Explaining Object Detectors via Collective Contribution of Pixels

Supplementary Material

A. Limitations and clarifications

Theoretical aspects of approximating accuracy in VX-CODE. Although the proposed method leverages Shapley values and interactions, theoretical analysis of the approximation accuracy is left for future work. However, we emphasize that the goal of this work is not to approximate Shapley values—**it is to effectively identify important regions by considering collective contribution.** Our method is inspired by and implements the idea of Shapley values and interactions in a computationally efficient manner, particularly via a greedy algorithm with a variant of the Shapley values and interaction, as shown in our formulation (Sec. 4.1). Consequently, our method works well in model explanation. We empirically evaluate the approximation accuracy by comparing it with Shapley values estimated via Monte Carlo sampling, and the results are presented in App. D.2.

Validity of reward function. To ensure a fair comparison, we use the same reward function Eq. (3) used in existing studies, including those based on Shapley values [27, 28, 40]. The design of an appropriate reward function is itself an open problem and is beyond the scope of this paper. Importantly, the proposed method is agnostic to the choice of reward function and can be applied as-is once an appropriate one is defined in future work.

Validity of evaluation metrics. In this study, we primarily evaluate the proposed method using the insertion and deletion metrics. These metrics have been long used in literature of visual explanation and feature selection, and they enable fair comparison across methods. Superior performance on these metrics is well aligned with providing convincing explanation of the models. Note that our method outperforms the baselines not only in these metrics, but also in another common metric, the pointing game (see App. D.4). Additionally, to demonstrate the effectiveness of the proposed method, we conduct extensive qualitative experiments. These include accurately identifying important regions outside the bounding box (Sec. 4 and App. D.7), introducing a weighted reward function and visualizing patches that contribute more to classification or localization (App. D.10), and analyzing object detectors using the proposed method (App. D.11). Such experiments have rarely been explored in the literature and represent one of the key contributions of this paper.

B. Visual explanations for classification

In the main paper, we omitted detailed discussions of related work on visual explanations for classification due to space constraints; here, we provide a more comprehensive review.

Various visual explanation methods have been proposed for image classification models [1, 7, 9, 15, 39, 44, 45, 47, 52, 58, 67]. Generally, these methods can be broadly classified into three categories: gradient-based, class-activation-based (CAM-based), and perturbation-based.

Gradient-based methods. These methods generate a saliency map by calculating the gradient of each pixel in the input image with respect to the model's output score, visualizing the important regions [3, 18, 48–51, 54, 62, 63]. For example, LRP [3] computes pixel-wise feature attributions by propagating relevance from the output layer to the input layer. CRP [18] extends LRP by emphasizing the relevance originating from the target class, calculating the disparity between the relevance from the target class and the average relevance from other classes.

CAM-based methods. These methods calculate the weights representing the importance of each channel in the feature maps, and visualize important regions by computing the weighted sum of these weights and the feature maps [7, 11, 15, 25, 45, 58, 67, 68]. CAM [68] incorporates global average pooling into the CNN and generates a heat map that highlights important regions. Grad-CAM [45] generalizes CAM, making it possible to incorporate it into any CNN. Grad-CAM++ [7] improves upon Grad-CAM by incorporating pixel-level weighting, thereby enhancing the quality of the heat map. Other methods that more accurately compute the weights representing the importance of each channel have been proposed [11, 15, 58, 67].

Perturbation-based methods. These methods identify important regions by introducing perturbations to the input image and quantifying the changes in the output [4, 9, 23, 34, 39, 44, 47, 52, 53]. LIME [5, 44] occludes by using super-pixel to compute the importance score. RISE [39] visualizes important regions by applying random masks to the input image and analyzing the output results generated from the masked images. SHAP [34] distributes confidence scores to contributions by utilizing Shapley values from game theory.

Methods that consider interactions [17] in addition to Shapley values have also been proposed [4, 52]. PredDIFF [4] is a method that measures changes in predictions by marginalizing features from a probabilistic perspective. This paper clarifies the relationship between PredDIFF and Shapley values, and introduces a new measurement method

that takes interactions into account. MoXI [52] utilizes a technique to approximate Shapley values and interactions to identify important regions for classification models. While MoXI considers only pairwise patch interactions and their Shapley values (equivalent to $r = 1$ in our method), our method considers interactions with arbitrary patches and their Shapley values. As described in Sec. 5.2, in explanations for object detectors, some cases fail to identify important patches when using $r = 1$. The proposed method addresses this limitation and provides more faithful explanations for object detectors. Additionally, we introduce π^*-index that clarifies the connection between its underlying concept and sequential coalitional games (see Sec. 4.3), and prove that this index satisfies the essential axioms of coalitional games (see App. C.9).

Methods such as SAG [47] and REX [9] have been proposed to generate multiple explanations for image classification models. These methods determine importance rankings for each patch or pixel from multiple perspectives. For example, SAG [47] divides an image into 7×7 patches and generates multiple explanations by identifying the minimal sufficient explanation using beam search. A similar approach to SAG and REX is the Visual Precision Search [8], which has been proposed for object foundation models to greedily identify important patches. These methods are essentially similar to the patch insertion in the proposed method. The key difference in the proposed method is that we highlight how these patch selections incorporate Shapley values and interactions and analyze object detectors from the perspective of collective contributions across multiple patches. Additionally, the proposed method incorporates not only patch insertion but also patch deletion.

Finally, we also mention logic-based explanations [10, 22, 35]. These methods aim to produce highly reliable explanations, but they treat the model as a white box and represent it using logical formulas. In contrast, the proposed method treats the model as a black box and does not rely on its internal structure.

C. Details of the proposed method

In this section, we provide details of the proposed method that are not included in the main paper.

C.1. Strategies for further reducing computational cost

When $r \geq 2$, the proposed method leads to an increase in the number of combinations ${}_{|N\setminus\mathcal{B}_{k-1}|}C_r$, making real-time execution challenging. To mitigate this, we introduce patch selection and step restriction. Specifically, patch selection reduces the number of patches involved in the combination process to below $|N \setminus \mathcal{B}_{k-1}|$, and step restriction confines the combination process to the early steps. While these methods reduce the total number of combinations and lower computational cost, they may lead to a decrease in accuracy. In Sec. 5.4, we show that these modules significantly reduce generation time with only a slight decrease in accuracy.

Algorithm 1 Process of patch insertion

Input: Reward function f,
index set N of patches,
the number of patches $r \in \{1, \dots, |N|\}$ inserted at each step,
patch selection parameter $m \in \{1, \dots, |N|\}$,
step restriction parameter $\gamma \in [0, 1]$.
Output: Sequence of subsets $\mathcal{B}_1, \dots, \mathcal{B}_E \subseteq N$

$\mathcal{B}_k \leftarrow \{\}$ for all $k = 0, \dots, E$
$T = N$, $k = 1$
while $|T| \neq 0$ **do**
  if $|T| < r$ **then**
    $r \leftarrow |T|$
  end if
  if $r \geq 2$ and $|\mathcal{B}_{k-1}| \leq \gamma |N|$ **then**
    `# Require` $|N \setminus \mathcal{B}_{k-1}|$ `forward process.`
    $P_k^m \leftarrow \underset{P \subseteq N\setminus\mathcal{B}_{k-1}, |P|=m}{\arg\max} \sum_{b\in P} f(\{\mathcal{B}_{k-1}\} \cup \{b\})$
    `# Require` ${}_mC_r$ `forward process.`
    $B_k \leftarrow \underset{B \subseteq P_k^m, |B|=r}{\arg\max} f(\{\mathcal{B}_{k-1}\} \cup B)$
    $\mathcal{B}_k \leftarrow \mathcal{B}_{k-1} \cup B_k$
    $T \leftarrow T \setminus B_k$
    $k \leftarrow k + 1$
  else
    `# Require` $|N \setminus \mathcal{B}_{k-1}|$ `forward process.`
    $b_k \leftarrow \underset{b \in N\setminus\mathcal{B}_{k-1}}{\arg\max} f(\{\mathcal{B}_{k-1}\} \cup \{b\})$
    $\mathcal{B}_k \leftarrow \mathcal{B}_{k-1} \cup \{b_k\}$
    $T \leftarrow T \setminus \{b_k\}$
    $k \leftarrow k + 1$
  end if
end while
return $\mathcal{B}_1, \dots, \mathcal{B}_E$

Patch selection. We limit the number of patches to be combined, assuming that combining each patch that has little impact on the reward is meaningless. Let m be the number of patches selected for combination, and P_k^m be the set of target patches to be combined at step k, where $|P_k^m| = m$. Then, P_k^m is determined as follows:

$$P_k^m = \underset{P \subseteq N\setminus\mathcal{B}_{k-1}}{\arg\max} \sum_{b\in P} g(b), \quad \text{s.t. } |P| = m,$$

Algorithm 2 Process of patch deletion

Input: Reward function f,
index set N of patches,
the number of patches $r \in \{1, \dots, |N|\}$ removed at each step,
patch selection parameter $m \in \{1, \dots, |N|\}$,
step restriction parameter $\gamma \in [0, 1]$.
Output: Sequence of subsets $\mathcal{B}_1, \dots, \mathcal{B}_E \subseteq N$

$\mathcal{B}_k \leftarrow \{\}$ for all $k = 0, \dots, E$
$T = N$, $k = 1$
while $|T| \neq 0$ **do**
  if $|T| < r$ **then**
    $r \leftarrow |T|$
  end if
  if $r \geq 2$ and $|\mathcal{B}_{k-1}| \leq \gamma |N|$ **then**
    # Require $|N \setminus \mathcal{B}_{k-1}|$ forward process.
    $P_k^m \leftarrow \underset{P \subseteq N \setminus \mathcal{B}_{k-1}, |P|=m}{\arg\max} \sum_{b \in P} -f(N \setminus (\{\mathcal{B}_{k-1}\} \cup \{b\}))$
    # Require ${}_mC_r$ forward process.
    $B_k \leftarrow \underset{B \subseteq P_k^m, |B|=r}{\arg\min} f(N \setminus (\{\mathcal{B}_{k-1}\} \cup B))$
    $\mathcal{B}_k \leftarrow \mathcal{B}_{k-1} \cup B_k$
    $T \leftarrow T \setminus B_k$
    $k \leftarrow k + 1$
  else
    # Require $|N \setminus \mathcal{B}_{k-1}|$ forward process.
    $b_k \leftarrow \underset{b \in N \setminus \mathcal{B}_{k-1}}{\arg\min} f(N \setminus (\{\mathcal{B}_{k-1}\} \cup \{b\}))$
    $\mathcal{B}_k \leftarrow \mathcal{B}_{k-1} \cup \{b_k\}$
    $T \leftarrow T \setminus \{b_k\}$
    $k \leftarrow k + 1$
  end if
end while
return $\mathcal{B}_1, \dots, \mathcal{B}_E$

where

$$g(b) = \begin{cases} f(\{\mathcal{B}_{k-1}\} \cup \{b\}) & \text{(insertion)} \\ -f\left(N \setminus (\{\mathcal{B}_{k-1}\} \cup \{b\})\right) & \text{(deletion)} \end{cases}.$$

Equation (C.1) selects the set of patches with the top m rewards. This module efficiently selects target patches to maximize the reward function through combination.

Step restriction. We limit the steps of the combination process by stopping once γn $(0 \leq \gamma \leq 1)$ patches have been identified, and the remaining patches are selected individually ($r = 1$). In our framework, the most important patches are typically identified in the initial steps (see App. 5.4). Therefore, restricting the combination process to these steps allows efficient identification of important patches.

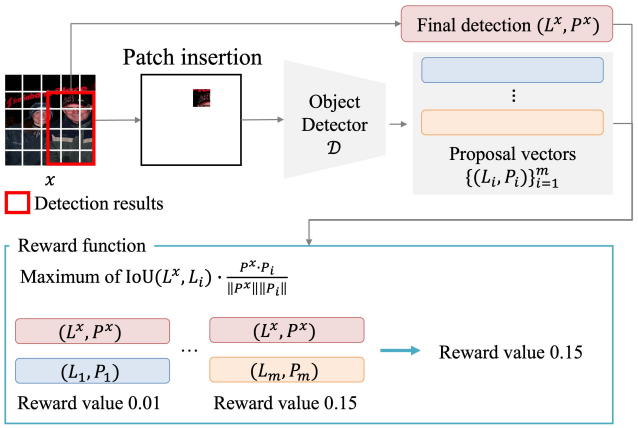


Figure H. Conceptual illustration of reward computation. Given a partially corrupted image, the detector outputs multiple proposals. The reward is computed by scoring each proposal against the target detection in terms of localization and classification agreement, and taking the maximum score.

These two methods reduce the computational cost from $\mathcal{O}(r^{-1}n^{r+1})$ to $\mathcal{O}(\frac{\gamma n}{r}m^r + (1-\gamma)n^2)$. We set $m = 30$ and $\gamma = 0.1$.

C.2. Details of the algorithm

We summarize the algorithm of VX-CODE. The processes of patch insertion and patch deletion are shown in Algorithm 1 and Algorithm 2, respectively. As shown in these algorithms, VX-CODE identifies r patches or a single patch at each step based on the value of the reward function, using a simple greedy algorithm.

C.3. Conceptual illustration of reward computation

Figure H illustrates the concept of the reward computation in Eq. (3). Given a partially corrupted image x_S, the detector outputs multiple proposals. For each proposal, the reward evaluates how well it matches the target detection (L^x, P^x) in terms of both localization and classification agreement. The final reward is defined as the maximum score over all proposals.

C.4. Specific example of patch insertion

We provide specific examples for cases with $r = 1$ and $r = 2$ in patch insertion described in Sec. 4.1.

C.4.1. Case with r=1

For step $k = 1$, from Eqs. (4.1) and (4.1), only the highest Shapley value $\phi(\{b_1\}|\{\{b_1\}\})$ becomes the optimal set $\mathcal{B}_1 = \{b_1\}$.

For $k \geq 2$, we find b_k that maximizes $f(\{\mathcal{B}_{k-1}\} \cup \{b_k\})$.

From Eqs. (4.1) and (4.1), this is formulated as follows:

$$
\begin{aligned}
b_k &= \underset{b\in N\setminus\mathcal{B}_{k-1}}{\arg\max}\ f(\{\mathcal{B}_{k-1}\}\cup\{b\}) - f(\emptyset) \\
&= \underset{b\in N\setminus\mathcal{B}_{k-1}}{\arg\max}\ \phi^{\mathrm{sc}}(\{\mathcal{B}_{k-1}\}\cup\{b\}) \\
&= \underset{b\in N\setminus\mathcal{B}_{k-1}}{\arg\max}\ I^{\mathrm{sc}}(\{\mathcal{B}_{k-1}\}\cup\{b\}) \\
&\qquad + \sum_{B'\in\mathcal{P}^*(\{\mathcal{B}_{k-1}\}\cup\{b\})} \phi^{\mathrm{sc}}(B') \\
&= \underset{b\in N\setminus\mathcal{B}_{k-1}}{\arg\max}\ I^{\mathrm{sc}}(\{\mathcal{B}_{k-1}\}\cup\{b\}) + \phi^{\mathrm{sc}}(\{b\}) + \phi^{\mathrm{sc}}(\mathcal{B}_{k-1}) \\
&= \underset{b\in N\setminus\mathcal{B}_{k-1}}{\arg\max}\ I^{\mathrm{sc}}(\{\mathcal{B}_{k-1}\}\cup\{b\}) + \phi^{\mathrm{sc}}(\{b\}).
\end{aligned}
$$

Therefore, this selection considers interactions between $\mathcal{B}_{k-1}$ and b_k, as well as the Shapley values of b_k in the self-context.

C.4.2. Case with r=2

For $k = 1$, we find b_1 and b_2 so that $f(\{b_1, b_2\})$ is maximized. From Eqs. (4.1) and (4.1), this is formulated as follows:

$$
\begin{aligned}
\{b_1, b_2\} &= \underset{\{b'_1,b'_2\}\subset N}{\arg\max}\ f(\{b'_1, b'_2\}) - f(\emptyset) \\
&= \underset{\{b'_1,b'_2\}\subset N}{\arg\max}\ \phi^{\mathrm{sc}}(\{b'_1, b'_2\}) \\
&= \underset{\{b'_1,b'_2\}\subset N}{\arg\max}\ I^{\mathrm{sc}}(\{b'_1, b'_2\}) \\
&\qquad + \sum_{B'\in\mathcal{P}^*(\{b'_1,b'_2\})} \phi^{\mathrm{sc}}(B') \\
&= \underset{\{b'_1,b'_2\}\subset N}{\arg\max}\ I^{\mathrm{sc}}(\{b'_1, b'_2\}) \\
&\qquad + \phi^{\mathrm{sc}}(\{b'_1\}) + \phi^{\mathrm{sc}}(\{b'_2\}).
\end{aligned}
$$

Therefore, this selection considers interactions between b_1 and b_2, as well as their Shapley values in the self-context.

For $k \geq 2$, we find b_{2k-1} and b_{2k} so that $f(\{\mathcal{B}_{k-1}\} \cup \{b_{2k-1}, b_{2k}\})$ is maximized. From Eqs. (4.1) and (4.1), this is formulated as follows:

$$
\begin{aligned}
&\{b_{2k-1}, b_{2k}\} \\
&= \underset{\{b'_{2k-1},b'_{2k}\}\subseteq N\setminus\mathcal{B}_{k-1}}{\arg\max}\ f(\{\mathcal{B}_{k-1}\}\cup\{b'_{2k-1}, b'_{2k}\}) - f(\emptyset) \\
&= \underset{\{b'_{2k-1},b'_{2k}\}\subseteq N\setminus\mathcal{B}_{k-1}}{\arg\max}\ \phi^{\mathrm{sc}}(\{\mathcal{B}_{k-1}\}\cup\{b'_{2k-1}, b'_{2k}\}) \\
&= \underset{\{b'_{2k-1},b'_{2k}\}\subseteq N\setminus\mathcal{B}_{k-1}}{\arg\max}\ I^{\mathrm{sc}}(\{\mathcal{B}_{k-1}\}\cup\{b'_{2k-1}, b'_{2k}\}) \\
&\qquad + \sum_{B'\in\mathcal{P}^*(\{\mathcal{B}_{k-1}\}\cup\{b'_{2k-1},b'_{2k}\})} \phi^{\mathrm{sc}}(B') \\
&= \underset{\{b'_{2k-1},b'_{2k}\}\subseteq N\setminus\mathcal{B}_{k-1}}{\arg\max}\ I^{\mathrm{sc}}(\{\mathcal{B}_{k-1}\}\cup\{b'_{2k-1}, b'_{2k}\}) \\
&\qquad + \phi^{\mathrm{sc}}(\{\mathcal{B}_{k-1}\}\cup\{b'_{2k-1}\}) \\
&\qquad + \phi^{\mathrm{sc}}(\{\mathcal{B}_{k-1}\}\cup\{b'_{2k}\}) \\
&\qquad + \phi^{\mathrm{sc}}(\{b'_{2k-1}\}\cup\{b'_{2k}\}) \\
&\qquad + \phi^{\mathrm{sc}}(\{b'_{2k-1}\}) \\
&\qquad + \phi^{\mathrm{sc}}(\{b'_{2k}\}) \\
&\qquad + \phi^{\mathrm{sc}}(\mathcal{B}_{k-1}) \\
&= \underset{\{b'_{2k-1},b'_{2k}\}\subseteq N\setminus\mathcal{B}_{k-1}}{\arg\max}\ I^{\mathrm{sc}}(\{\mathcal{B}_{k-1}\}\cup\{b'_{2k-1}, b'_{2k}\}) \\
&\qquad + \phi^{\mathrm{sc}}(\{\mathcal{B}_{k-1}\}\cup\{b'_{2k-1}\}) \\
&\qquad + \phi^{\mathrm{sc}}(\{\mathcal{B}_{k-1}\}\cup\{b'_{2k}\}) \\
&\qquad + \phi^{\mathrm{sc}}(\{b'_{2k-1}\}\cup\{b'_{2k}\}) \\
&\qquad + \phi^{\mathrm{sc}}(\{b'_{2k-1}\}) \\
&\qquad + \phi^{\mathrm{sc}}(\{b'_{2k}\}).
\end{aligned}
$$

Therefore, this selection considers interactions among the three elements $\mathcal{B}_{k-1}$, b_{2k-1}, and b_{2k}, as well as Shapley values derived from each patch combination in self-context.

C.5. Patch deletion

We explain the process of patch deletion. In this process, we utilize the Shapley value, which measures the contribution in the absence of a player.

$$
\phi_{\mathrm{d}}(i \mid N) \overset{\mathrm{def}}{=} \sum_{D\subseteq N, i\in D} P_{\mathrm{d}}(D\setminus\{i\} \mid N)[f(D) - f(D\setminus\{i\})],
$$

where $P_{\mathrm{d}}(A \mid B) = \frac{(|B|-|A|-1)!|A|!}{|B|!}$. This Shapley value quantifies the average impact attributable to the removal of player i. To effectively incorporate Shapley values and interactions, our approach adopts a greedy strategy based on the following full-context values.

Definition 6 *Let N be a set of all players. The **full-context Shapley value** of $S \subset N$ is defined by*

$$
\phi^{\mathrm{fc}}_{\mathrm{d}}(S \mid N) \overset{\mathrm{def}}{=} P_{\mathrm{d}}(N\setminus S \mid N)[f(N) - f(N\setminus S)].
$$

The full-context values were introduced in [52]. These values restrict D in Eq. (C.5) to a fixed set. Next, we define full-context interaction as follows:

Definition 7 *The **full-context interaction** of $S \subset N$ is defined by*

$$I_{\mathrm{d}}^{\mathrm{fc}}(S) \stackrel{\mathrm{def}}{=} \phi_{\mathrm{d}}^{\mathrm{fc}}(S \mid (N \setminus S) \cup \{S\}) - \sum_{S' \in \mathcal{P}^*(S)} \phi_{\mathrm{d}}^{\mathrm{fc}}(S' \mid N \setminus (S \setminus S')).$$

where $\mathcal{P}^*(S)$ denotes the set of all strict subsets of S, i.e., $\mathcal{P}^*(S) = \mathcal{P}(S) \setminus \{S\}$.

Similar to patch insertion, our greedy strategy in the deletion setup in Problem 1 works as follows.

Initial step (k=1). The set of patches $B_1 \subset N$ that minimize $f(N \setminus B_1)$ is selected.

$$\begin{aligned} B_1 &= \underset{B \subseteq N, |B|=r}{\arg\min}\ f(N \setminus B) - f(N) \\ &= \underset{B \subseteq N, |B|=r}{\arg\max}\ f(N) - f(N \setminus B) \\ &= \underset{B \subseteq N, |B|=r}{\arg\max}\ \phi_{\mathrm{d}}^{\mathrm{fc}}(B \mid N), \end{aligned}$$

In the last equality, we omit $P_{\mathrm{d}}(N \setminus B \mid N)$, as it remains constant regardless of the patch selection. Note that we treat B_1 as a single player. While this apparently considers the Shapley value of a fixed set size r only, we can show that this is not the case. From Eq (7), we have the following decomposition.

$$\phi_{\mathrm{d}}^{\mathrm{fc}}(B \mid N) = I_{\mathrm{d}}^{\mathrm{fc}}(B) + \sum_{B' \in \mathcal{P}^*(B)} \phi_{\mathrm{d}}^{\mathrm{fc}}(B' \mid N \setminus (B \setminus B')).$$

The first term of Eq. (C.5) is the interaction among r patches in B. The second term is the sum of the full-context Shapley values over all subsets of B except B itself. Hence, the selection of B_1 takes into account various size of subsets and interactions of patches (for $r = 1$, only the individual contribution of B is considered).

General k. The new set B_k is determined in combination with the sets $\mathcal{B}_{k-1} = \bigcup_{i=1}^{k-1} B_i$ from the previous steps. As in the initial step, we have

$$\begin{aligned} B_k &= \underset{B \subseteq N \setminus \mathcal{B}_{k-1}, |B|=r}{\arg\min}\ f(N \setminus (\{\mathcal{B}_{k-1}\} \cup B)) - f(N) \\ &= \underset{B \subseteq N \setminus \mathcal{B}_{k-1}, |B|=r}{\arg\max}\ f(N) - f(N \setminus (\{\mathcal{B}_{k-1}\} \cup B)) \\ &= \underset{B \subseteq N \setminus \mathcal{B}_{k-1}, |B|=r}{\arg\max}\ \phi_{\mathrm{d}}^{\mathrm{fc}}(\{\mathcal{B}_{k-1}\} \cup B \mid N). \end{aligned}$$

Again, $\phi_{\mathrm{d}}^{\mathrm{fc}}(\{\mathcal{B}_{k-1}\} \cup B \mid N)$ can be interpreted as a sum of interaction and Shapley values.

$$\begin{aligned} &\phi_{\mathrm{d}}^{\mathrm{fc}}(\{\mathcal{B}_{k-1}\} \cup B \mid N) \\ &= I_{\mathrm{d}}^{\mathrm{fc}}(\{\mathcal{B}_{k-1}\} \cup B) \\ &\quad + \sum_{B' \in \mathcal{P}^*(\{\mathcal{B}_{k-1}\} \cup B)} \phi_{\mathrm{d}}^{\mathrm{fc}}(B' \mid N \setminus ((\{\mathcal{B}_{k-1}\} \cup B) \setminus B')). \end{aligned}$$

where $\mathcal{B}_{k-1}$ is regarded as a single patch. Hence, the selection of B_k considers the interaction between $r+1$ patches (r patches in B_k and a single patch treated as $\mathcal{B}_{k-1}$), as well as the Shapley values of each patch combination.

C.6. Specific example of patch deletion

We provide specific examples for cases with $r = 1$ and $r = 2$ in patch deletion described in App. C.5.

C.6.1. Case with r=1

For step $k = 1$, from Eqs. (C.5) and (C.5), only the highest full-context Shapley value $\phi_{\mathrm{d}}^{\mathrm{fc}}(\{b_1\}|N) = n^{-1}[f(N) - f(N \setminus \{b_1\})]$ becomes the optimal set $B_1 = \{b_1\}$.

For $k \geq 2$, we find b_k that minimizes $f(N \setminus \{\{\mathcal{B}_{k-1}\} \cup \{b_k\}\})$. From Eqs. (C.5) and (C.5), this is formulated as follows:

$$\begin{aligned} b_k &= \underset{b \in N \setminus \mathcal{B}_{k-1}}{\arg\min}\ f(N \setminus (\{\mathcal{B}_{k-1}\} \cup \{b\})) - f(N) \\ &= \underset{b \in N \setminus \mathcal{B}_{k-1}}{\arg\max}\ f(N) - f(N \setminus (\{\mathcal{B}_{k-1}\} \cup \{b\})) \\ &= \underset{b \in N \setminus \mathcal{B}_{k-1}}{\arg\max}\ \phi_{\mathrm{d}}^{\mathrm{fc}}(\{\mathcal{B}_{k-1}\} \cup \{b\} \mid N) \\ &= \underset{b \in N \setminus \mathcal{B}_{k-1}}{\arg\max}\ I_{\mathrm{d}}^{\mathrm{fc}}(\{\mathcal{B}_{k-1}\} \cup \{b\}) \\ &\quad + \sum_{B' \in \mathcal{P}^*(\{\mathcal{B}_{k-1}\} \cup \{b\})} \phi_{\mathrm{d}}^{\mathrm{fc}}(B' \mid N \setminus ((\{\mathcal{B}_{k-1}\} \cup \{b\}) \setminus B')) \\ &= \underset{b \in N \setminus \mathcal{B}_{k-1}}{\arg\max}\ I_{\mathrm{d}}^{\mathrm{fc}}(\{\mathcal{B}_{k-1}\} \cup \{b\}) \\ &\qquad + \phi_{\mathrm{d}}^{\mathrm{fc}}(\{b\} \mid N \setminus \mathcal{B}_{k-1}) + \phi_{\mathrm{d}}^{\mathrm{fc}}(\mathcal{B}_{k-1} \mid N \setminus \{b\}). \end{aligned}$$

Therefore, this selection considers interactions between $\mathcal{B}_{k-1}$ and b_k, as well as Shapley values in the full-context.

C.6.2. Case with r=2

For $k = 1$, we find b_1 and b_2 so that $f(N \setminus \{b_1, b_2\})$ is minimized. From Eqs. (C.5) and (C.5), this is formulated as

follows:

$$
\begin{aligned}
\{b_1, b_2\} &= \underset{\{b'_1,b'_2\}\subset N}{\arg\min}\ f(N \setminus \{b'_1, b'_2\}) - f(N) \\
&= \underset{\{b'_1,b'_2\}\subset N}{\arg\max}\ f(N) - f(N \setminus \{b'_1, b'_2\}) \\
&= \underset{\{b'_1,b'_2\}\subset N}{\arg\max}\ \phi^{\mathrm{fc}}_{\mathrm{d}}(\{b'_1, b'_2\} \mid N) \\
&= \underset{\{b'_1,b'_2\}\subset N}{\arg\max}\ I^{\mathrm{fc}}_{\mathrm{d}}(\{b'_1, b'_2\}) \\
&\quad + \sum_{B'\in\mathcal{P}^*(\{b'_1,b'_2\})} \phi^{\mathrm{fc}}_{\mathrm{d}}(B' \mid N \setminus (\{b'_1, b'_2\} \setminus B')) \\
&= \underset{\{b'_1,b'_2\}\subset N}{\arg\max}\ I^{\mathrm{fc}}_{\mathrm{d}}(\{b'_1, b'_2\}) \\
&\quad + \phi^{\mathrm{fc}}_{\mathrm{d}}(\{b'_1\} \mid N \setminus \{b'_2\}) + \phi^{\mathrm{fc}}_{\mathrm{d}}(\{b'_2\} \mid N \setminus \{b'_1\}).
\end{aligned}
$$

Therefore, this selection considers interactions between b_1 and b_2, as well as their Shapley values in the full-context.

For $k \geq 2$, we find b_{2k-1} and b_{2k} so that $f(N \setminus \{\{\mathcal{B}_{k-1}\} \cup \{b_{2k-1}, b_{2k}\}\})$ is minimized. From Eqs. (C.5) and (C.5), this is formulated as follows:

$$
\begin{aligned}
&\{b_{2k-1}, b_{2k}\} \\
&= \underset{\{b'_{2k-1},b'_{2k}\}\subseteq N\setminus\mathcal{B}_{k-1}}{\arg\min}\ f(N \setminus (\{\mathcal{B}_{k-1}\} \cup \{b'_{2k-1}, b'_{2k}\})) - f(N) \\
&= \underset{\{b'_{2k-1},b'_{2k}\}\subseteq N\setminus\mathcal{B}_{k-1}}{\arg\max}\ f(N) - f(N \setminus (\{\mathcal{B}_{k-1}\} \cup \{b'_{2k-1}, b'_{2k}\})) \\
&= \underset{\{b'_{2k-1},b'_{2k}\}\subseteq N\setminus\mathcal{B}_{k-1}}{\arg\max}\ \phi^{\mathrm{fc}}_{\mathrm{d}}(\{\mathcal{B}_{k-1}\} \cup \{b'_{2k-1}, b'_{2k}\} \mid N) \\
&= \underset{\{b'_{2k-1},b'_{2k}\}\subseteq N\setminus\mathcal{B}_{k-1}}{\arg\max}\ I^{\mathrm{fc}}_{\mathrm{d}}(\{\mathcal{B}_{k-1}\} \cup \{b'_{2k-1}, b'_{2k}\}) \\
&\quad + \sum_{B'\in\mathcal{P}^*(\{\mathcal{B}_{k-1}\}\cup\{b'_{2k-1},b'_{2k}\})} \phi^{\mathrm{fc}}_{\mathrm{d}}(B' \mid N \setminus \\
&\qquad\qquad (((\{\mathcal{B}_{k-1}\} \cup \{b'_{2k-1}, b'_{2k}\}) \setminus B')) \\
&= \underset{\{b'_{2k-1},b'_{2k}\}\subseteq N\setminus\mathcal{B}_{k-1}}{\arg\max}\ I^{\mathrm{fc}}_{\mathrm{d}}(\{\mathcal{B}_{k-1}\} \cup \{b'_{2k-1}, b'_{2k}\}) \\
&\qquad + \phi^{\mathrm{fc}}_{\mathrm{d}}(\{b'_{2k-1}\} \cup \{b'_{2k}\} \mid N \setminus \mathcal{B}_{k-1}) \\
&\qquad + \phi^{\mathrm{fc}}_{\mathrm{d}}(\{\mathcal{B}_{k-1}\} \cup \{b'_{2k-1}\} \mid N \setminus \{b'_{2k}\}) \\
&\qquad + \phi^{\mathrm{fc}}_{\mathrm{d}}(\{\mathcal{B}_{k-1}\} \cup \{b'_{2k}\} \mid N \setminus \{b'_{2k-1}\}) \\
&\qquad + \phi^{\mathrm{fc}}_{\mathrm{d}}(\{b'_{2k-1}\} \mid N \setminus (\{\mathcal{B}_{k-1}\} \cup \{b'_{2k}\})) \\
&\qquad + \phi^{\mathrm{fc}}_{\mathrm{d}}(\{b'_{2k}\} \mid N \setminus (\{\mathcal{B}_{k-1}\} \cup \{b'_{2k-1}\})) \\
&\qquad + \phi^{\mathrm{fc}}_{\mathrm{d}}(\mathcal{B}_{k-1} \mid N \setminus (\{b'_{2k-1}\} \cup \{b'_{2k}\})).
\end{aligned}
$$

Therefore, this selection considers interactions among the three elements $\mathcal{B}_{k-1}$, b_{2k-1}, and b_{2k}, as well as Shapley values derived from each patch combination in full-context.

Algorithm 3 Generation of a heat map based on identified patches

Input: Set of identified patches $\{b_1, \ldots, b_n\}$, set of values from a reward function $\{f_1, \ldots, f_n\}$.
Output: Heat map $M \in \mathbb{R}^{h\times w}$ for an input image of size $h \times w$.

$M = \mathbf{0}$
for $i = 1, \ldots, n$ **do**
  if patch insertion **then**
    if $i = 1$ **then**
      $s \leftarrow 1.0$
    else
      $s \leftarrow 1.0 - f_{i-1}$
    end if
  else if patch deletion **then**
    if $i = 1$ **then**
      $s \leftarrow 1.0$
    else
      $s \leftarrow f_{i-1}$
    end if
  end if
  `# Identify the diagonal corners of the patch` b_i `in the image.`
  $\{(x_1^{\mathrm{b}}, y_1^{\mathrm{b}}), (x_2^{\mathrm{b}}, y_2^{\mathrm{b}})\} \leftarrow \mathrm{get\_position}(b_i)$

$$
M_{i,j} \leftarrow \begin{cases} s & (\text{if } y_1^{\mathrm{b}} \leq i \leq y_2^{\mathrm{b}} \\ & \quad \text{and } x_1^{\mathrm{b}} \leq j \leq x_2^{\mathrm{b}}), \\ M_{i,j} & (\text{otherwise}). \end{cases}
$$

end for
return M

C.7. Heat maps generation from identified patches

We describe the process for generating heat maps from identified patches. To efficiently identify important regions, it is desirable to emphasize patches that lead to significant changes in the reward function values, either through patch insertion or patch deletion. To address this, given a set of identified patches, $\{b_1, \ldots, b_n\}$ and their corresponding reward function values for each step, $\{f_1, \ldots, f_n\}$, where f_k is the reward function value at step k, we determine the importance of each patch based on the corresponding reward function value. Specifically, the first identified patch, which contributes most significantly, is assigned the highest importance value of 1.0. For the subsequent patches, we assume that they contribute to the remaining change in the score (an increase for patch insertion or a decrease for patch deletion) and define their importance directly from

the reward value at the previous step. We show the pseudo-code for the heat map generation process in Algorithm 3. In Algorithm 3, $\mathrm{get\_position}(b_i)$ returns the diagonal corners $(x_1^{\mathrm{b}}, y_1^{\mathrm{b}})$ and $(x_2^{\mathrm{b}}, y_2^{\mathrm{b}})$ of the patch b_i in the image. This process generates a heat map that effectively highlights regions with significant changes in the reward function values.

In the above framework, for $r \geq 2$, it is necessary to assign a reward function value individually to each patch in the identified patch set B_k. Therefore, we individually assign this value by sequentially adding each patch in B_k to $\mathcal{B}_{k-1}$ and computing the corresponding reward function value. In this process, the order of adding patches is determined such that the sum of the corresponding rewards is maximized (patch insertion) or minimized (patch deletion). Although this requires $r!$ computations for all possible arrangements of the r patches, the practical values of r are small (e.g., $r \leq 5$), making the computational cost negligible in the overall algorithm.

C.8. Implementation details

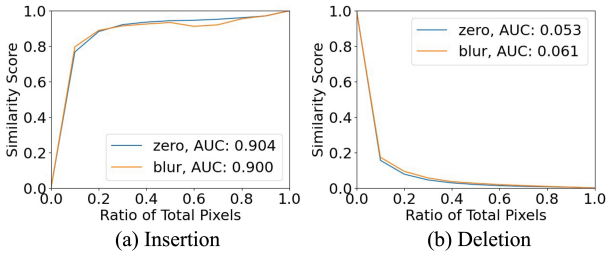
We describe the method for patch decomposition. Unlike image classification models, object detectors detect instances of varying sizes. Consequently, the size of each feature (e.g., a human head, hand, etc.) depends on the instance size. Ideally, each divided patch should capture these features. Therefore, we determine the patch size based on the size of the detected bounding boxes. Specifically, the patch size is determined as follows:

$$d_{\mathrm{h}} = d_{\mathrm{w}} = \begin{cases} 24 & (0 < R_{\mathrm{box}} \leq 0.01) \\ 16 & (0.01 < R_{\mathrm{box}} \leq 0.2) \\ 8 & (0.2 < R_{\mathrm{box}}) \end{cases},$$

where d_{h} and d_{w} represent the vertical and horizontal divisions of the image, respectively, and R_{box} denotes the ratio of the detected box area to the total image area.

Next, we describe the patches considered for calculation in patch insertion and patch deletion. For patches divided according to Eq. (C.8), patch insertion and patch deletion are performed. In this process, particularly when the detected instance size is small, it is assumed that features from regions far from the instance contribute little to the result (e.g., an object located in the lower right of the image is irrelevant to information in the upper left). Additionally, as the patch size becomes smaller, computing for all patches incurs a high computational cost. Therefore, for the detection results with $R_{\mathrm{box}} \leq 0.2$, we limit the target patches for reward computation. Let $\{x_1, y_1, x_2, y_2\}$ be a set of coordinates of the detected bounding box, and let h and w be the height and width of the image. Then, we compute the reward only for the patches with center coordinates (C_x, C_y) within the following range.

$$x_1 - \frac{w}{7} \leq C_x \leq x_2 + \frac{w}{7}$$
$$y_1 - \frac{h}{7} \leq C_y \leq y_2 + \frac{h}{7}.$$



(a) Insertion (b) Deletion

Figure I. Comparison results of the two filling methods, zero filling and blurred pixel values, using (a) the insertion metric and (b) the deletion metric. The similarity score is computed using Eq. (3).

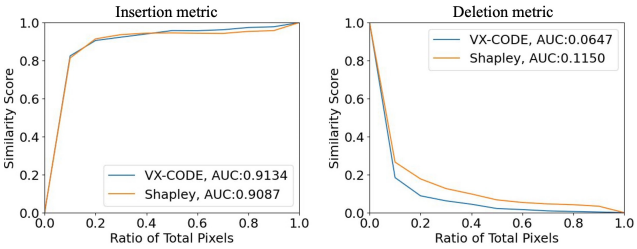


Figure J. The comparison of insertion and deletion metrics between VX-CODE and Shapley values estimated via Monte Carlo sampling.

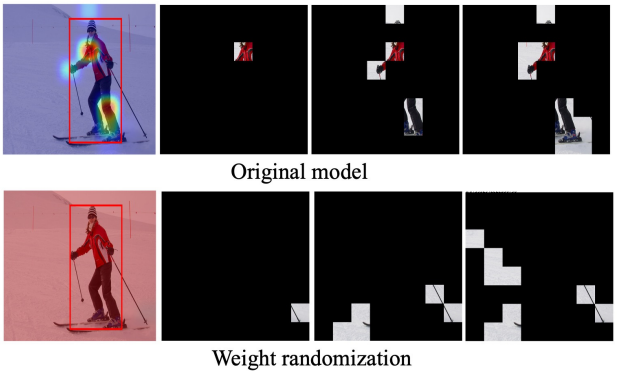


Figure K. Comparison of heat maps generated by VX-CODE with patch insertion for the original and weight-randomized models.

C.9. Axiomatic properties of the π^*-index

Shapley values are known to satisfy several useful axioms, such as *Null player*, *Linearity*, *Efficiency*, and *Symmetry*. Let ϕ_i be the Shapley value for player i and $N = \{1, ..., n\}$

Table 5. SSIM and rank correlation results under weight randomization of DETR.

Metric	SSIM	Rank correlation
Patch insertion	0.144	-0.028
Patch deletion	0.312	-0.014

be the index set of players. Each axiom is defined as follows:

- **Null player**
  A null player is assigned a value of 0. A player i is null in f if $f(S \cup \{i\}) = f(S)$.
- **Linearity**
  For any reward function f and g and any real number α and β,

$$\phi_i(\alpha f + \beta g) = \alpha\, \phi_i(f) + \beta\, \phi_i(g)$$

- **Efficiency**
  The full value of the grand coalition is completely distributed among all players:

$$\sum_{i=1}^{n} \phi_i(f) \;=\; f(N).$$

- **Symmetry**
  If two players i and j are interchangeable (i.e., $f(S \cup \{i\}) = f(S \cup \{j\})$ for $S \subseteq N \setminus \{i, j\}$),

$$\phi_i(f) = \phi_j(f).$$

We here revisit the π^*-index . As shown in Eq. (5), in a sequential coalitional game where players are recruited into the game, k-player in π^* is assigned the π^*-index as follows:

$$\psi(k|\pi^*) = f(\pi^*(1), \ldots, \pi^*(k)) - f(\pi^*(1), \ldots, \pi^*(k-1)).$$

Similarly, in a sequential coalitional game where players are removed from the game (we refer to this situation as a player elimination setting), π^*-index for this setting is defined as follows:

Definition 8 *Let $\pi^* \in \Pi$ be a permutation such that*

$$\pi^* = \arg\min_{\pi \in \Pi} \sum_{k=1}^{n} f(N \setminus \{\pi(1), \ldots, \pi(k)\}).$$

*The $\boldsymbol{\pi^*}$**-index** of the k-player for the player elimination setting is defined by*

$$\psi^{\text{es}}(k|\pi^*) = f(N \setminus \{\pi^*(1), \ldots, \pi^*(k-1)\}) - f(N \setminus \{\pi^*(1), \ldots, \pi^*(k)\}).$$

The ψ^{es} measures the contribution of each player by the decrease in reward upon their removal. It is worth noting that the greedy patch deletion process described in App. C.5 can be interpreted as the player elimination setting in a sequential coalitional game. As shown in Appendix C.5, at each step the algorithm selects the set of patches whose removal produces the largest decrease in the reward, which is equivalent to selecting the set with the largest full-context Shapley value. In this sense, the patch deletion process procedure induces a specific permutation $\hat{\pi}$ that defines a sequential coalitional game in the player elimination setting.

Next, we demonstrate that the π^*-index satisfies *Null player*, *Linearity* and *Efficiency*. In the following, we implicitly assume $f(\emptyset) = 0$. We denote the π^*-index for the k-player with the reward function f as $\psi_k(f)$. Based on Eqs. (C.9) and (8), the π^*-index satisfies each axiom as follows:

- **Null player**
  If the k-player is null player, then we have,

$$f(\pi^*(1), \ldots, \pi^*(k)) = f(\pi^*(1), \ldots, \pi^*(k-1)).$$

  Therefore, we obtain

$$\psi_k(f) = f(\pi^*(1), \ldots, \pi^*(k)) - f(\pi^*(1), \ldots, \pi^*(k-1)) = 0.$$

  It is straightforward to verify that the same result holds for the player elimination setting, using following fact:

$$f(N \setminus \{\pi^*(1), \ldots, \pi^*(k-1)\}) = f(N \setminus \{\pi^*(1), \ldots, \pi^*(k)\}).$$

- **Linearity**
  We define a new game as a linear combination of two reward functions: $u(S) = \alpha f(S) + \beta g(S)$. Then, the following holds:

$$\begin{aligned}
&\psi_k(u) \\
&= \psi_k(\alpha f + \beta g) \\
&= \alpha[f(\pi^*(1), \ldots, \pi^*(k)) - f(\pi^*(1), \ldots, \pi^*(k-1))] \\
&\quad + \beta[g(\pi^*(1), \ldots, \pi^*(k)) - g(\pi^*(1), \ldots, \pi^*(k-1))] \\
&= \alpha\psi_k(f) + \beta\psi_k(g).
\end{aligned}$$

  It is straightforward to verify that the same result holds for the player elimination setting.
- **Efficiency**
  We can show that the following holds by computing the

Table 6. Results of pointing game (PG) and energy-based pointing game (EPG) on MS-COCO. We use bounding box (B) and instance segmentation mask (M) annotations. These results are for 1,000 detected instances from DETR with a predicted class score > 0.7 and IoU > 0.8 with the ground truth. In VX-CODE, we set $r = 1$. The best values are indicated in bold.

Metric	PG (B) ↑	PG (M) ↑	EPG (B) ↑	EPG (M) ↑
Grad-CAM	.321	.242	.194	.133
Grad-CAM++	.322	.224	.173	.112
D-RISE	.889	.769	.130	.079
SSGrad-CAM	.618	.490	.314	.224
ODAM	.602	.455	.315	.215
SSGrad-CAM++	.721	.585	.320	.220
VX-CODE (patch insertion)	.951	.905	.520	.351
VX-CODE (patch deletion)	**.965**	**.953**	**.644**	**.443**

summation of each π^*-index .

$$\begin{aligned}\sum_{i=1}^{n}\psi_i(f) &= f(\pi^*(1)) - f(\emptyset)\\ &\quad + f(\pi^*(1),\pi^*(2)) - f(\pi^*(1))\\ &\quad + \dots\\ &\quad + f(\pi^*(1),\dots,\pi^*(n))\\ &\qquad - f(\pi^*(1),\dots,\pi^*(n-1))\\ &= f(\pi^*(1),...,\pi^*(n)) - f(\emptyset)\\ &= f(N).\end{aligned}$$

Similarly, the following holds for the player elimination setting:

$$\begin{aligned}\sum_{i=1}^{n}\psi_i^{\text{es}}(f) &= f(N) - f(N\setminus\pi^*(1))\\ &\quad + f(N\setminus\pi^*(1)) - f(N\setminus\{\pi^*(1),\pi^*(2)\})\\ &\quad + \dots\\ &\quad + f(N\setminus\{\pi^*(1),\dots,\pi^*(n-1)\})\\ &\qquad - f(N\setminus\{\pi^*(1),\dots,\pi^*(n)\})\\ &= f(N) - f(N\setminus\{\pi^*(1),\dots,\pi^*(n)\})\\ &= f(N) - f(\emptyset)\\ &= f(N).\end{aligned}$$

The π^*-index is computed using a single permutation, π^*, which is the optimal case that satisfies Eq. (5) or Eq. (8). Therefore, in the context of the π^*-index, the *Symmetry* axiom—which is defined under averaging over all permutations— is no longer applicable and naturally drops out.

D. Additional experimental results

In this section, we present additional experimental results not included in the main part.

D.1. Comparison of filling methods

As described in Sec. 5, we fill the masked regions with zeros in the proposed method. Here, we compare two filling methods, zero filling and blurred pixel values, using the insertion and deletion metrics. Figure I shows the results. We observe that both methods produce nearly identical results. These findings suggest that the choice of filling method has no significant impact on the proposed method.

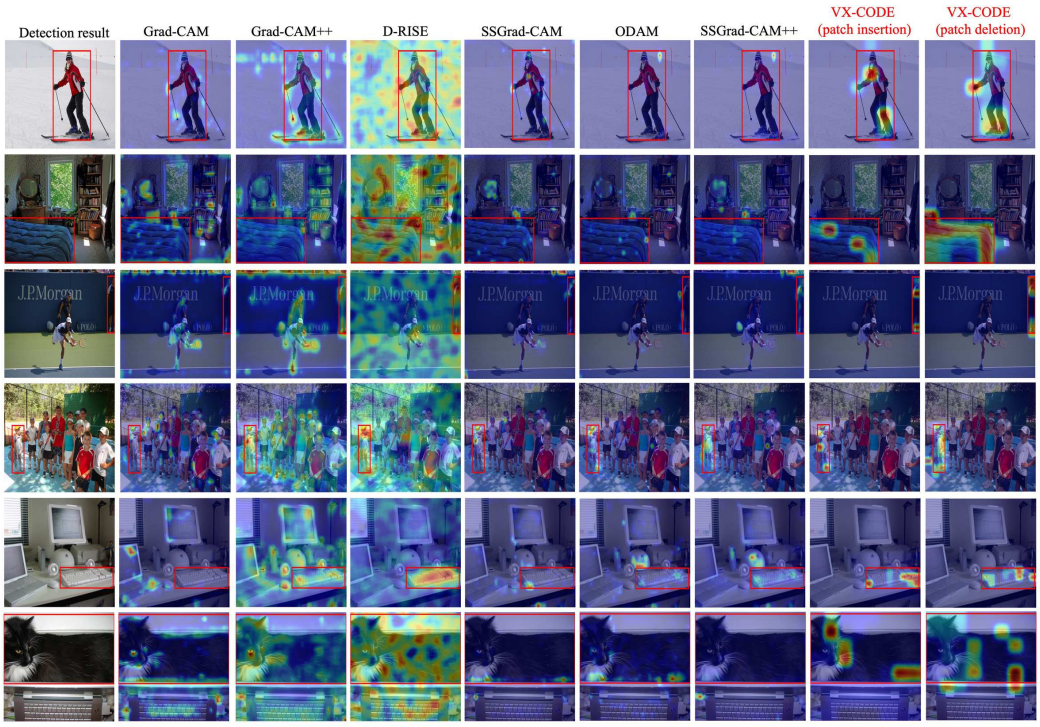
D.2. Comparison of Monte-Carlo based Shapley values

We compare the proposed method with Shapley values estimated via Monte Carlo sampling. Figure J shows the results of the insertion and deletion metrics, computed for 100 instances detected by DETR on MS-COCO. The number of Monte Carlo samples is set to 200, following prior studies, which already incurs a higher computational cost than our method with $r = 1$. As shown in Fig. J, the proposed method with $r = 1$ already outperforms the Shapley values estimated by Monte Carlo sampling.

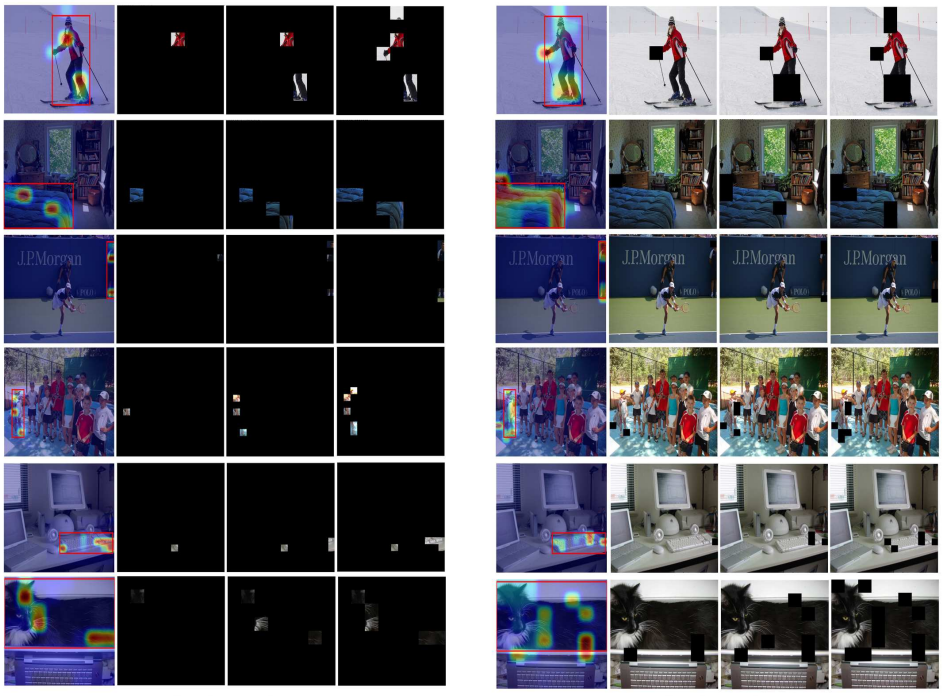
D.3. Sanity checks

We conduct sanity checks to evaluate the sensitivity of the proposed method to the object detectors. Sanity checks assess whether each explanation method provides meaningful explanations [1, 37]. Specifically, they examine how the outputs of the methods change when model parameters and labels are randomized. If a method lacks sensitivity to such randomizations and produces results similar to those of edge detectors, it suggests that the method does not rely on the training data or the model [1]. Consequently, such methods are inadequate for tasks such as identifying outliers in the data, as they are independent of both the model and the data.

We perform the checks using model weight randomization. Figure K shows an example generated by VX-CODE with patch insertion for DETR. The proposed method iden-



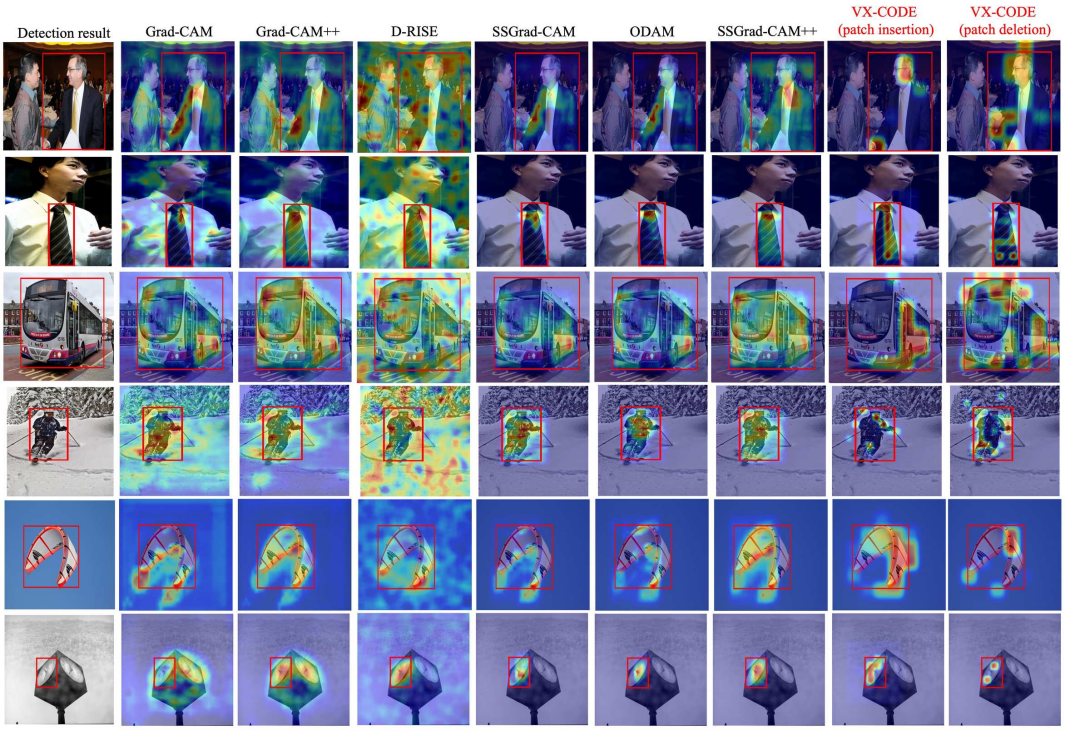
(a) Heat maps



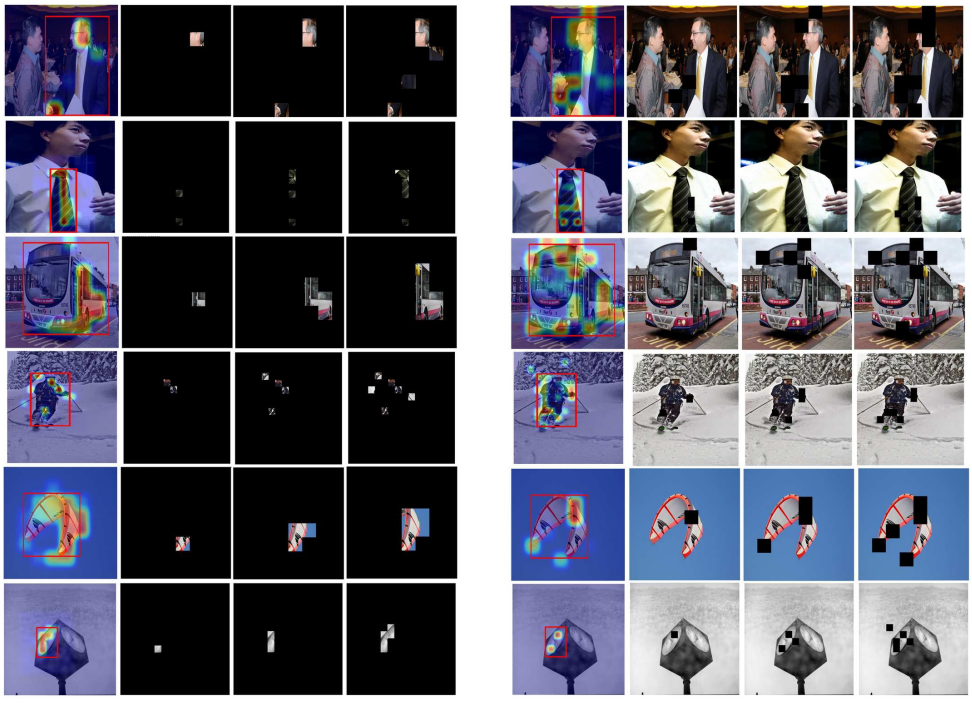
Patch insertion Patch deletion

(b) Identified patches from VX-CODE

Figure L. Additional results comparing the visual explanations generated by each method for detections from DETR. In VX-CODE, we set $r = 1$. (a) Heat maps generated by each method. (b) Patches identified from VX-CODE with patch insertion (left) and patch deletion (right). Each heat map corresponds to those shown in (a).



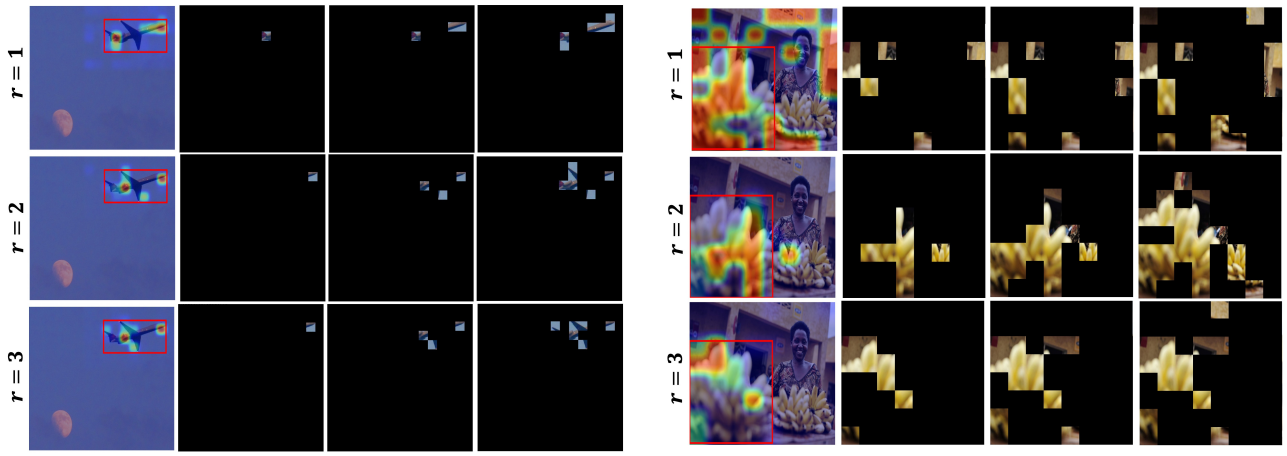
(a) Heat maps



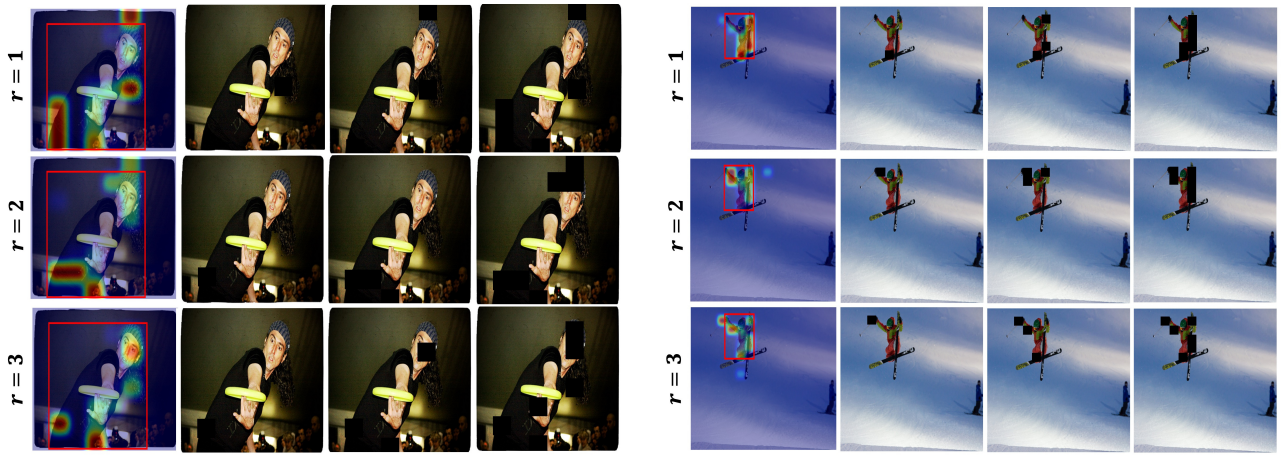
Patch insertion Patch deletion

(b) Identified patches from VX-CODE

Figure M. Additional results comparing the visual explanations generated by each method for detections from Faster R-CNN. In VX-CODE, we set $r = 1$. (a) Heat maps generated by each method. (b) Patches identified from VX-CODE with patch insertion (left) and patch deletion (right). Each heat map corresponds to those shown in (a).



(a) Patch insertion



(b) Patch deletion

Figure N. Additional results comparing the identified patches and generated heat maps by VX-CODE when changing r. (a) Results for patch insertion. (b) Results for patch deletion.

tifies uninformative patches under randomized weights. Table 5 presents the SSIM and rank correlation results computed for 100 detected instances using DETR on MS-COCO. These metrics evaluate the similarity between the explanations for the original and randomized models (with values closer to 1.0 indicating higher similarity). We observe significant drops in these metrics. These results indicate that the proposed method provides meaningful explanations that are sensitive to the model's parameters.

D.4. Pointing game

To evaluate the localization ability of the generated heat map, we conduct evaluations using pointing game [63] and energy-based pointing game [58]. The pointing game measures the proportion of heat maps in which the maximum value falls within the ground truth, such as a bounding box or instance mask. Meanwhile, the energy-based pointing game assesses the proportion of the heat map energy that lies within the ground truth. Note that these metrics only evaluate the localization ability of the generated heat map with respect to the ground truth instance rather than the faithfulness of the heat map. For example, as discussed in the main paper, object detectors may consider regions outside the bounding box when detecting objects, rendering these metrics potentially less effective in such cases.

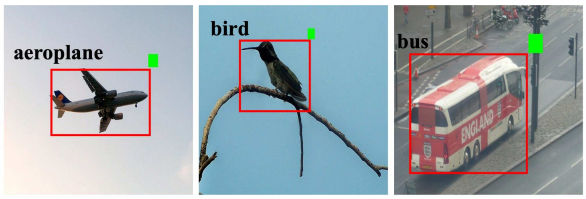
Table 6 summarizes the results for those metrics. We compute each metric for the 1,000 instances detected by DETR on MS-COCO, with a predicted class score > 0.7 and IoU > 0.8 with the ground truth. The results for VX-CODE are computed using the heat maps generated from patch insertion and patch deletion with $r = 1$. As shown in Tab. 6, both VX-CODE with patch insertion and patch deletion demonstrate high localization ability for the detected instances. Notably, VX-CODE with patch deletion outperforms patch insertion across these metrics. As described

Table 7. Results of AUC for insertion (Ins), deletion (Del), and overall (OA) metrics for the classes *aeroplane*, *bird*, *bottle*, and *bus*. In VX-CODE, we set $r = 1$. Each result is computed over 50 detection instances. The best values are indicated in bold.

Class	*aeroplane*			*bird*			*bottle*			*bus*		
Metric	Ins ↑	Del ↓	OA ↑	Ins ↑	Del ↓	OA ↑	Ins ↑	Del ↓	OA ↑	Ins ↑	Del ↓	OA ↑
Grad-CAM	.517	.026	.491	.601	.091	.510	.740	.017	.723	.534	.148	.386
Grad-CAM++	.658	.016	.642	.718	**.022**	.696	.853	.011	.842	.692	.028	.664
D-RISE	.680	**.014**	.666	.672	.025	.647	.843	**.009**	.834	.581	**.023**	.558
SSGrad-CAM	.649	.053	.596	.641	.078	.563	.803	.017	.786	.629	.090	.539
ODAM	.400	.089	.311	.470	.074	.396	.516	.048	.468	.502	.091	.411
SSGrad-CAM++	.796	.044	.752	**.822**	.061	.761	.902	.014	.888	.768	.050	.718
VX-CODE	**.891**	.034	**.857**	.807	.035	**.772**	**.904**	.014	**.890**	**.873**	.059	**.814**

Table 8. Results of AUC for insertion (Ins), deletion (Del), and overall (OA) metrics for the classes *dog*, *motorbike*, and *sofa*. In VX-CODE, we set $r = 1$. Each result is computed over 50 detection instances. The best values are indicated in bold.

Class	*dog*			*motorbike*			*sofa*		
Metric	Ins ↑	Del ↓	OA ↑	Ins ↑	Del ↓	OA ↑	Ins ↑	Del ↓	OA ↑
Grad-CAM	.605	.046	.559	.498	.199	.299	.465	.071	.394
Grad-CAM++	.697	.019	.678	.750	.023	.727	.620	.011	.609
D-RISE	.690	**.014**	.676	.679	**.013**	.666	.655	**.008**	.647
SSGrad-CAM	.591	.076	.515	.682	.118	.564	.524	.054	.470
ODAM	.510	.091	.419	.535	.142	.393	.460	.125	.335
SSGrad-CAM++	.772	.050	.722	.803	.056	.747	.747	.041	.706
VX-CODE	**.840**	.040	**.800**	**.899**	.039	**.860**	**.852**	.034	**.818**



(a) Synthetic data

(b) Without annotation data

Figure O. (a) Example of synthetic data: A square marker is placed in the upper-right corner outside the bounding box. These data include annotations for the class and bounding box. (b) Example of data without annotations: These data lack annotations and are treated as background during training.

in Sec. 5.2, patch deletion selects patches that reduce the class score, as defined in Eq. (3), This strategy focuses on eliminating critical instance features, enhancing its ability to localize objects effectively. These results support the advantages of patch deletion for object localization.

D.5. Additional results for visualizations

In Sec. 5.2, we show the explanations generated by each method. Figure L and Fig. M provide the additional examples of explanations generated by each method for detections from DETR and Faster R-CNN, respectively. Each figure includes the heat maps generated by each method and the patches identified by VX-CODE in the early steps. As described in Sec. 5.2, the proposed method identifies the important patches for detections by considering the collective contributions of multiple patches.

D.6. Additional results for patch combination

In Sec. 5.2, we demonstrate the effect of increasing r. Here, we show the additional results to further support this analysis. Figure N illustrates the effect of r on patch identification. As shown in Fig. N, the proposed method identifies more features, reflecting the effect of considering the collective contributions of a larger set of patches. For example, as shown on the left side of Fig. N (a) (detection for the aero-

plane), the identified features for $r = 1$ include the front and rear parts of the aeroplane, while the wings are additionally identified for $r = 2$ and $r = 3$, indicating that these features collectively contribute to the detection.

Increasing r also leads to more accurate identification of important patches. As shown on the right side of Fig. N (a) (detection of the banana), the case with $r = 1$ fails to identify the critical patches in the initial steps, whereas $r = 2$ and $r = 3$ successfully identify these patches at earlier steps. This improvement is particularly effective for patch insertion. Patch insertion gradually adds patches to an empty image. Therefore, at earlier steps, the reward function value changes only slightly due to the limited pixel information, making it difficult to identify important patches. Increasing r mitigates this problem by considering Shapley values and interactions among more patches, leading to more accurate identification of important patches.

D.7. Additional results for biased model

In Sec. 4, we demonstrate the effectiveness of the proposed method through experiments using a biased model. Here, we provide details about the experimental setup, including the synthetic dataset and the trained model, along with additional results.

Figure O illustrates an example of the constructed synthetic dataset. As shown in Fig. O (a), a square marker is placed outside the upper-right corner of the bounding box for seven classes: *aeroplane*, *bird*, *bottle*, *bus*, *dog*, *motorbike*, and *sofa*. These data include annotations for the corresponding class and its bounding box. In our constructed dataset, these synthetic data account for approximately 70%. In the remaining 30%, data corresponding to each class but without annotations are included and are treated as background during training. DETR trained on such data considers both the features of the marker outside the bounding box and those of the instance itself to detect objects and distinguish classes appropriately. In practice, this model achieves a mean average precision (mAP@50) with an IoU threshold > 0.5 of 83.41 for the seven classes on the synthetic test dataset. On the other hand, the mAP@50 drops to 34.73 on the original test dataset, where the square marker is not placed. These results indicate that DETR trained on the constructed synthetic dataset is heavily biased toward the square marker outside the bounding box.

We present results for the deletion, insertion, and overall metrics for each class in Tab. 7 and Tab. 8. Table 7 shows the results for the classes *aeroplane*, *bird*, *bottle*, and *bus*, and Tab 8 presents the results for the classes *dog*, *motorbike*, and *sofa*. Each result is computed over 50 detected instances. The proposed method outperforms existing methods, particularly in the insertion metric, where both instance and marker features must be identified for correct detection. As shown in Fig. 2, the proposed method effectively identified these two regions. These results show that VX-CODE provides explanations that consider both the instance and marker features in the detection. Interestingly, D-RISE and Grad-CAM++ marginally outperform VX-CODE in the deletion metric. In this metric, the similarity score significantly decreases when either the instance or the marker is identified. As shown in Fig. 2, D-RISE and Grad-CAM++ strongly highlight only the marker, contributing to their better performance. However, these methods fail to capture the instance features, leading to poor performance in the insertion metric.

Figure P shows additional examples of visualizations generated by each method for the biased model, and Fig. Q shows the patches identified from VX-CODE. As discussed in Sec 4, the proposed method efficiently identifies the features of both the instance and the marker, whereas existing methods tend to highlight only the marker (e.g., the result of Grad-CAM++ and D-RISE for the detection of the dog) or emphasize all features (e.g., the result of SSGrad-CAM++ for the detection of the bottle).

Table 9. Results of AUC for insertion (Ins), deletion (Del), and overall (OA) metrics. In VX-CODE, we set $r = 1$. Each result is computed over 1,000 detection instances with a predicted class score above 0.7 from Faster R-CNN on MS-COCO. The best values are indicated in bold.

Metric	Ins ↑	Del ↓	OA ↑
Grad-CAM	.627	.234	.393
Grad-CAM++	.837	.115	.722
D-RISE	.864	.089	.775
SSGrad-CAM	.819	.078	.741
ODAM	.866	.077	.789
SSGrad-CAM++	.898	.060	.838
VX-CODE	**.922**	**.025**	**.897**

Table 10. Results of AUC for insertion (Ins), deletion (Del), and overall (OA) metrics. In VX-CODE, we set $r = 1$. Each result is computed over 1,000 detection instances with a predicted class score above 0.7 from DETR on MS-COCO. The best values are indicated in bold.

Metric	Ins ↑	Del ↓	OA ↑
Grad-CAM	.799	.474	.325
Grad-CAM++	.729	.521	.208
D-RISE	.881	.397	.484
SSGrad-CAM	.884	.400	.484
ODAM	.885	.383	.502
SSGrad-CAM++	.864	.401	.463
VX-CODE	**.926**	**.238**	**.688**

Figure P. Additional results comparing the visualizations generated by each method for the biased model. In VX-CODE, we set $r = 1$. The results correspond to the classes *aeroplane*, *bird*, *bottle*, *bus*, *dog*, *motorbike*, and *sofa*, listed from the top row downward.

D.8. Quantitative evaluation for class explanations

We focus on explanations specifically for the predicted class. For this purpose, the reward function in VX-CODE is defined as the probability $p_y \in [0, 1]$ of the predicted class y for the original detected instance.

Table 9 shows the insertion, deletion, and overall metrics, based on 1,000 instances detected by Faster R-CNN on MS-COCO. The proposed method outperforms the other methods, demonstrating its ability to accurately identify the important regions for the predicted class.

D.9. Analysis for bounding box explanations

We evaluate explanations for bounding box generation. In this experiment, to generate explanations solely for the bounding boxes, we set the following reward function for VX-CODE and D-RISE as a variant of Eq. (3).

$$f(\mathcal{D}(x); L^x) = \max_{L \in \mathcal{D}(x)} \mathrm{IoU}(L^x, L),$$

where $\mathcal{D}$ is the object detector, L^x is the predicted bounding box for the original image, and $\mathrm{IoU}(L^x, L)$ denotes the IoU between the proposal bounding box L and L^x. For methods utilizing gradients (Grad-CAM, Grad-CAM++, SSGrad-CAM, ODAM, and SSGrad-CAM++), we first generate the heat maps for each bounding box coordinate in $\{x_1, y_1, x_2, y_2\}$ following [66]. Then, we obtain the heat map H_{box} by normalizing these heat maps and summing them together, as follows:

$$H_{\mathrm{box}} = \sum_{i \in \{x_1, y_1, x_2, y_2\}} \frac{H_i}{\max(H_i)},$$

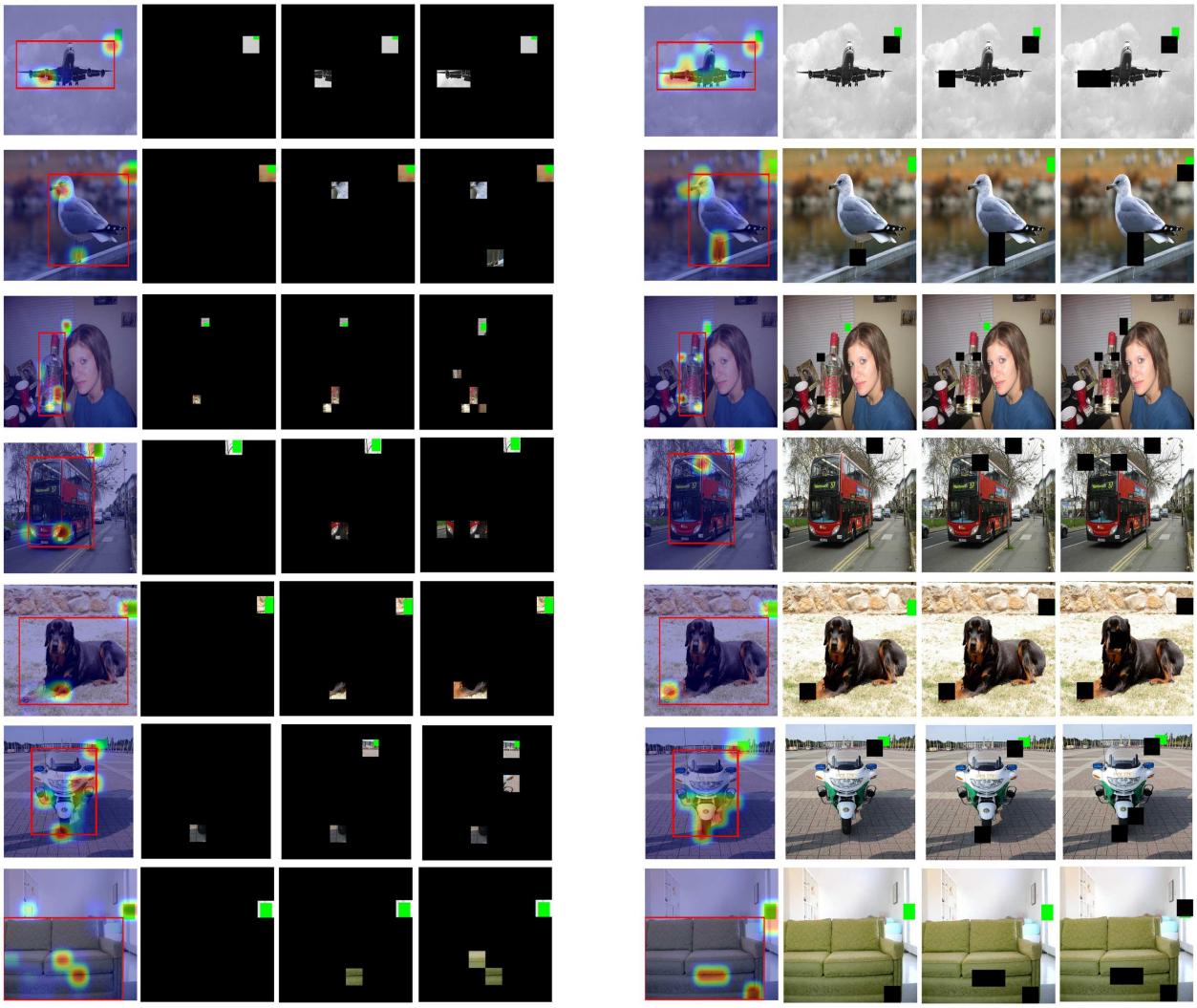


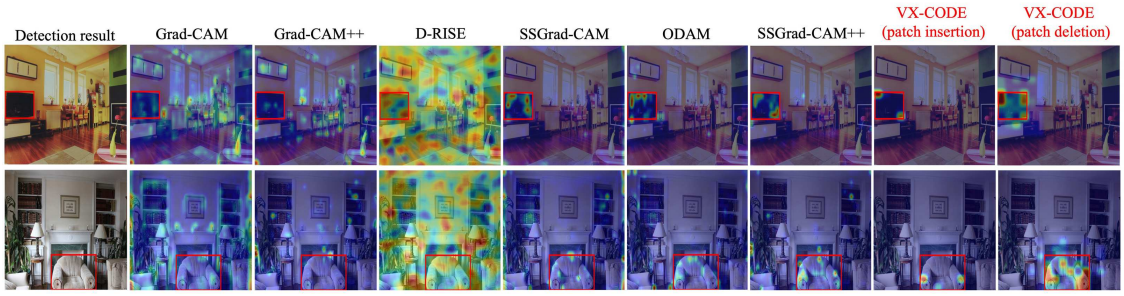
Figure Q. Patches identified from VX-CODE with patch insertion (left) and patch deletion (right) corresponding to the heat maps shown in Fig. P.

where $H_i \in \mathbb{R}^{H \times W}$ is the heat map for each bounding box coordinate on a feature map of size $H \times W$, and $\max(\cdot)$ calculates the maximum value of a matrix.

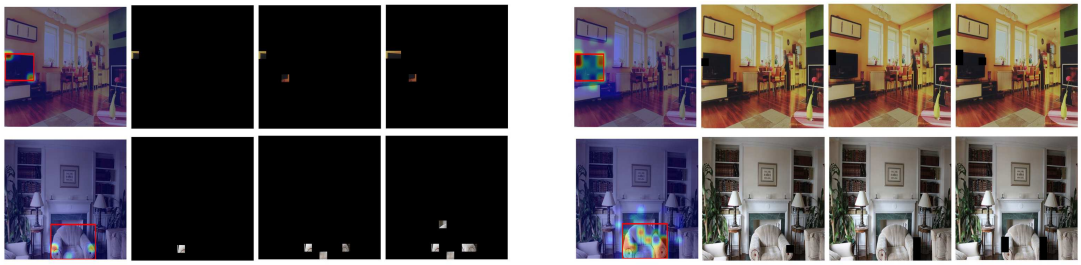
Table 10 shows the results of insertion, deletion, and overall metrics, based on 1,000 instances from the results of DETR on MS-COCO. Here, we use Eq. (D.9) as the similarity score calculated at each step. The proposed method outperforms the others, demonstrating its ability to accurately identify the important regions for the bounding box generation. Figure R compares the visualization results. Generally, the size of a bounding box is determined by several points (at least two). Therefore, considering their collective contributions is important. As shown in Fig. R, the proposed method accurately identifies important regions by considering such collective contributions. In contrast, D-RISE, which uses the same reward function, and methods utilizing gradients generate noisy results or also highlight irrelevant regions. These results support its superiority in the quantitative results shown in Tab. 10.

D.10. Multi-perspective explanations for detection results

The proposed method identifies important patches based on a reward function. One advantage of the proposed method is that it allows for the generation of multi-perspective explanations for a detection result through the adjustment of this reward function. To this end, we weight each term in



(a) Heat maps



Patch insertion Patch deletion

(b) Identified patches from VX-CODE

Figure R. Comparison results of the visual explanations for bounding box predictions generated by each method. In VX-CODE, we set $r = 1$. (a) Heat maps generated by each method. (b) Patches identified from VX-CODE with patch insertion (left) and patch deletion (right). Each heat map corresponds to those shown in (a).

the reward function of Eq. (3) with $\alpha \in [0, 1]$ as follows:

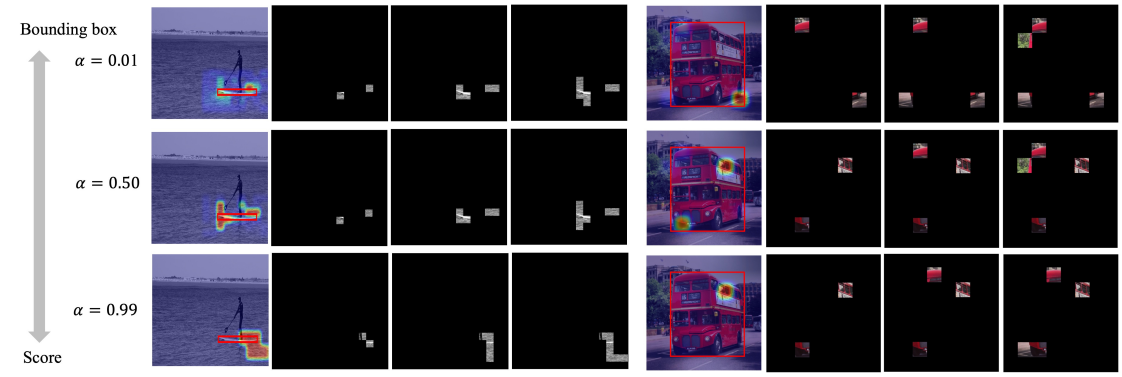
$$f(\mathcal{D}(x); (L^x, P^x)) = \max_{(L,P)\in\mathcal{D}(x)} \{\mathrm{IoU}(L^x, L)\}^{1-\alpha} \cdot \left\{ \frac{P^x \cdot P}{\|P^x\|\|P\|} \right\}^{\alpha}.$$

As shown in this equation, if α is close to 0, it identifies patches that focus more on the prediction of bounding boxes. If α is close to 1, it identifies patches that focus more on the prediction of class scores. If $\alpha = 0.5$, it identifies patches that balance their focus between the prediction of class scores and bounding boxes, which is essentially the same as using Eq. (3).

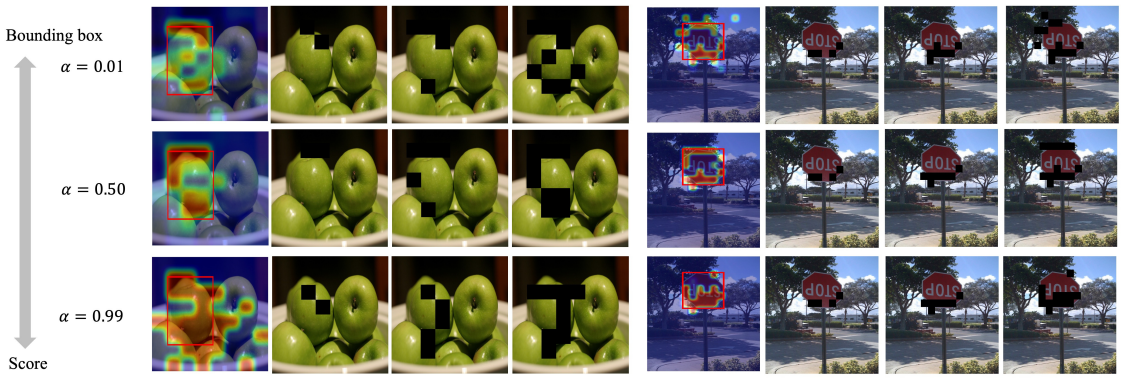
Figure S shows the identified patches and the generated heat maps from VX-CODE with (a) patch insertion and (b) patch deletion in the cases where α is 0.01, 0.50, and 0.99 These examples illustrate how the focus shifts between bounding box prediction and class prediction depending on the weight in the reward function. For example, as shown in the left example of (a) patch insertion, for the detection of the surfboard with $\alpha = 0.01$, the proposed method preferentially highlights area near the front and rear of the surfboard, indicating that these features significantly contribute to the bounding box prediction. In the case with $\alpha = 0.99$, the proposed method preferentially highlights the rear of the surfboard and the sea outside the bounding box, indicating that these features significantly contribute to the class (*surfboard*) prediction. In the case with $\alpha = 0.50$, the proposed method highlights these features (the front and rear of the surfboard and the sea outside the bounding box) in a balanced manner. A similar trend can be seen in the examples of (b) patch deletion. For example, as shown in the left example, for the detection of the apple with $\alpha = 0.01$, the proposed method preferentially highlights regions near the bounding box outline. In the case with $\alpha = 0.99$, the proposed method not only highlights the detected apple but also other apples. This identification is natural because the selection focuses on the class prediction of *apple*, and these features influence it in other proposals generated by the object detector. In the case with $\alpha = 0.50$, the proposed method only highlights features of the apple within the detected instance. As shown in these examples, the proposed method can adjust its focus between bounding box generation and class prediction by adjusting the weight in the reward function, enabling the generation of multi-perspective explanations.

D.11. Analysis of object detectors using VX-CODE

We present an example of analyzing object detectors using explanations generated by VX-CODE. Here, we analyze the differences in behavior between transformer-based and CNN-based architectures. Several studies have explored the differences in characteristics between transformers and

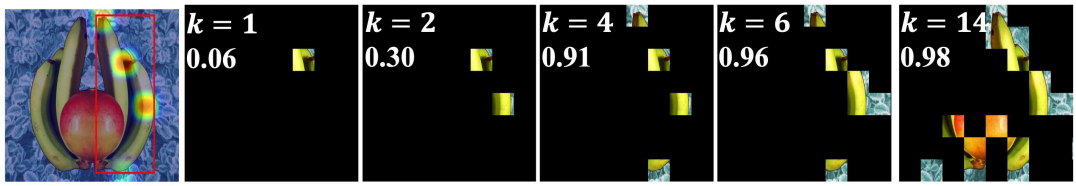


(a) Patch insertion

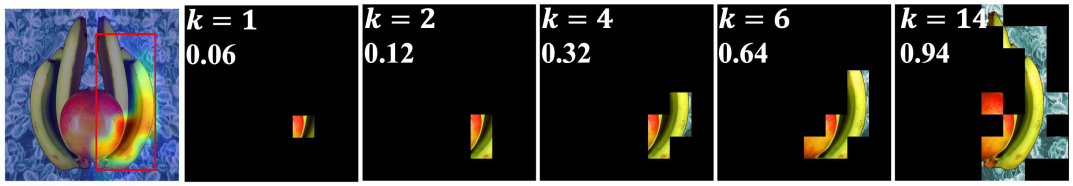


(b) Patch deletion

Figure S. The differences in regions identified by VX-CODE when varying α in Eq. (D.10), with (a) patch insertion and (b) patch deletion.



(a) DETR



(b) Faster R-CNN

Figure T. The differences in identified regions by VX-CODE with patch insertion between (a) DETR and (b) Faster R-CNN. Each figure shows the number of identified patches k and the corresponding reward value.

CNNs, suggesting that transformers utilize more global information and exhibit more compositionality compared to CNNs [24, 41, 57]. In this experiment, to analyze such differences in object detectors, we use VX-CODE with patch insertion to identify important regions for the same detection results detected by DETR and Faster R-CNN.

Figure T shows the identified patches and generated heat maps for detections of a banana produced by DETR and

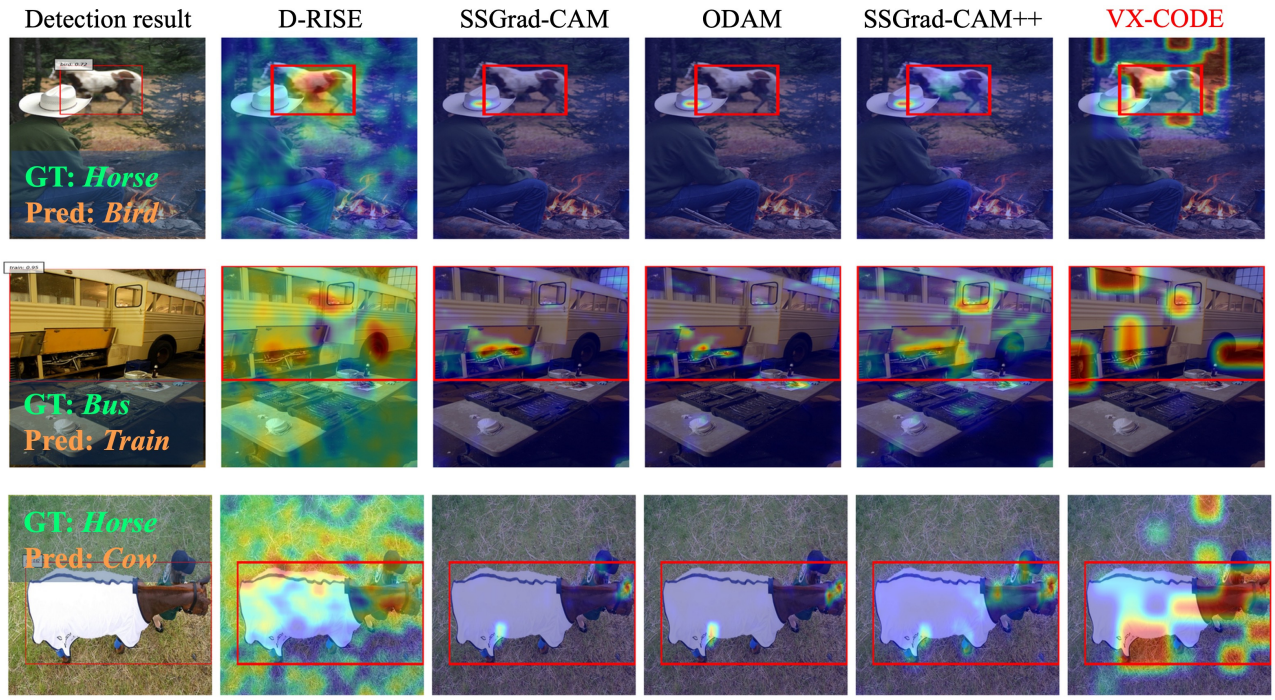


Figure U. Visualization results for misclassification generated by D-RISE, SSGrad-CAM, ODAM, SSGrad-CAM++, and VX-CODE with patch insertion. In the first row, the horse is misclassified as a bird; in the second row, the bus is misclassified as a train; and in the third row, the horse is misclassified as a dog.

Faster R-CNN. For DETR, patches in multiple parts are identified in the early steps, and the reward value defined in Eq. (3) reaches 0.91 after four patches are identified. In contrast, for Faster R-CNN, patches in specific regions are preferentially identified, and the reward value reaches 0.94 after 14 patches are identified. These results indicate that DETR, a transformer-based architecture, recognizes instances in a more compositional manner and with less information compared to the CNN-based architecture Faster R-CNN.

As demonstrated in this experiment, the explanations obtained from VX-CODE facilitate the analysis of object detectors. Further investigations into such analyses using the proposed method are left for future work.

D.12. Additional results for failure cases

In Secs. 4 and 5.3, we discuss explanations for failure cases. Here, we provide additional results. Figure U and Fig. V show the explanations generated by each method for misclassification and mislocalization, respectively. These results demonstrate the effectiveness of the proposed method in analyzing failure cases. For example, as shown in the first row of Fig. U, where the object detector misclassifies the horse as a bird, the proposed method identifies not only the animal's body but also the tree outside the bounding box, indicating that these features (the animal's body and the tree) contribute to the misdetection of the bird. In contrast, other methods do not highlight the tree outside the bounding box and fail to capture the effect of these features. The proposed method is also effective in analyzing mislocalization. For example, as shown in the first row of Fig. V, where the object detector mislocalizes the person by including the chair, the proposed method identifies both the features of the person and the towel covering the chair, indicating that these features contributed to the mislocalization. In contrast, other methods either highlight only one of these features or produce noisy explanations.

Table 11. Results of AUC for insertion (Ins), deletion (Del), overall (OA) metrics. The best values are indicated in bold.

Detector	DETR			Faster R-CNN		
Metric	Ins↑	Del↓	OA↑	Ins↑	Del↓	OA↑
VPS	**.908**	.088	.820	.901	.120	.781
VX-CODE	.900	**.058**	**.842**	**.912**	**.070**	**.842**

D.13. Comparison with VPS

We compare the proposed method with VPS [8], which was recently introduced for object-level foundation models, using DETR and Faster R-CNN. Table 11 presents the insertion (Ins), deletion (Del), and overall (OA) results obtained

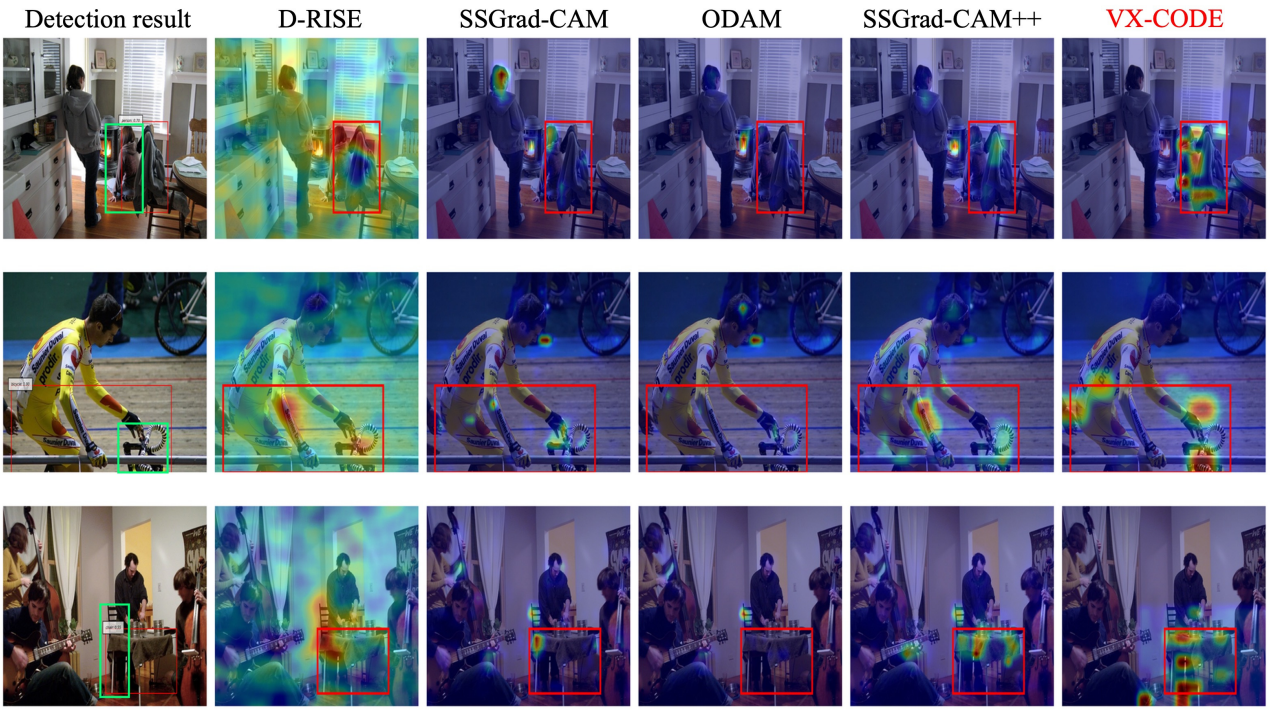


Figure V. Visualization results for mislocalization generated by D-RISE, SSGrad-CAM, ODAM, SSGrad-CAM++, and VX-CODE with patch insertion. In each detection result, the green box represents the ground truth bounding box, while the red box represents the predicted bounding box. The predicted labels, from top to bottom, are *person*, *bicycle*, and *chair*.

by applying both methods to these models. Each value in Tab. 11 represents the average AUC over 500 instances with predicted class scores above 0.7 on MS-COCO. We use VX-CODE with $r = 1$. The proposed method outperforms VPS across almost all metrics, demonstrating its effectiveness.

D.14. Heat maps comparison with multiple instances

A key difference between detector explanations and classification explanations is that the former should be instance-specific. To better demonstrate this property, we additionally provide qualitative results on images containing multiple objects from the same category. Specifically, we consider cases where the detector outputs several bounding boxes of the same class and visualize explanations for a selected detection.

Figure W shows the results. VX-CODE yields instance-focused evidence by capturing multiple complementary cues (e.g., the head and arms) that support the target detection. In contrast, several baselines tend to produce noisy or ambiguous explanations that are harder to interpret in such multi-instance scenes.

Table 12. Quantitative results on RetinaNet (100 detected instances from COCO). The best values are indicated in bold.

Method	Ins ↑	Del ↓	OA ↑
Grad-CAM	0.890	0.420	0.470
Grad-CAM++	0.899	0.407	0.492
DRISE	0.825	0.433	0.392
SSGrad-CAM	0.924	0.369	0.555
ODAM	0.929	0.354	0.575
SSGrad-CAM++	0.932	0.348	0.584
VX-CODE	**0.933**	**0.331**	**0.602**

D.15. Results on a one-stage detector

In the main paper, we focus on two-stage, transformer-based detectors. To further verify that VX-CODE generalizes beyond this setting, we additionally evaluate it on the one-stage detector RetinaNet [31].

Figure X compares the heat map explanations produced by each method. VX-CODE also yields meaningful evidence regions on RetinaNet. Table 12 reports quantitative results computed on 100 detected instances from COCO. Overall, VX-CODE outperforms the baselines, demonstrating its effectiveness on a one-stage detector.

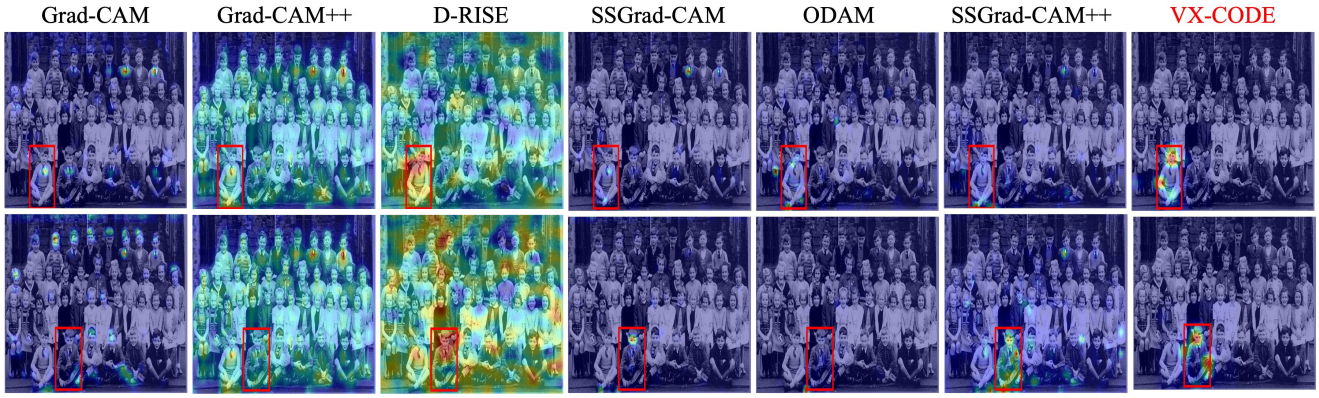


Figure W. Heat maps for an image containing multiple instances of the same category. VX-CODE heat maps are generated via patch insertion.

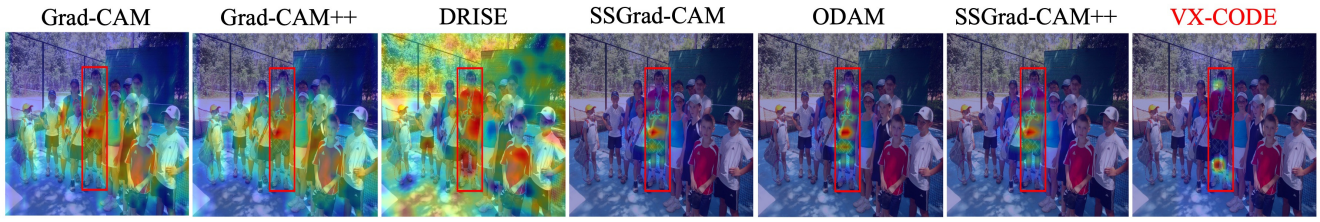


Figure X. Heat map comparisons for RetinaNet. VX-CODE heat maps are generated via patch insertion.

Table 13. Average pairwise interaction score I on COCO with DETR (300 instances). Higher values indicate stronger collective effects between selected patches.

Method	**Interaction score** ↑
Grad-CAM	0.007
Grad-CAM++	0.012
DRISE	0.031
SSGrad-CAM	0.017
ODAM	0.020
SSGrad-CAM++	0.025
VX-CODE	**0.051**

D.16. Additional analysis of collective effects

We further analyze whether the selected patches exhibit collective effects. For two selected patches i and j, we define the pairwise interaction score as

$$I = f(\{i, j\}) - f(\{i\}) - f(\{j\}) + f(\emptyset),$$

where f denotes the reward function. This corresponds to the interaction term in Def. 2 when $N = \{i, j\}$, and measures the synergy obtained by inserting patches i and j jointly into an empty image.

For each method, we aggregate the explanation heat map into patch-level scores, select the top-5 patches, and compute I for all patch pairs. We evaluate this on COCO using DETR over 300 instances. Table 13 reports the average interaction score.

VX-CODE achieves the highest interaction score, indicating that it tends to select patches with stronger collective effects than the competing methods. This result complements the Del/Ins evaluation in the main paper and supports the effectiveness of VX-CODE in identifying jointly important evidence.