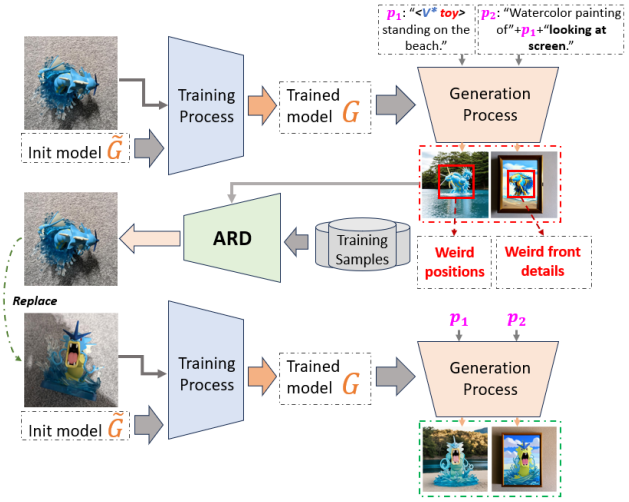


Figure 7. **Application of position repair.**

par in both cases of noise level (for the task of accurately synthesizing given concept).

B. Qualitative study

We show in Fig. 12 an additional qualitative result to further demonstrate the performance of our method compared to all baselines. We utilize text prompt "`A charcoal sketch of a <V* cellular telephone> with classic detailing.`" from the customized concept "`<V* cellular telephone>`" to synthesize the new image. Compared to other methods, our method shows the best result with highest score prediction, making our approach applicable to many region-level attribution applications. In the comparison, VFA and ICPE achieve quite a high score (around 0.5) but sometimes misidentify the objects (the second image in the second row is a Shinkansen train, while the query image is the telephone). On the other hand, when looking at the retrieval of AbC and AbU, we can realize that these methods focus more on the style (the first image in the last row may depict a visual similar to charcoal sketch) and the shape as well as the color of the concept (the first two images predicted by AbC show the same rectangle shape with the given concept). These results would suggest that image-level attribution methods are negatively affected by the background and are difficult to use effectively in region-level attribution problems.

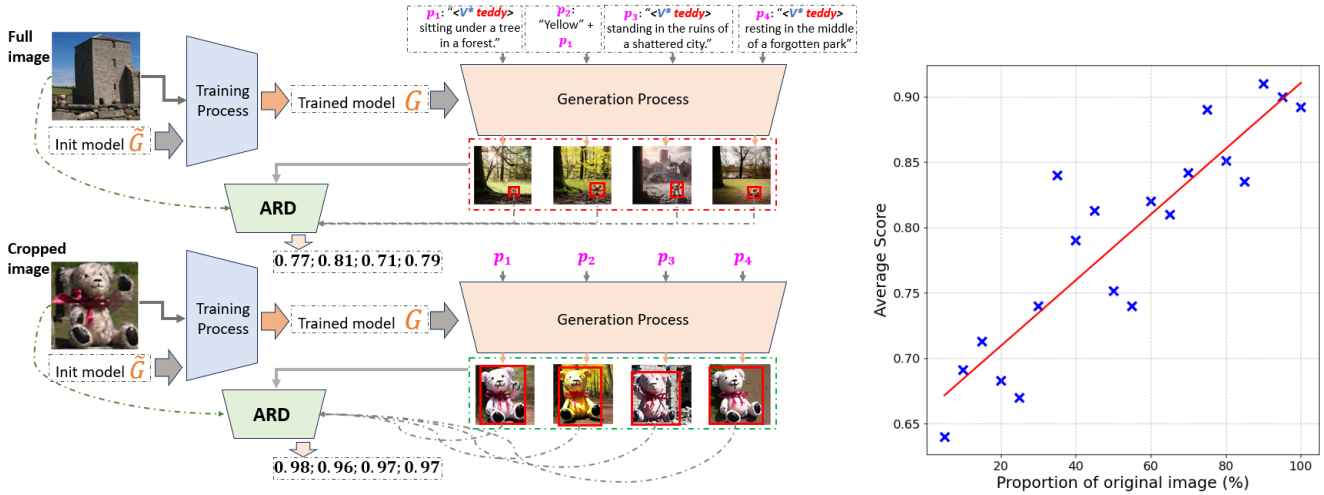
To further demonstrate the confusion of image-level method, we compare the performance of our method (ARD) and AbC - an image-level data attribution method. We use a prompt "`A <V* bench> is resting in the middle of a traditional Japanese tea room.`" from the customized concept "`<V* bench>`" and show this qualitative comparison in Fig. 13. Our objective here is to identify images that contribute to the concept "`<V* bench>`". However, the results illustrate that AbC is misled by the background image in some cases, particularly in the first image of the second row, where the tree in the background shares similar features (dense branches and reddish-pink leaves) with a tree in the query image. In addition, the low scores produced by AbC reflect its high uncertainty in measuring attribution.

C. Template configuration

In this section, we represent details of the text templates used in our main experiments. In the COCO-type dataset, we use a set of templates (Tab. 4) for all classes. For the LVIS-type dataset, we show details of templates used for each group in Tab. 5 and group division in Tab. 6.

D. Model configuration and Performance discussion

Definition of negative images. We define negatives for a query image (generated image) as all images that were not used during the customization of the pretrained model to generate the query image. As each exemplar/identity is used independently for customization in our settings with Break-a-Scene, these identities are considered separately



(a) An example of how to get images of the given concept with different sizes.

(b) Attribution score over different concept sizes.

Figure 8. **Application in understanding behavior with training concept's size.**

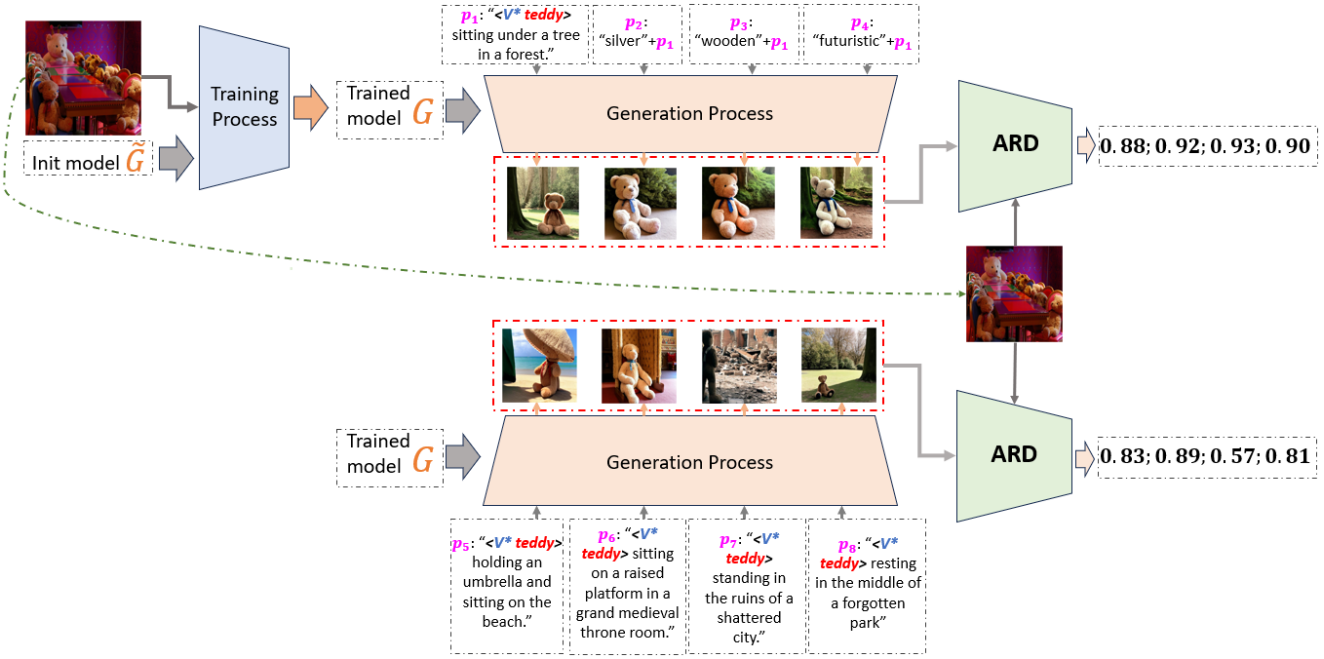


Figure 9. **Application in understanding behavior with different text prompts.**

when constructing our datasets. For example, when using a black-cat exemplar for customization, all other exemplars (whether cat images or non-cat images) are considered negatives.

Used version of Text-to-Image model. In our experiments, we use Stable Diffusion 2.1 as the text-to-image generative model to evaluate our ARD. This version of Stable Diffusion was also used as the base model in the original Break-a-Scene study. Since Break-a-Scene can be adapted to various generative models, we can synthesize a dataset from an off-the-shelf model to train ARD on the distribution of that model. Moreover, the performance of the pretrained CountGD model (shown in the next paragraph), along with ARD's ability to quickly adapt, demonstrates ARD's potential to generalize to other tasks. We expect our method to remain effective on the latest Stable Diffusion 3.5, as well as on other generative models across different tasks.

Comparisons with pretrained CountGD. We utilize the original pretrained CountGD (without any fine-tuning steps) to test with our LVIS dataset. The adapted CountGD

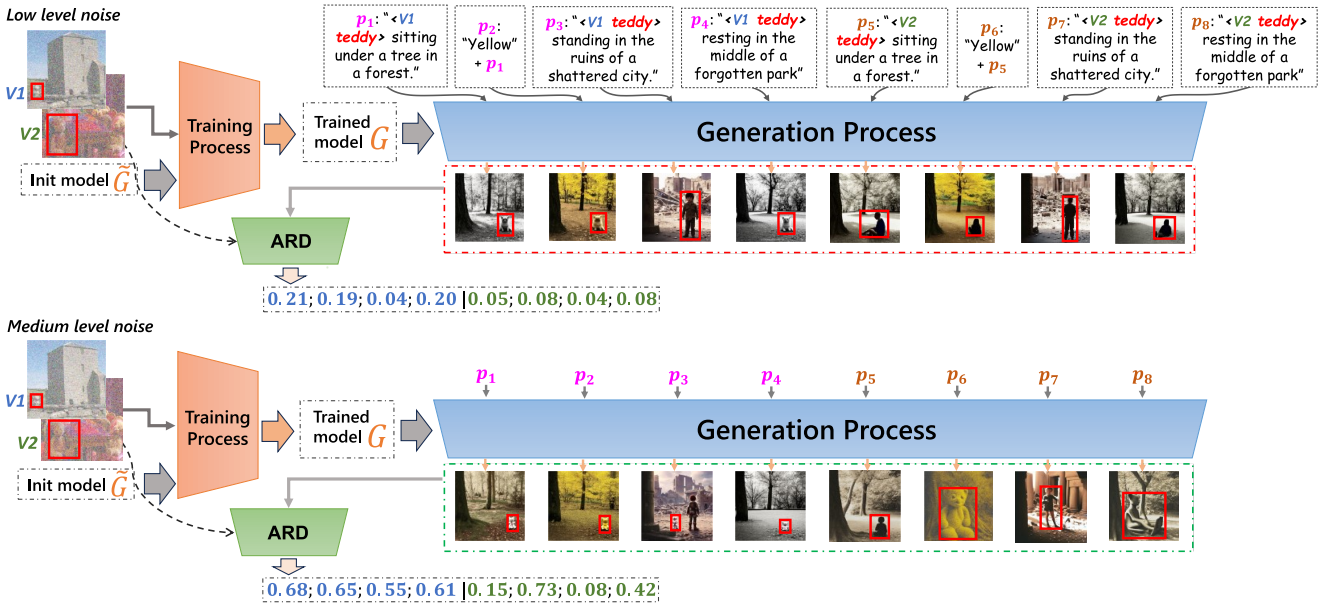


Figure 10. **Application in understanding behavior with noise:** Gaussian noise.

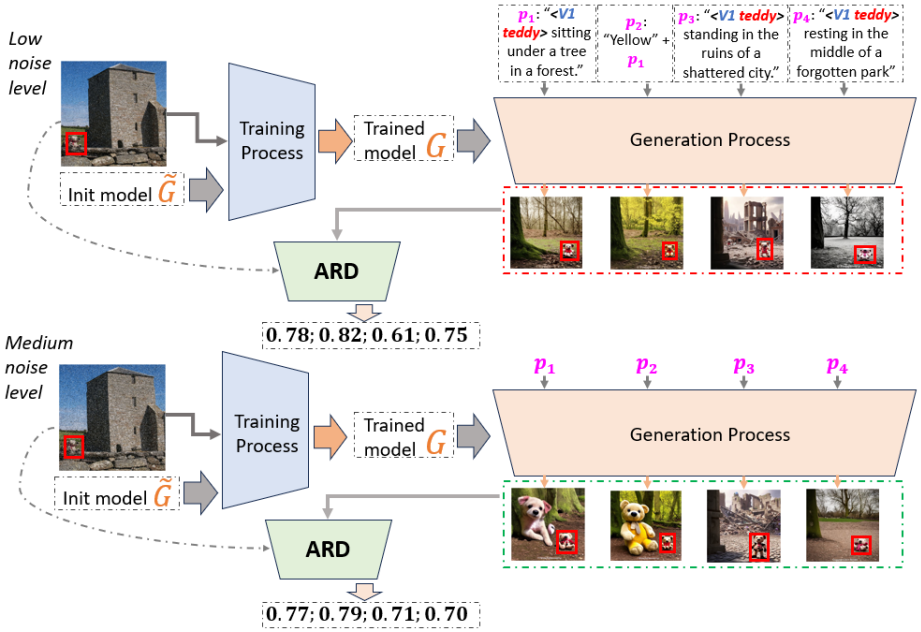


Figure 11. **Application in understanding behavior with noise:** Salt-and-Pepper noise.

model achieved Recall@1 of 0.0491, InverseRank@5 of 0.0749, and box accuracy of 43%, all notably lower than our method's results of 0.0968, 0.1534, and 72%, respectively. These results demonstrate that our model, ARD, has a strong starting point compared to other baselines. In addition, our training strategy effectively adapts the CountGD model to our attribution task and significantly enhances its performance.

Hyperparameter. In phase 2 of our model's training, we use a hyperparameter k to create a batch of samples. The value of k was chosen empirically; we tested $k = 2, 4, 6$ on the COCO dataset. The corresponding Recall@5 scores were 0.41, 0.46, and 0.44, respectively, with $k = 4$ yielding the best performance.

Settings and Performance of baselines. For image-level methods like AbC and AbU, we used them as-is to evaluate

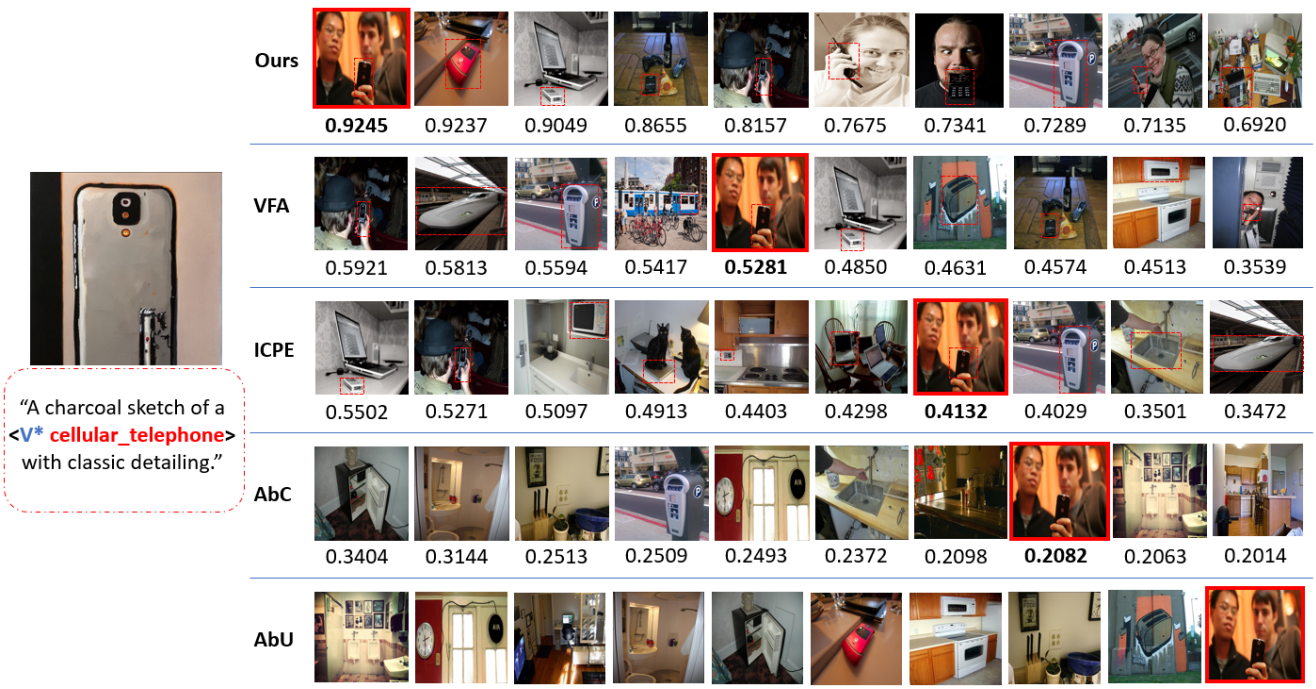


Figure 12. **Qualitative comparison between our method and all four baselines**, in which dashed boxes in the first three rows of (**Ours, VFA, ICPE**) represent the predicted boxes and the red boxes for entire image mean ground-truth target. We show top-10 retrievals along with predicted scores. Here we do not show the score predicted from **AbU** because this method estimates the retrieval ranking based on the change in loss.

Table 4. **Text templates for COCO-type dataset.**

Base class	**Text template**
bear, cat, horse, bird, dog, cow, teddy bear, zebra, sheep, elephant	"A photo of {} standing in a forest.", "A photo of {} standing near the lake.", "A photo of {} standing on the beach.", "A photo of {} standing in a city street.", "A photo of {} standing on a mountain peak.", "An oil painting of {} standing on the beach.", "A watercolor illustration of {} standing on the beach.", "A charcoal sketch of {} standing on the beach.", "A digital painting of {} standing on the beach.", "A pencil drawing of {} standing on the beach."

their ability to retrieve images containing the target regions. For VFA and ICPE, which are not attribution methods, we modified only the training framework (not the architecture) to better align with our task.

In terms of baselines' performance, AbC and AbU are image-level attribution methods, so their results can be distracted by irrelevant regions in the image rather than the target region of interest. For ICPE and VFA, the main distinction lies in their training objectives and the underlying pretrained models. ICPE and VFA aim to maximize object detection likelihood, while ARD– is trained for region-level attribution, optimizing the attribution score. Furthermore, ARD– builds on countGD, pretrained to align both visual and textual information, while pretrained models of ICPE and VFA are designed specifically for object detection without cross-modal alignment. Another reason behind the failure of few-shot detectors in our task is the 1-shot fine-tuning setting, in which ICPE and VFA must learn a new concept from only one exemplar. To mitigate overfitting, these methods incorporate mechanisms that aim to learn a generic distribution of the concept. However, these mechanisms result in reduced concept fidelity when retrieving the most attributed regions.

E. Deployability and Run-time discussion

On an NVIDIA RTX 2080Ti, our method (ARD) performs region-level attribution for 5 images per second. This can be further accelerated via parallel processing, faster hardware, or network quantization. Additionally, it can be combined with faster methods like AbC to pre-filter unrelated images. In terms of applicability with respect to data access limitations, our method operates under the same assumption as

Figure 13. **Qualitative comparison between our method and AbC - an image-level data attribution method.** Figure shows the top-10 retrieved images from **ARD** and **AbC**. *Upper half*: numbers represent the attribution scores of regions detected by **ARD** for the generated region of the *"bench"*. *Bottom half*: numbers indicate **AbC**'s attribution scores for retrieved images with respect to the generated image.

most existing data attribution approaches. Specifically, it is intended for use by model owners who have access to the training data. These users are typically those interested in analyzing the influence of individual training samples on model behavior. In such settings, full access to the training data is both natural and entirely feasible.

F. Domain-gap discussion

A major concern is the potential domain gap between images generated by a text-to-image model before and after applying a customization technique, which directly affects our method's practical applicability. While this is a valid concern, we believe that the gap is minimal. Text-to-image models are trained on large-scale datasets, so customizing with a few images is unlikely to cause significant distributional shifts. Moreover, the customization process targets subject identity rather than altering overall content or style. To validate this, we conducted 2 cases of experiments: ***Case 1*****:** we used a given prompt and original model to generate an image, at the same time we customized the model using a real image (this image must include a concept of the same class as the object generated by original model), and then synthesized a second image using a similar prompt; ***Case 2*****:** we generated an image using the original model, customized the model using that image, and then generated a second image. Illustrations of our experiments are shown in Fig. 14, while the first experiment returned the second image closer to the images in our methods (as we also used real images to customize text-to-image model in our framework), the difference between concepts of two images makes our first comparison unfair in terms of objects (concepts). Hence, we introduce the second experiment to resolve this misalignment.

In both cases, we compared the two using CLIP and DINO features; CLIP measures semantic similarity aligned with language descriptions, while DINO captures visual similarity in terms of objects, structure, and style. Across 100 pairs of ***Case 2***, the average CLIP and DINO similarities were 0.95 and 0.88, respectively. We also conducted the same assessment for ***Case 1*** and obtained the average CLIP and DINO similarities of 0.93 and 0.82, respectively. The results indicate strong semantic and visual consistency before and after customization.

G. Limitations

The capability of our attribution detector, ARD, is currently limited to measuring the attribution of objects only. Further research on the attribution of arbitrary or structured (fixed-shape) regions could enhance a wide range of region-level data attribution applications. Moreover, the performance of ARD heavily depends on the pretrained model CountGD and the quality of the dataset synthesized from a customized text-to-image generative model. Therefore, in practice, it is important to improve the quality of the pretrained model and customization techniques, as well as to expand the dataset size, in order to build an effective attribution detector for specific tasks.

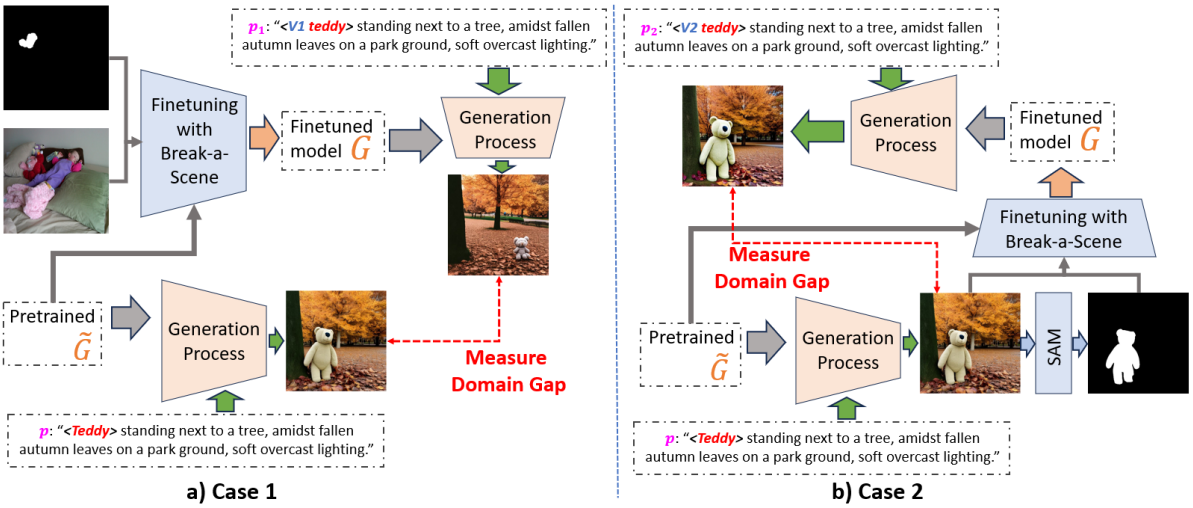


Figure 14. **Experiments to answer the concern about domain gap between images generated by a text-to-image model before and after applying a customization technique**.

Table 5. **Text templates for each group in LVIS-type dataset.**

Group name	**Text template**
living_creatures	"A {} is resting under the shade of a large tree.", "A {} is wandering through a dense forest.", "A {} is captured in a stunning wildlife photograph.", "A {} is lying on soft grass in a countryside setting.", "A {} is silhouetted against a golden sunset.", "An oil painting of a {} exploring a misty valley.", "A charcoal sketch of a {} sitting on a rock.", "An ink drawing of a {} in a traditional setting.", "A photorealistic rendering of a {} in a peaceful landscape.", "A vintage poster featuring a {} in a classic nature scene."
wearables	"A {} is hanging on a coat rack in a dimly lit hallway.", "A {} is displayed in a luxury boutique window.", "A {} is placed neatly on a wooden table in a tailor shop.", "A {} is resting on a chair beside a fireplace.", "A {} is folded inside a suitcase ready for travel.", "An oil painting of a {} on a vintage dresser.", "A charcoal sketch of a {} lying on a wooden table.", "An ink drawing of a {} in a 19th-century wardrobe.", "A photorealistic rendering of a {} in a stylish setting.", "A vintage poster featuring a {} in a retro fashion advertisement."
furniture_decor	"A {} is placed near a grand fireplace in a rustic home.", "A {} is positioned under a large crystal chandelier.", "A {} is standing alone in a vast empty hall.", "A {} is resting in the middle of a traditional Japanese tea room.", "A {} is located in a sunlit corner of a cozy reading room.", "An oil painting of a {} in a cozy library.", "A charcoal sketch of a {} in an antique furniture store.", "An ink drawing of a {} in a classical home interior.", "A photorealistic rendering of a {} in a modern home.", "A vintage poster featuring a {} in an old-style decor catalog."
tools_objects	"A {} is resting on a sturdy workbench.", "A {} is covered in rust and age marks.", "A {} is lying on a construction site floor.", "A {} is positioned on an artisan crafting table.", "A {} is stored inside a wooden toolbox.", "An oil painting of a {} in an old tool shed.", "A charcoal sketch of a {} in a mechanic workspace.", "An ink drawing of a {} in a vintage repair shop.", "A photorealistic rendering of a {} in industrial lighting.", "A vintage poster featuring a {} in a historical craftsmanship ad."
transportation	"A {} is parked near a quiet countryside road.", "A {} is cruising along a scenic coastal highway.", "A {} is covered in morning dew on a foggy street.", "A {} is silhouetted against a sunrise on a mountain road.", "A {} is parked beside a train station with steam rising.", "An oil painting of a {} in a bustling city street.", "A charcoal sketch of a {} driving through a foggy night.", "An ink drawing of a {} in a historical cityscape.", "A photorealistic rendering of a {} speeding on a freeway.", "A vintage poster featuring a {} in a retro advertisement."
tech_electronics	"A {} is placed on a marble table in a futuristic lab.", "A {} is covered in dust inside an old electronics shop.", "A {} is sitting on a work desk cluttered with notes.", "A {} is resting on a wooden shelf beside books.", "A {} is stored safely inside a padded carrying case.", "An oil painting of a {} surrounded by mechanical parts.", "A charcoal sketch of a {} with classic detailing.", "An ink drawing of a {} in a cyberpunk environment.", "A photorealistic rendering of a {} with neon reflections.", "A vintage poster featuring a {} in a retro advertisement."
food_drink	"A {} is placed on a rustic wooden cutting board.", "A {} is sitting next to a steaming cup of herbal tea.", "A {} is served on a silver plate in an elegant setting.", "A {} is resting in a woven basket at a farmers' market.", "A {} is stacked in a classic still-life arrangement.", "An oil painting of a {} with soft pastel colors.", "A charcoal sketch of a {} in a rustic kitchen.", "An ink drawing of a {} on a decorative plate.", "A photorealistic rendering of a {} with fresh ingredients.", "A vintage poster featuring a {} in a food advertisement."

Continued on next page

Group name	**Text template**
outdoor_nature	“A {} is standing tall against a misty mountain backdrop.”, “A {} is partially covered in snow during winter.”, “A {} is growing near a quiet lake under the sun.”, “A {} is surrounded by lush greenery in a rainforest.”, “A {} is positioned along a rocky path in a canyon.”, “An oil painting of a {} under a golden sunset.”, “A charcoal sketch of a {} in a serene landscape.”, “An ink drawing of a {} in a botanical illustration.”, “A photorealistic rendering of a {} in a vibrant ecosystem.”, “A vintage poster featuring a {} as part of a travel campaign.”
sports_play	“A {} is resting on a freshly cut grassy field.”, “A {} is placed next to a pair of running shoes.”, “A {} is stored in a well-used gym bag.”, “A {} is surrounded by cheering crowds in a stadium.”, “A {} is hanging from a rack in a sports store.”, “An oil painting of a {} in dynamic motion.”, “A charcoal sketch of a {} in a championship match.”, “An ink drawing of a {} in a dramatic action pose.”, “A photorealistic rendering of a {} in a high-energy game.”, “A vintage poster featuring a {} promoting a sports event.”
misc_objects	“A {} is lying on an antique wooden desk.”, “A {} is placed on a bookshelf among dusty volumes.”, “A {} is glowing softly under warm candlelight.”, “A {} is stored carefully in an elegant glass case.”, “A {} is positioned near a rain-covered window.”, “An oil painting of a {} in a cozy reading nook.”, “A charcoal sketch of a {} on an artist table.”, “An ink drawing of a {} in a historical archive.”, “A photorealistic rendering of a {} in a soft-lit room.”, “A vintage poster featuring a {} in an old advertisement.”

Table 6. **Base classes in each group of LVIS-type dataset.**

Group name	**Base class**
living_creatures	baboon, bat_(animal), bear, beetle, bird, cow, black_sheep, bull, bulldog, butterfly, calf, camel, cat, chicken_(animal), cock, cockroach, cougar, crab_(animal), crow, dalmatian, deer, dog, dolphin, domestic_ass, dove, dragonfly, duck, cub_(animal), duckling, eagle, eel, elephant, elk, falcon, ferret, fish, flamingo, foal, frog, gazelle, giant_panda, giraffe, goat, goldfish, goose, gorilla, grizzly, gull, hamster, heron, hippopotamus, hog, horse, hornet, hummingbird, kitten, koala, ladybug, lamb_(animal), lion, lizard, mallard, mammoth, manatee, monkey, octopus_(animal), ostrich, owl, parakeet, parrot, pelican, penguin, pigeon, puffer_(fish), puffin, pug-dog, puppy, rabbit, ram_(animal), rat, rhinoceros, rodent, seabird, seahorse, shark, sheep, shepherd_dog, snake, spider, crawfish, squirrel, starfish, tiger, turtle, vulture, walrus, wolf, zebra, polar_bear, pony, pet
wearables	apron, arctic_(type_of_shoe), armband, armor, bandanna, ballet_skirt, baseball_cap, baseball_glove, beanie, belt, belt_buckle, beret, bib, visor, blazer, blouse, bolo_tie, bonnet, boot, bow_(decorative_ribbons), bow-tie, bowler_hat, boxing_glove, suspenders, bracelet, brassiere, breech cloth, bridal_gown, bulletproof_vest, cap_(headwear), cape, cardigan, chain_mail, choker, cloak, coat, corset, costume, coverall, cowboy_hat, crown, diaper, dress, dress_hat, dress_suit, tux, underdrawers, leather, earplug, earring, eyepatch, fedora, flannel, flip-flop_(sandal), flipper_(footwear), fleece, gasmask, glove, goggles, halter_top, hat, helmet, headband, headscarf, jacket, jean, jersey, jewelry, jumpsuit, kilt, kimono, knee_pad, lab_coat, leggings_(clothing), life_jacket, mask, mitten, neckerchief, necklace, necktie, nightshirt, overalls_(clothing), pajamas, parka, polo_shirt, poncho, robe, scarf, shawl, shirt, shoe, short_pants, ski_parka, skirt, skullcap, sling_(bandage), slipper_(footwear), sock, sombrero, sportswear, suit_(clothing), sunhat, sweat_pants, sweatband, sweater, sweatshirt, swimsuit, tank_top_(clothing), tartan, tights_(clothing), trench_coat, trousers, turban, turtleneck_(clothing), underwear, vest, wristband, wristlet, wig, broach, cincture, hairnet, handkerchief, veil, lanyard, legging_(clothing), pantyhose, raincoat, sandal_(type_of_shoe), sunglasses, tiara, wedding_ring
furniture_decor	armoire, bath_mat, bath_towel, bathrobe, bathtub, bed, bedpan, bedspread, bench, billboard, birdbath, birdcage, birdhouse, blackboard, blanket, bookcase, bunk_bed, bulletin_board, cabinet, locker, chair, chaise_longue, chandelier, clock, clock_tower, clothes_hamper, coatrack, coffee_table, crib, curtain, cushion, cupboard, deck_chair, desk, dining_table, dresser, drawer, snowman, easel, electric_chair, fan, fireplace, folding_chair, footstool, futon, hamper, headboard, highchair, kitchen_table, lamp, lamppost, lampshade, lantern, loveseat, manger, mattress, mirror, ottoman, pew_(church_bench), pillow, playpen, recliner, rocking_chair, sofa, stool, table, table_lamp, tablecloth, tapestry, towel_rack, trunk, wardrobe, washbasin, beanbag, music_stool, sofa_bed, vase

Continued on next page

Group name	Base class
tools_objects	alarm_clock, ashtray, atomizer, award, ax, backpack, handbag, suitcase, Band_Aid, bandage, barrel, barrette, barrow, basket, battery, bead, beeper, bell, Bible, binder, binoculars, bob, bobbin, bobby_pin, bottle_opener, bread-bin, broom, bucket, business_card, button, calendar, can_opener, calculator, candle_holder, walking_cane, canister, canteen, bottle_cap, car_battery, cast, cash_register, checkbook, clip, clipboard, clippers_(for_plants), clutch_bag, coaster, coat_hanger, combination_lock, compass, cork_(bottle_plug), corkboard, corkscrew, crutch, doorknob, doormat, dropper, drill, dagger, dental_floss, cistern, cleansing_agent, frying_pan, sewing_machine, shower_head, shower_curtain, stirrup, oven, duct_tape, dustpan, earphone, eraser, file_(tool), fire_alarm, fire_extinguisher, fire_hose, first-aid_kit, flashlight, funnel, garbage, garbage_truck, garden_hose, grater, hammer, hairbrush, hair_dryer, handcart, handcuff, handle, handsaw, hinge, hook, igniter, iron_(for_clothing), ironing_board, kettle, key, keycard, knife, knitting_needle, knob, knocker_(on_a_door), ladder, ladle, latch, lawn_mower, lightbulb, lightning_rod, mallet, marker, measuring_cup, measuring_stick, microscope, motor, nailfile, needle, nutcracker, oar, padlock, paintbrush, palette, peeler_(tool_for_fruit_and_vegetables), pencil, pencil_box, pencil_sharpener, pendulum, pitchfork, pliers, plow_(farm_equipment), pocketknife, poker_(fire_stirring_tool), power_shovel, puncher, razorblade, reamer_(juicer), rolling_pin, rubber_band, safety_pin, scissors, scraper, screwdriver, scrubbing_brush, shaver_(electric), sharpener, shredder_(for_paper), shovel, skewer, spatula, spear, stapler_(stapling_machine), steering_wheel, stepladder, step_stool, stirrer, strainer, tape_measure, thermometer, thimble, thread, thumbtack, tongs, toolbox, toothbrush, toothpaste, toothpick, vacuum_cleaner, watering_can, wrench, cigarette_case, clothespin, clasp, coil, colander, crowbar, cufflink, detergent, faucet, file_cabinet, freshener, fume_hood, griddle, grill, hookah, hose, inkpad, mixer_(kitchen_tool), nosebag_(for_animals), noseband_(for_animals), oil_lamp, parking_meter, pegboard, pepper_mill, radiator, scale_(measuring_instrument), scoreboard, shaker, shampoo, Sharpie, shaving_cream, shears, shield, shot_glass, strap, straw_(for_drinking), vent, walking_stick, automatic_washer, wok
transportation	ambulance, baby_buggy, barge, bicycle, blimp, boat, bulldozer, bullet_train, bus_(vehicle), cab_(taxi), camper_(vehicle), canoe, car_(automobile), railcar_(part_of_a_train), elevator_car, cargo_ship, horse_carriage, cart, convertible_(automobile), cruise_ship, police_cruiser, dinghy, fighter_jet, ferry, freight_car, golfcart, gondola_(boat), helicopter, hot-air_balloon, houseboat, jeep, jet_plane, kayak, limousine, minivan, motor_scooter, motor_vehicle, motorcycle, passenger_car_(part_of_a_train), passenger_ship, pickup_truck, race_car, raft, school_bus, seaplane, snowmobile, space_shuttle, stagecoach, tow_truck, tractor_(farm_equipment), trailer_truck, train_(railroad_vehicle), tricycle, truck, unicycle, wagon, yacht, horse_buggy, cabin_car, river_boat, army_tank, wagon_wheel, dirt_bike
tech_electronics	monitor_(computer_equipment), remote_control, air_conditioner, amplifier, antenna, blender, blinker, camera, camera_lens, camcorder, CD_player, cellular_telephone, computer_keyboard, drone, dispenser, generator, iPod, laptop_computer, microphone, speaker_(stereo_equipment), mouse_(computer_equipment), mousepad, printer, projector, radio_receiver, radar, record_player, router_(computer_equipment), spotlight, stereo_(sound_system), subwoofer, tachometer, telephone, telephone_booth, telephoto_lens, television_camera, television_set, timer, typewriter, vending_machine, webcam, monitor_(computer_equipment) computer_monitor, traffic_light, videotape

Continued on next page

Group name	Base class
food_drink	almond, apple, applesauce, apricot, artichoke, asparagus, avocado, bagel, baguet, banana, batter_(food), bean_curd, beef_(food), beer_bottle, beer_can, bell_pepper, birthday_cake, blackberry, blueberry, boiled_egg, bread, broccoli, brownie, brussels_sprouts, bubble_gum, bun, burrito, butter, cake, candy_bar, candy_cane, cantaloup, cappuccino, carrot, casserole, cherry, chickpea, chili_(vegetable), chocolate_bar, chocolate_cake, chocolate_milk, chocolate_mousse, cider, cigar_box, cigarette, coconut, cocoa_(beverage), coffee_maker, coffeepot, coleslaw, cornbread, cornmeal, crabmeat, cracker, crescent_roll, cucumber, cup, trophy_cup, cupcake, date_(fruit), doughnut, edible_corn, eclair, egg, egg_roll, egg_yolk, eggplant, fig_(fruit), fish_(food), French_toast, fruit_juice, fudge, garlic, ginger, gourd, grape, green_bean, green_onion, grits, ham, hamburger, honey, hot_sauce, hummus, icecream, popsicle, jam, kiwi_fruit, lasagna, lemon, lemonade, lettuce, lime, liquor, lollipop, mandarin_orange, mashed_potato, meatball, melon, milk, milkshake, mint_candy, muffin, mug, mushroom, nut, octopus_(food), omelet, onion, orange_(fruit), orange_juice, pancake, papaya, pastry, patty_(food), pea_(food), peach, peanut_butter, pear, pickle, pie, pizza, pita_(bread), plate, platter, pop_(soda), pork_rib, potato, pretzel, pudding, quesadilla, quiche, raspberry, root_beer, salad, salad_plate, salami, salmon_(food), salsa, sausage, sherbert, smoothie, soup, soup_bowl, soupspoon, sour_cream, soya_milk, squid_(food), steak_(food), stew, strawberry, string_cheese, sugar_bowl, sushi, sweet_potato, Tabasco_sauce, taco, tequila, toast_(food), tomato, tortilla, truffle_(chocolate), turnip, vinegar, waffle, water_bottle, watermelon, wedding_cake, whipped_cream, wine_bottle, yogurt, zucchini, cauliflower, cayenne_(spice), celery, clementine, crisp_(potato_chip), cookie, escargot, gelatin, jelly_bean, lamb-chop, legume, lip_balm, olive_oil, pepper, persimmon, pineapple, prune, prawn, pumpkin, rib_(food), salmon_(fish), saltshaker, sandwich, saucepan, saucer, steak_knife, vodka, waffle_iron
outdoor_nature	aquarium, awning, bamboo, bait, birdfeeder, bouquet, buoy, cabana, candle, carnation, cleat_(for_securing_rope), cornice, flower_arrangement, gravy_boat, gravestone, log, nest, parasol, plume, saddle_(on_an_animal), saddle_blanket, sunflower, sugarcane_(plant), weathervane, window_box_(for_plants), seashell, blinder_(for_horses)
sports_play	basketball_backboard, ball, balloon, baseball, baseball_bat, basketball, barbell, bass_horn, beachball, gameboard, bow_(weapon), bowling_ball, chessboard, checkerboard, chopping_board, chopstick, poker_chip, diving_board, football_(American), football_helmet, frisbee, golf_club, hockey_stick, home_plate_(baseball), mound_(baseball), paddle, parasail_(sports), ping-pong_ball, pole, pool_table, racket, roller_skate, Rollerblade, sled, ski, ski_boot, ski_pole, skateboard, snowboard, soccer_ball, softball, surfboard, table-tennis_table, tennis_ball, tennis_racket, trampoline, volleyball, water_ski, slide, triangle_(musical_instrument), tambourine

Continued on next page

Group name	Base class
misc_objects	turkey_(food), trash_can, banner, birthday_card, book, booklet, bookmark, brass_plaque, bullhorn, deadbolt, bolt, box, briefcase, identity_card, card, carton, cassette, chalice, chime, chinaware, coin, coloring_material, condiment, cone, control, crossbar, crayon, cream_pitcher, crumb, cube, cylinder, cymbal, die, dish, dish_antenna, dishrag, dishtowel, dishwasher, dishwasher_detergent, dixie_cup, dog_collar, doll, dollar, dollhouse, diary, clarinet, musical_instrument, satchel, shoulder_bag, sleeping_bag, violin, duffel_bag, dumbbell, dumpster, envelope, figurine, flag, flagpole, flash, glass_(drink_container), globe, gift_wrap, hairpin, hand_glass, hand_towel, hardback_book, harmonium, hatbox, heart, heater, hourglass, ice_maker, ice_pack, ice_skate, inhaler, jar, jewel, joystick, keg, kennel, kitchen_sink, kite, Lego, license_plate, life_buoy, machine_gun, magazine, magnet, mail_slot, mailbox_(at_home), manhole, map, martini, mascot, masher, matchbox, milestone, milk_can, money, napkin, newspaper, newsstand, notebook, notepad, packet, pad, paper_plate, paper_towel, paperback_book, paperweight, parachute, passport, pennant, penny_(coin), perfume, piggy_bank, pistol, plastic_bag, pocket_watch, postbox_(public), postcard, poster, propeller, pouch, projectile_(weapon), radish, receipt, rearview_mirror, reflector, ring, road_map, runner_(carpet), shopping_bag, shopping_cart, shower_cap, signboard, silo, sink, speaker_(stero_equipment), spice_rack, sparkler_(fireworks), spectacles, spoon, tobacco_pipe, wooden_leg, stop_sign, brake_light, street_sign, streetlight, stylus, syringe, tag, taillight, tank_(storage_vessel), tape_(sticky_cloth_or_paper), tarp, tassel, tea_bag, teacup, teakettle, teapot, teddy_bear, thermos_bottle, thermostat, tinfoil, tinsel, tissue_paper, toaster, toaster_oven, toilet, toilet_tissue, tray, umbrella, urinal, urn, wallet, wall_clock, wall_socket, watch, water_cooler, water_faucet, water_heater, water_jug, water_gun, water_scooter, water_tower, wet_suit, wheel, wheelchair, whistle, wind_chime, windmill, windshield_wiper, windsock, wine_bucket, wineglass, wooden_spoon, wreath, bagpipe, banjo, baseball_base, pirate_flag, bottle, bowl, pipe_bowl, can, tote_bag, chap, pacifier, comic_book, cooker, cooking_utensil, cooler_(for_food), cornet, cowbell, crape, crate, crock_pot, crouton, crucifix, hair_curler, curling_iron, Dixie_cup, drum_(musical_instrument), drumstick, eggbeater, refrigerator, Ferris_wheel, fire_engine, fireplug, fishbowl, fishing_rod, flap, flute_glass, food_processor, fork, forklift, gag, gargle, gargoyle, gemstone, grocery_bag, guitar, gun, hammock, headlight, headset, headstall_(for_horses), hotplate, mast, mat_(gym_equipment), medicine, microwave_oven, painting, pan_(for_cooking), pan_(metal_container), parchment, pen, person, phonograph_record, piano, pin_(non_jewelry), pinecone, pinwheel, pipe, pitcher_(vessel_for_liquid), place_mat, pot, flowerpot, potholder, pottery, puppet, quilt, rag_doll, rifle, saddlebag, sail, sawhorse, saxophone, scarecrow, sculpture, soap, solar_array, sponge, statue_(sculpture), stove, mop, sword, telephone_pole, cover, towel, toy, tripod, vat, yoke_(animal_equipment)