

# What Changed? Detecting and Evaluating Instruction-Guided Image Edits with Multimodal Large Language Models

## Supplementary Material

### A. Difference Detection Training Dataset

**Stage 1: Additional Details.** To effectively train our difference detection model, we build our first stage on LVIS [22], which provides a broad range of annotated objects. Given that LVIS encompasses 1,723 distinct categories, inconsistencies may arise during annotation, whereby annotators might label the same or similar objects under different categories or overlook certain objects. To address our objective of predicting missing objects in each pair, we filter them using the open-vocabulary detector OWL-ViT [36]. OWL-ViT is then tasked with verifying whether objects annotated in the first image (but not in the second) are genuinely absent in the second image and vice versa, thereby preventing erroneous ground-truth labels.

Moreover, considering the density of annotations in LVIS, we further refine our collection by eliminating annotations that are overly small, specifically, those with at least one dimension measuring less than 16 pixels. The final dataset consists of 118k images with a total of 795k, 725k, and 94k ADD, REMOVE, and EDIT operations respectively.

**Stage 2: Additional Details.** Building upon the previous stage, the second stage dataset is designed to refine our model for detecting differences in edited images. The dataset is constructed using LVIS annotations, applying specific filtering criteria to ensure data quality. In detail, we sample a maximum of 4 objects per image. For each selected object, we consider its segmentation mask and filter out records that cover less than 3% of the image area, while the maximum allowed overlap between selected object masks is 5%. As for the inpainting generation pipeline, the Kandinsky 2.2 inpaint model [42] is employed with 100 diffusion steps and a guidance scale of 4.

The final dataset comprises a total of 97k images for training and 19k images for testing. Within the training set, each of the three operations (ADD, REMOVE, and EDIT) is represented by approximately 33k instances to avoid unbalanced predictions. Additionally, 19k images remain unchanged. Similarly, in the test set, there are around 7k instances for each of the three operations, along with 5k images that remain unaltered. Finally, samples from both datasets used in stage 1 and stage 2 are provided in Fig. 4.

### B. Training Details

In each stage, the training of all the models is conducted within the same experimental setting. For instance, we apply QLoRA with rank 64, scaling factor 16, and a dropout

rate of 0.1. Both difference detection and coherence estimation fine-tuning employ 4-bit quantization and the paged AdamW 8-bit optimizer [34].

For image encoding, images are always center-cropped based on the smaller dimension to maintain a square aspect ratio. When employing Idefics3, images are internally resized to 1,456 pixels and encoded in a list of 364-pixel crops, generating a grid of 16 crops separately encoded. For Qwen2-VL, the image is encoded using a bidimensional positional embedding. Image tokens are then compressed by an MLP layer that encodes  $2 \times 2$  adjacent tokens into a single one. The input image resolution is kept at 1,456. For mPLUG-Owl3, the images are directly encoded through a SigLIP400m-384 [53].

**Difference Detection.** All stages of the difference detection training use 8 NVIDIA A100 64GB GPUs. Both training stages are conducted with a learning rate of  $1 \times 10^{-4}$  and a total batch size of 8. In the first training stage, Idefics converges after 9k training steps, with evaluations performed every 1k steps. Differently, in the second training stage, convergence is reached after 30k training steps, with evaluations conducted every 5k steps. No image preprocessing is applied in the first stage. During the second stage, however, we observe that the model may be susceptible to artifacts introduced by the inpainting model, which can compromise its generalizability across different editing outputs. To remove the effect of these low-level imprints, during training, we apply random JPEG compression between 15% and 50% magnitude, following an established practice in deep-fake detection [50]. At inference time, JPEG compression is set at 33%.

**Coherence Estimation.** Training is performed on a single NVIDIA A100 64GB GPU with a batch size of one and a learning rate of  $1 \times 10^{-5}$ . In this case, Idefics reaches convergence at 550 training steps, with evaluations every 50 steps.

### C. Evaluation Details

**Difference Detection with Open-Vocabulary Detectors.** Standard object detection models like OWLv2 [37] and Grounding DINO [33] must be adapted to perform difference detection, as they operate on single images independently. To enable this, we first apply object detection separately to the original and edited images using all the LVIS classes as labels. The resulting detections are then compared to identify differences.

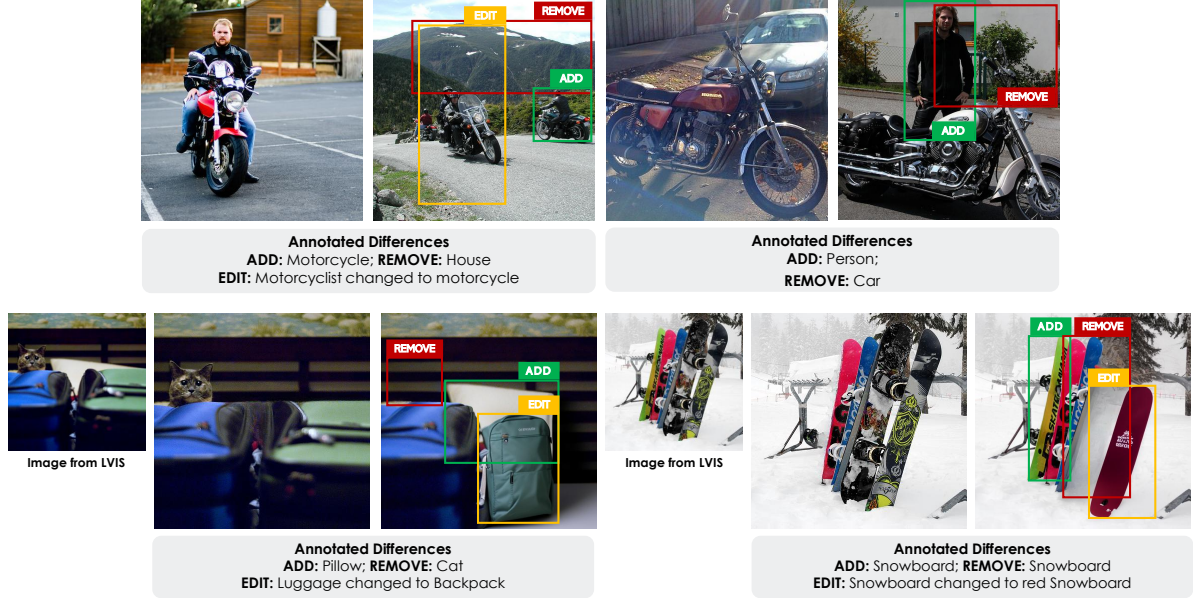


Figure 4. Qualitative examples from the difference detection training datasets: the first row shows samples from the dataset used in stage 1, and the second row from the dataset employed in stage 2.

Given a detection in the original image: if a detection in the edited image has an IoU greater than 0.5 and a different label, it is classified as a **EDIT** difference. If no detection in the edited image exceeds the IoU threshold, it is considered a **REMOVE** difference. Conversely, if a detection in the edited image has no corresponding detection in the original image above the threshold, it is labeled as an **ADD** difference. Finally, if two detections with the same label have an IoU over the threshold, then no difference is found.

**Difference detection with GLaMM.** The GLaMM model [41] is employed in a zero-shot setting as a standard single-image object detector, and the algorithm described previously is used for difference extraction in this case too.

## D. Ablation Study on Localization

In this section, we build on the experiments in Tab 4 and further motivate the relevance of the localized output of DICE. Indeed, localization allows the evaluator model to focus on relevant regions and enhances interpretability. To prove this, we employ Gemma3-27B [48] and Qwen2.5-VL-32B [2] in a fully zero-shot setting, measuring correlation with human ratings with and without DICE detections in the prompt. For Prompt Adherence, we provide bounding boxes of differences labeled as “coherent”; for Background Preservation, we use boxes predicted as “non-coherent”. These boxes are overlaid on the image, and their corresponding textual descriptions are appended to the prompt, following the strategy of our coherence evaluator. As shown in Tab. 5, adding localization significantly boosts correlation with human evaluations. This confirms that ground-

	Gemma3-27B			Qwen2.5-VL-32B		
	$\rho$	$\rho_s$	$\tau$	$\rho$	$\rho_s$	$\tau$
<i>Background Preservation</i>						
w/o DICE detections	13.9	12.2	10.5	20.1	20.3	17.5
<b>w/ DICE detections</b>	<b>36.2</b>	<b>36.3</b>	<b>32.8</b>	<b>37.3</b>	<b>35.9</b>	<b>31.5</b>
<i>Prompt Adherence</i>						
w/o DICE detections	31.0	27.0	23.0	67.7	70.2	<b>59.8</b>
<b>w/ DICE detections</b>	<b>32.8</b>	<b>34.3</b>	<b>30.3</b>	<b>67.9</b>	<b>70.6</b>	59.5

Table 5. Impact of localization on human correlations.


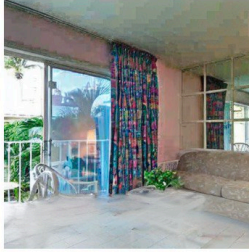
ing detected objects enables MLLMs to better interpret and reason about visual edits. Notably, the substantial gains in Background Preservation suggest that MLLMs evaluators may be biased toward verifying whether the instructed change occurred, while neglecting unintended modifications. Our method explicitly identifies such unintended changes during the detection phase, leading to more robust evaluations.

## E. User Study

To evaluate the performance of instruction-based image editing models, we conducted a user study in which participants assessed edited images based on prompt adherence and background preservation.

**Data Generation.** The editing models used for user study data generation are set according to their default configuration. In particular, for InstructDiffusion [18] the image guidance scale is set to 1.25 and the text guidance scale is set to 5, while for all the other models we use 1.5 and 7 respectively. For HIVE [55] and InstructDiffusion the number of inference steps is 100, while for all the other models

**Prompt**  
delete the table

**Original Image**  **Edited Image** 

**Votes**

Rate prompt following  
1 - Not Applied At All 5 - Perfectly Applied

Rate background preservation  
1 - Changed Completely 5 - Preserved Completely

Figure 5. User study interface displaying the original and the edited image alongside the editing prompt.

we set the inference steps to 20.

Participants rated the degree to which the requested edit was applied on a scale from 1 to 5: *not applied at all*, *slightly applied*, *moderately applied*, *well applied*, and *perfectly applied*. Similarly, they evaluated the preservation of background elements using another scale from 1 to 5: *changed completely*, *moderately altered*, *mostly preserved*, *nearly fully preserved*, and *preserved completely*. The study was conducted through an interactive interface that allowed participants to compare the original and the edited image while reading the prompt. Fig. 5 illustrates the interface used in the study.

## F. Additional Qualitative Results

Fig. 6 illustrates a wide range of successful edits handled by DICE across diverse scenarios, featuring highly precise bounding boxes around the modified elements. In several examples, the system accurately detects color changes (*e.g.*, altering the color of a bird or an umbrella) and clearly distinguishes added objects, such as a watermelon or a necktie. The pipeline also robustly identifies removed elements, whether it is a coffee table or a giraffe, and captures more subtle edit operations, for instance replacing a bird with a boy or a cat with a dog. These qualitative results highlight how DICE maintains spatial accuracy and class awareness when performing additions, removals, and edits, thereby demonstrating its adaptability to different editing requests.

Additionally, the coherence evaluator provides clear, step-by-step reasoning for each detected change, explaining why it is coherent or not. For example, as shown in Fig. 7, in the “change the color of the rose to blue” prompt, the

system points out that “cyan” is still a valid shade of blue and correctly labels the edit as coherent. Likewise, when asked to “add a basketball to the top of the car”, it confirms not only that the basketball has been introduced but also that it is located precisely where the prompt requires – on top of the car. These reasoned explanations highlight the transparency of our pipeline, helping users understand which aspects of the prompt were fulfilled and why certain edits might fall short of the requested modifications.

## G. Limitations

DICE can occasionally exhibit inaccuracies in detected differences, as shown in Fig. 8, that can end up affecting the coherence estimation. For instance, in the first example, changing the left hat to white actually matches the prompt. However, the pipeline labels the detected edit as “fedora”, which confuses the coherence evaluator into classifying it as non-coherent. A similar problem arises in the second image, where the correct recoloring of the dog to brown is labeled as “beige puppy”, causing confusion – especially because the new color of the dog blends into the background.

Another failure can occur when the coherence evaluator is overly strict. In the “replace dog with watermelon” example, the editing model does replace the dog with a watermelon, yet the coherence evaluator rejects the result because it expects a total transformation. Similarly, in “replace boy with girl,” the difference detector notes that a person has been edited but fails to recognize the new person as female, and the coherence evaluator does not correct this oversight, causing it to label the modification as incoherent even though it actually meets the prompt requirements.

Further ambiguities in the prompt can lead to inaccurate predictions. Indeed, sometimes, the prompt does not specify all the elements needed to produce a univocal decision, especially when it is unclear whether the prompt specifies an addition or a substitution of elements in the scene.

---

**Custom System Prompt:**

You are a system that detects differences between two images.

- Extract the elements that are changed in the second image with respect to the first one.
- Create a new entry for each distinct change.
- For each entry, use the following format:

"<CHANGE\_COMMAND>: <CHANGED\_ELEMENT>, (<BOUNDING\_BOX>)"

CHANGE\_COMMAND:

- ADD: If a new element appears in the second image that was not present in the first.
- REMOVE: If an element from the first image is missing in the second.
- EDIT: If an element in the second image is different but in the same location as an element in the first image.

CHANGED\_ELEMENT: Describe the element that has changed.

BOUNDING\_BOX: Use normalized coordinates [x0, y0, x1, y1] for the changed element position in the second image, where (x0, y0) is the top-left corner, and (x1, y1) is the bottom-right corner. The coordinates should be scaled between 0 and 1, with 0 representing one edge of the image and 1 representing the opposite edge.

---

Table 6. Prompting template for the difference detection stage of the DICE pipeline.

---

**Custom System Prompt:**

You are evaluating if a specific change detected by an AI vision model matches the request in the original edit prompt.

**## Task**

Determine if the detected change, as described and bounded by the provided colored bbox, matches the request in the original edit prompt.

A match is valid only if the localized detected change is 100% compatible with the requested prompt.

Any unwanted modification of the original image (even small) should avoid a match.

**## Context**

-The original image and the edited image are provided, in this order. The edited image is the original with some changes applied. Focus only on the area specified by the bbox in the detected change.

-Another AI model has detected a change in the image, including its bbox.

-ADD: An object is only added in the edited image (on the background).

-EDIT: An object is substituted with another one in the edited image.

-REMOVE: An object is removed in the edited image.

-Be strict: An EDIT means that an object has been removed and substituted with another one, ensure nothing was removed unless explicitly stated in the prompt. If an object has been removed unexpectedly, then you should say NO.

**## Example Response**

-Reasoning: <REASONING>

-Decision: "YES" or "NO"

**User Prompt:****## Instructions**

1. The original edit prompt is: {SUBSTITUTE\_PROMPT}
  2. The detected change to evaluate is: {SUBSTITUTE\_CHANGE}
  3. Use only the text and the observations from the specified bbox area (colored) in both the original and edited images to decide if the specific detected change aligns with the original edit prompt.
- 

Table 7. Prompting template for the coherence estimation stage of the DICE pipeline.



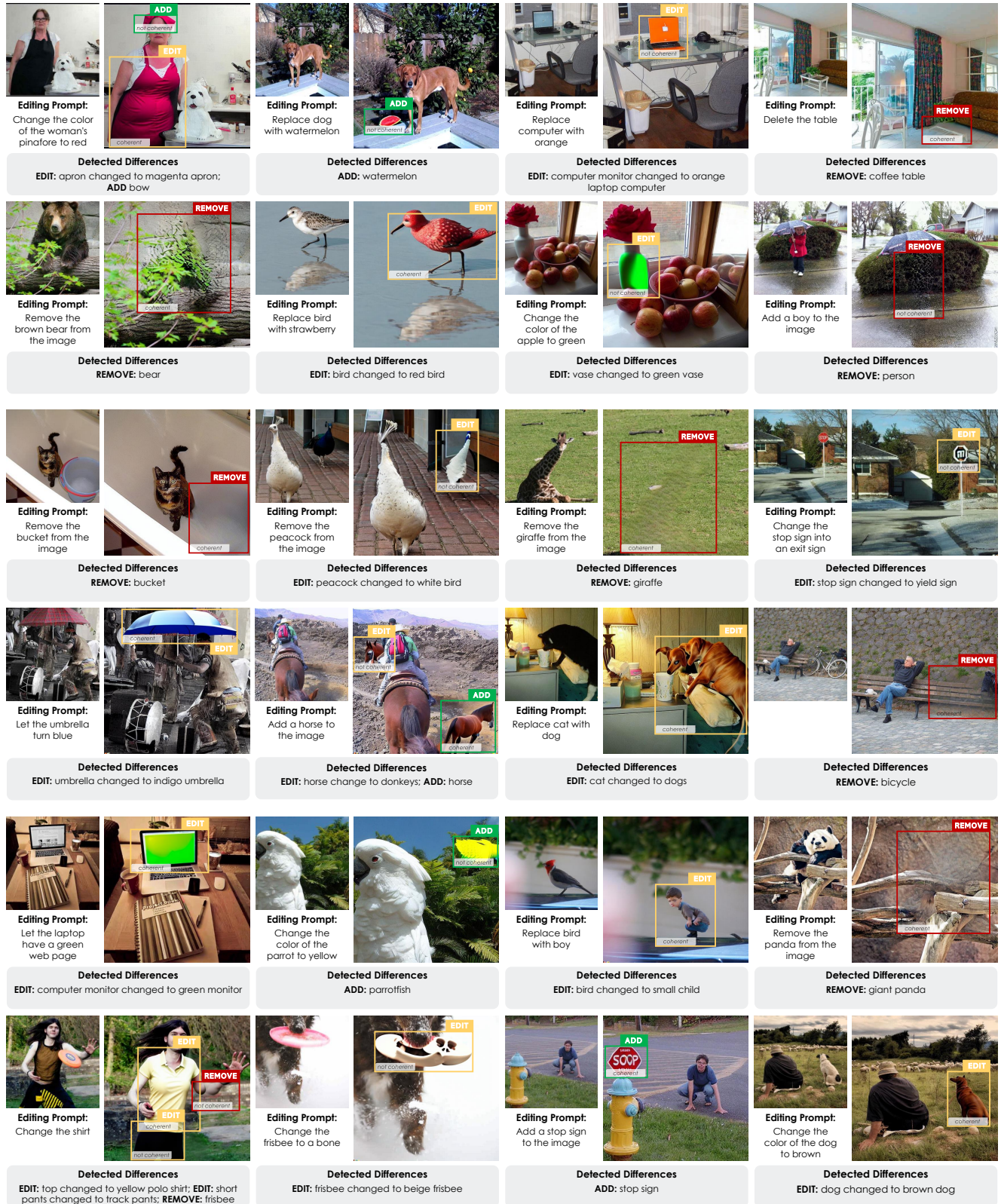


Figure 6. Additional qualitative results. Each instruction-based edit shows the original image (left) and the edited version (right), alongside the given prompt.





Figure 7. Examples illustrating the reasoning of the coherence evaluator that justifies its ‘YES’ or ‘NO’ decisions. Each box pairs a detected difference with an explanation of why the edit either fulfills or fails the user’s request, highlighting the ability of the pipeline to handle both spatial and semantic context.

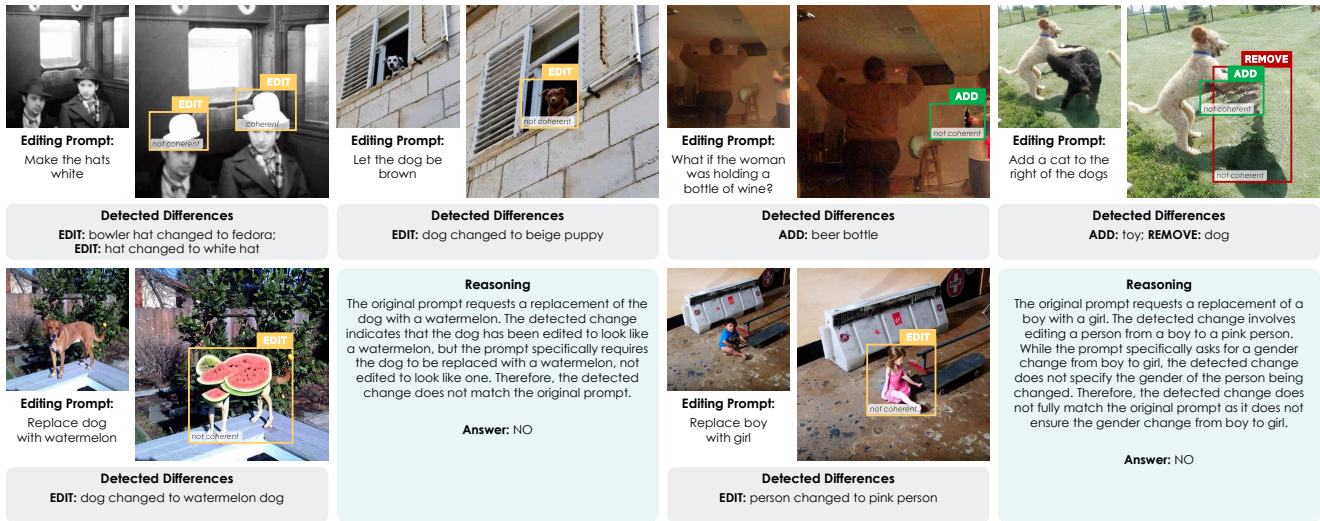


Figure 8. Failure cases where detection errors may impact coherence evaluation, potentially leading to misclassifications. Inaccurate identification of edits can introduce ambiguity for the coherence evaluator, while strict coherence criteria might occasionally reject valid changes, highlighting the interdependence of both stages.