Patch-wise Retrieval: A Bag of Practical Techniques for Instance-level Matching

Supplementary Material

S1. Overview

This supplementary material provides additional experiments and implementation details that could not be included in the main paper due to space constraints. Following the structure of the main text, we elaborate on the following aspects:

- **Section S2** explains experimental details on the *Impact of Image Characteristics*, such as instance size, position, and brightness, on retrieval performance.
- **Section S3** expands on the *Investigation of Visual Backbones*, comparing CNN and Transformer encoders and the effect of pretraining dataset size.
- **Section S4** details additional comparisons of *Different Region Selection Strategies*, including Patchify, Sliding Window, and Region Proposal methods.
- **Section S5** provides a deeper dive into our *Comparison with SOTA Methods* in both single-stage and reranking settings.
- **Section S6** includes extended analysis on *Product Quantization*, covering memory efficiency, training feature selection, and compression-performance trade-offs.
- **Section S7** further analyze and report thresholded version and average LocScore.
- **Section S8** shows additional *Qualitative Results* that highlight retrieval accuracy and localization behavior across diverse scenarios.

These materials are intended to support and complement the findings in the main paper, offering deeper insight into our proposed approach and its practical implications.

S2. Impact of Image characteristics

S2.1. Analysis

We investigate the effectiveness of global and local features with respect to three factors: instance size, location, and brightness. The first two factors are directly related to our proposed metrics, LocScore; the last factor is a common challenge in instance retrieval. Figure S1 shows the comparison between local and global features. Overall, we observe that local features generally improve performance, demonstrating robustness under varying image conditions.

As shown in Figure S1a, global features outperform local features when the instance occupies a large portion of the image (rightmost group). However, as the instance size decreases, local features begin to outperform global ones, demonstrating their advantage in small object retrieval. Next, we assess the effect of object location by measuring Euclidean distance between the center of the image and the center of the ground-truth bounding box. As shown in Figure S1b, local features maintain more stable retrieval accuracy as the instance moves away from the center, demonstrating greater robustness to spatial displacement.

Lastly, we analyze the impact of image brightness, quantified as the average L-channel value in the Lab color space. In Figure S1c, both global and local features achieve higher performance on images with medium brightness, while performance drops on very dark or very bright images. Nevertheless, local features consistently outperform global features across all brightness levels, even though the overall trend is similar. For a detailed experimental setup, please refer to Section S2.

S2.2. Experimental Details

We adopt CLIP and DINOv2 as the visual encoder for the evaluation. We compare global and local (L3) features with mAP and LocScore on ILIAS core set.

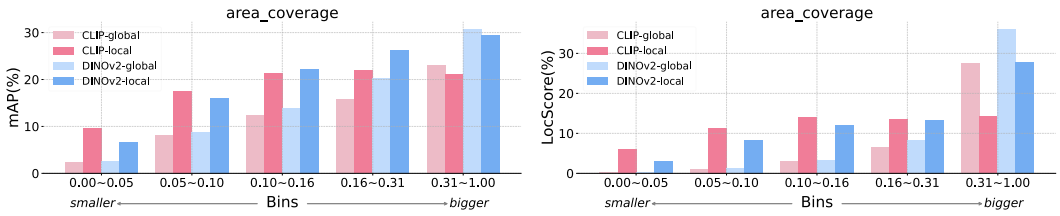
Object bounding box ratio We sort database images with object size by computing the object bounding box area. Then we divide the total object size values equally into 5 bins. For the evaluation, queries with no true positive images in a specific bin are excluded. Global feature works well in big objects but not in small ones.

Object distance from image center We compute L2 distance between the bounding box and the image center and sort them. Then we divide all values equally into 5 bins. Similarly, queries with no true positive images in a specific bin are excluded. Global features work well when the object is centered and not so well when it is not centered.

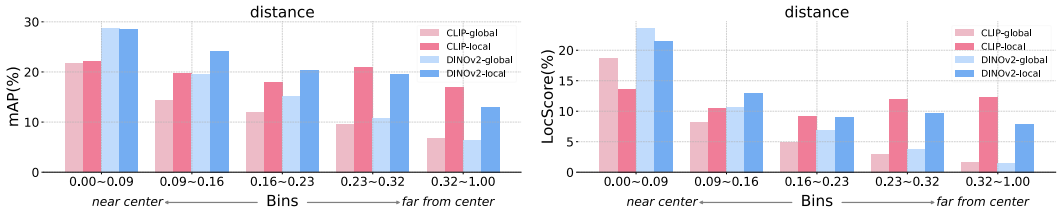
Brightness of image Unlike the previous two experiments, the evaluation benchmark datasets did not exhibit a meaningful distribution with respect to brightness. To address this, we applied tone mapping by scaling the L channel in the Lab color space to 0.2×, 0.6×, 1.0×, 1.4×, and 1.8× of its original value, creating five brightness-adjusted versions of each dataset. We then measured performance on each transformed set.

S3. Investigation of visual backbone

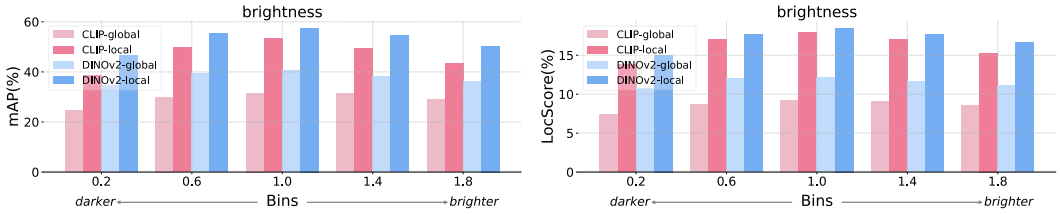
Recently, many visual encoders have been proposed, and we have to choose which visual encoder we use. In this section, we further explore which visual encoders offer the most effective representations for instance retrieval. We analyze a diverse set of models, including both CNN-based (e.g.,



(a) Object bounding box ratio



(b) Object distance from image center



(c) Brightness of image

Figure S1. mAP(%) and LocScore(%) of global and local features in terms of object size, distance from image center, and brightness of image. We measure the performance on ILIAS. Both global and local features show a similar trend. In most cases, local features consistently outperform global features across all settings.

VGG [31], ResNet [7], Inception [33], ConvNext [19]) and Transformer-based (e.g., DINOv2, CLIP, SigLIP [44]) encoders. This analysis aims to provide deeper insights into the architectural factors that contribute to strong instance-level retrieval performance. Note that we extract local features using Patchify. Throughout all experiments in this paper, unless otherwise specified, we use the Large model configuration and an input image resolution of 384×384.

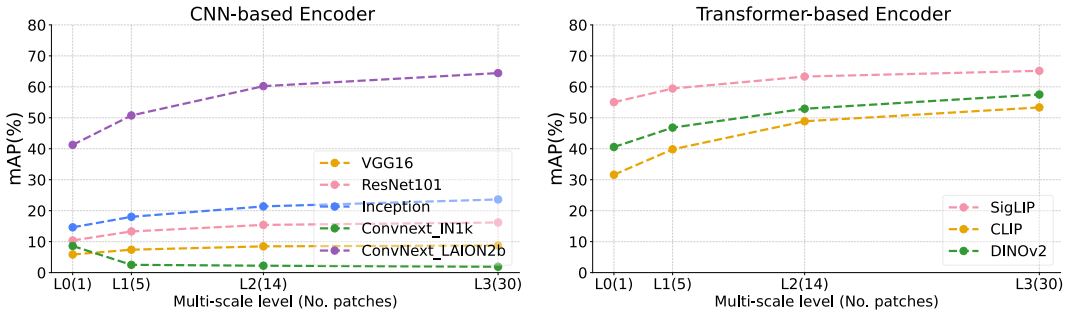
Characteristic of visual encoders We compare instance retrieval performance across encoder models with CNN-based and Transformer-based backbones. As shown in Figure S2, based on our analysis using the ILIAS core dataset, as we use more local features, the performance generally increases. ConvNeXt trained on ImageNet is an exceptional case where the performance decreases. Although ConvNeXt pretrained on LAION-2B shows high performance in CNN-based models, transformer-based models generally achieve higher performance compared to CNN-based models.

Impact of pretraining data size of encoder In terms of the exception case of ConvNeXt pretrained on LAION-2B, we hypothesize that the size of the dataset used during pretraining plays a significant role. As shown in Table 5 and Figure S3, model performance consistently improves with the scale of training data. Regardless of feature dimensionality, increasing the amount of training data improves generalization performance, which in turn enhances instance retrieval performance. This result indicates that representation generalization is influenced by both model capacity and data scale, consistent with findings in prior work [43, 47], and further confirms that this relationship holds for instance retrieval task as well that such an effect also extends to instance retrieval tasks.

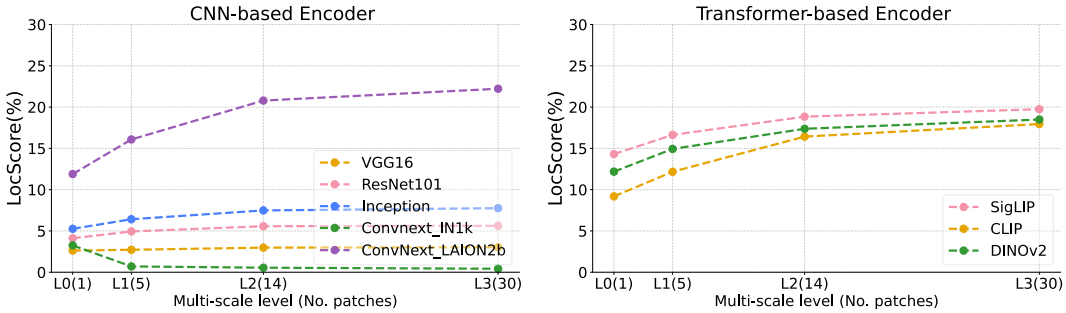
S4. Different region selection strategies

We describe the experimental details for evaluating different region selection strategies. We adopt SigLIP as the visual encoder throughout, and conduct evaluations on both the INSTRE and ILIAS core datasets. The following outlines each strategy illustrated in Figure S4:

Global This method follows the standard global feature approach, where the entire image is passed through the encoder to produce a single feature vector.



(a) mAP

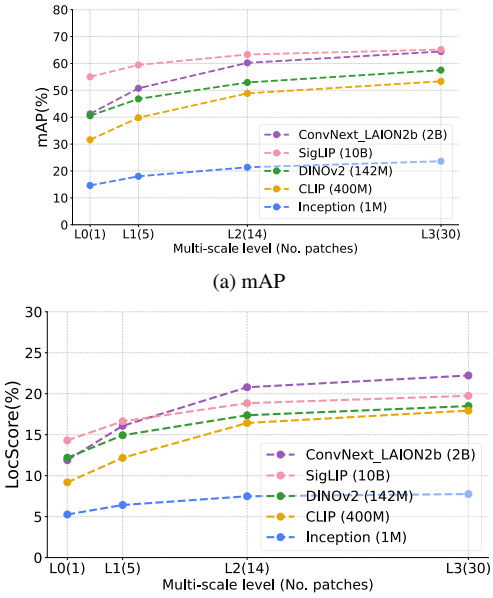


(b) LocScore

Figure S2. Performance comparison between CNN-based backbone and Transformer-based models based on (a) mAP (%) and (b) LocScore (%) metrics on ILIAS. For comparison, we apply the same scale to plots of CNN-based and Transformer-based models. We observe that transformer-based models show higher performance compared to CNN-based models, except for ConvNext_laion2b.

Table 5. Impact of pretraining data size on instance retrieval performance of encoders.

Dataset (Size)	Encoder (Feat. Dim.)	Type	mAP	
			INSTRE	ILIAS
ImageNet-1K (1M)	VGG16 (4096)	global	26.90	5.86
		local	53.56	8.76
	ResNet101 (2048)	global	36.29	10.40
		local	61.38	16.19
	Inception (1536)	global	27.94	14.62
		local	45.82	23.63
	ConvNext (1024)	global	38.97	8.61
		local	56.25	1.90
LVD-142M (142M)	DINOv2 (1024)	global	57.70	40.56
		local	72.54	57.51
LAION-400M (400M)	CLIP (768)	global	73.83	31.60
		local	87.57	53.35
LAION-2B (2B)	ConvNext (768)	global	77.95	41.25
		local	92.87	64.44
WebLI (10B)	SigLIP (1024)	global	78.48	55.03
		local	87.01	65.16



(a) mAP

(b) LocScore

Figure S3. Performance comparison on ILIAS under varying pretraining data sizes. We observe that performance is low if the amount of pre-training data is not sufficient.

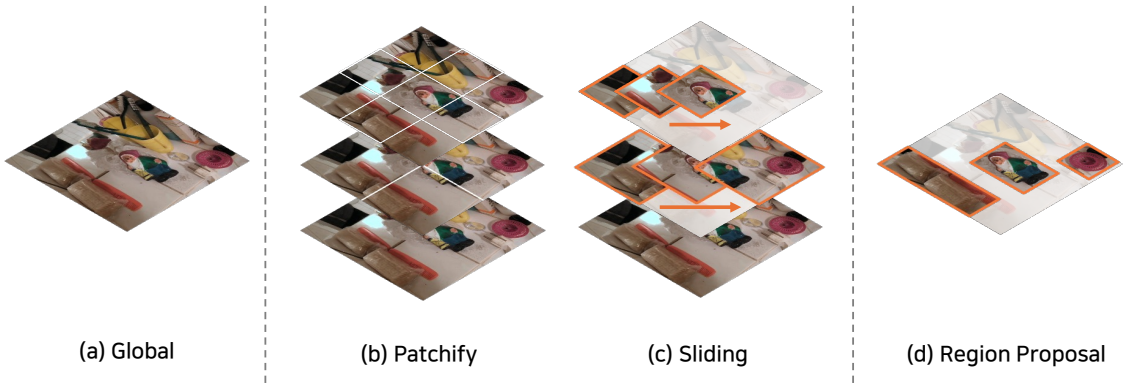


Figure S4. Methods used in analysis. In Global, we use only one embedding. In Patchify, the local features are extracted from the patches of an image. Sliding is similar to Patchify, but the patches have overlap. In Region Proposal, the detector is used to find the informative object in an image and extract the local features by cropping the object.

Patchify This is our proposed method. The image is divided into non-overlapping grid patches according to a predefined configuration (e.g., L0, L1, etc.). Each patch is independently passed through the image encoder to extract local descriptors. L0 corresponds to a single 1×1 patch (i.e., the global setting), L1 includes both 1×1 and 2×2 patches, L2 adds 3×3 patches on top of L1, and L3 further includes 4×4 patches. This cumulative multi-scale design allows for spatial interpretability, as the most similar patch can be traced back to a specific region in the image, revealing which part contributed most to the retrieval.

Sliding To assess whether denser and more exhaustive spatial coverage can improve performance, we implement a sliding window approach on the Patchify setting with two different strides: 0.5 and 0.25 relative to the patch size. Unlike Patchify, the sliding window method introduces overlapping patches for denser spatial coverage. This allows us to assess how performance changes as spatial precision improves due to denser sampling. Unlike naive patchify with a grid-based approach, this method produces overlapping patches, potentially increasing the likelihood of including a well-localized region. The goal is to evaluate whether finer sampling improves retrieval accuracy through better spatial alignment.

Region Proposal We further evaluate a more semantically aware region extraction method using Grounding DINO [18], a recent region proposal network capable of producing high-quality object-level bounding boxes. We extract up to 20 bounding boxes per image with a fixed text prompt "object" and use output bounding boxes as regions for feature extraction. Since these proposals are semantically meaningful and often closely aligned with ground-truth objects, this setting allows us to assess the potential upper bound of retrieval performance when informative and well-localized regions are used. Note that the use of region proposal method for our method is not limited to only Grounding DINO. If any of methods that can produce meaningful and tight region can be used. For example, Segment Anything [13]. Our method can benefit from producing object-aligned patches and informative positives for PQ training, which aligns with our finding that PQ benefits from informative patches.

Table 6. Comparison with SOTA methods on ILIAS core. Patchify achieves competitive performance as a single-stage method and further boosts performance when integrated into reranking pipelines.

Setting	Feature Extraction Method	mAP (%)
Single-stage	SigLIP (Linear Adaptation) [14]	63.96
	Patchify (SigLIP [44])	**65.16**
Two-stage	DINOv2 w/ registers [4]	63.37
	SigLIP [14]	75.61
	Patchify (SigLIP [44])	79.54
	Patchify (SigLIP, Linear Adaptation [14])	**83.52**

S5. Comparison with SOTA Methods

In this section, we describe the implementation details for the experiments conducted in Section 4.3.2. Furthermore we also report retrieval performance is measured on ILIAS core set [14], which was not reported in the main paper.

S5.1. Implementation details

AMES For AMES [32], we adopt its asymmetric transformer-based reranking method. During inference, the top-m images retrieved using global similarity are reranked based on local similarities computed via transformer interaction between the query and database local descriptors. We use the binary distilled variant of AMES with a universal model trained across varying local descriptor counts. The ensemble similarity between global and local features is

computed as a weighted combination, tuned via grid search.

ILIAS Baseline We also benchmark against the reranking strategy proposed in the ILIAS benchmark [14]. This method involves reranking a shortlist of images retrieved via global descriptors using dense feature correlation with ground-truth instance boxes. As this approach depends on densely sampled or region proposal-based local features, it tends to have higher memory and computation overhead.

Hyperparameter Search Since reranking approaches combine global and local similarities, hyperparameter tuning is critical. We perform a grid search over the following parameters to identify the best configuration:

- Number of reranked candidates: `[0, 100, 200, 400, 800, 1000]`
- Global-local balance weight λ: `[0.0, 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 1.0]`
- Temperature scaling γ: `[0.0, 0.05, 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 1.0]`

We report the performance using the best configuration selected from this grid for each method.

S5.2. Retrieval performance on ILIAS core set

We compare against two SOTA methods: DINOv2 w/ registers [4] from AMES [32], and SigLIP with linear adaptation [14] from ILIAS. As summmarized in Table 6, Patchify achieves stronger performance than the fine-tuned baseline on both single-stage and reranking methods. This is aligned with the results on mini-ILIAS, which was reported in the main paper.

S5.3. LocScore on mini-ILIAS

As shown in Table 8, consistent with the earlier results, we observe the same trend: as mAP increases, LocScore increases as well.

S6. Product Quantization

To enable scalable retrieval with local features, we adopt Inverted File with Product Quantization (IVFPQ), a widely used technique that combines coarse clustering with quantization to allow efficient approximate nearest neighbor search with significantly reduced memory and latency overhead. To configure IVFPQ, we perform a grid search over the number of subvectors $m \in \{16, 32, 64\}$ and the number of clusters $nlist \in \{2048, 4096\}$, and select $m = 64$, $nlist = 4096$, and $nbits = 8$ based on retrieval performance.

Table 7 summarizes the impact of increasing patch granularity on database size before and after applying IVFPQ.

Table 7. Impact of multi-scale feature levels on storage size (MB) before and after applying product quantization. CR denotes the compression ratio. We observe that compression is scalable to high memory database.

Level (No. Patches)	INSTRE		ILIAS	
	Non-quant.	Quant (CR)	Non-quant.	Quant (CR)
L0 (1)	103.68	19.77 (5.24)	19.08	18.21 (1.05)
L1 (5)	518.42	27.41 (18.91)	95.41	19.62 (4.86)
L2 (14)	1420	44.61 (31.83)	267.15	22.78 (11.73)
L3 (30)	3040	71.71 (42.39)	572.46	28.41 (20.15)

As the number of patches increases from L0 to L3, uncompressed storage grows rapidly, by more than 29× on INSTRE and 30× on ILIAS. However, with IVFPQ, this increase is drastically reduced to just 3.6× and 1.6×, respectively. These results confirm that IVFPQ effectively scales to high-resolution patch representations, enabling practical deployment of patch-based retrieval with minimal memory overhead.

We investigate how the choice of training features affects PQ effectiveness. As summarized in Table 4, the highest performance is achieved when PQ is trained with features extracted from ground-truth bounding boxes (G.T.), which provide strong semantic alignment with the target instances. In contrast, using patches from intermediate configurations (e.g., L1 and L2) results in lower performance, possibly due to noisy or irrelevant background content in those features. Interestingly, PQ trained with global features (L0) outperforms L1 and L2, suggesting that lower-resolution but semantically focused representations may be more beneficial than ambiguous fine-grained features.

Qualitative examples in Figure S6 support this observation: G.T.-trained PQ accurately retrieves the correct instance under challenging conditions such as occlusion, scale change, and viewpoint variation, while PQ trained on L2 fails to distinguish between visually similar but incorrect instances. These findings highlight the importance of using informative training features for PQ optimization and offer valuable guidance for future work on compressed local feature retrieval.

S7. LocScore

This section provides extended analyses of the localization behavior measured by LocScore that were omitted from the main paper due to space limits. While the main paper introduced the metric and its motivation, here we investigate how different IoU criteria affect retrieval behavior and how localization quality varies across feature levels and model variants.

Figures S7 and S8 visualize the thresholded LocScore(δ) for multiple IoU thresholds on ILIAS-core and INSTRE. These results reveal that different models degrade at differ-

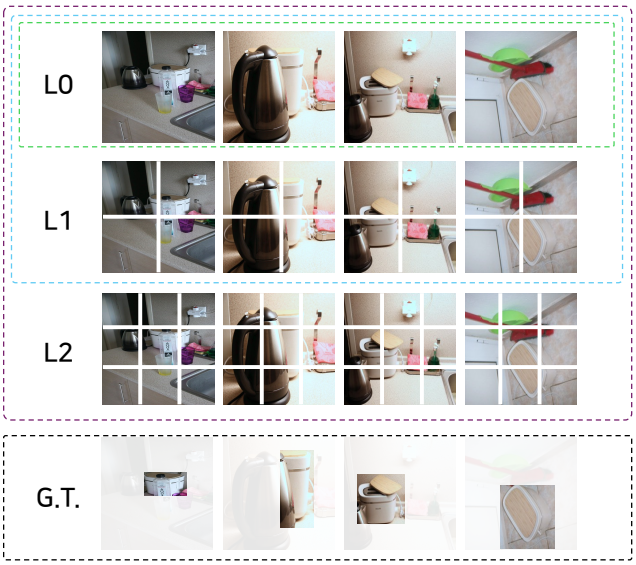


Figure S5. Images used to train PQ indices in each Patchify level and G.T. We can observe that features from G.T. are related to the instance, directly.

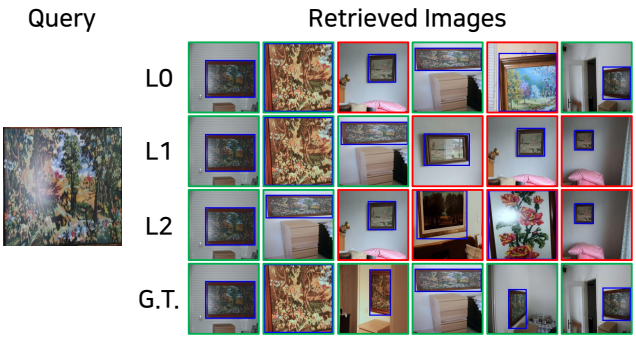


Figure S6. Qualitative retrieval results using different PQ training features at L3. PQ trained on G.T. finds the correct instance compared to others. Thus, we also consider the informative features when training PQ.

ent rates as the spatial criterion becomes stricter. For example, methods that rely on coarse global features show a rapid decline beyond $\delta = 0.3$, whereas patch-based retrieval remains more stable up to $\delta = 0.5$. This indicates that localization robustness is tightly linked to the granularity of visual evidence used by the model.

Figures 9 and 10 report the LocScore averaged across thresholds, providing a summary measure that reflects overall spatial reliability. This aggregated score smooths out threshold-specific fluctuations and highlights consistent performance trends that are not fully visible from single-threshold evaluations. In particular, models with strong mid-level representations maintain competitive scores across a wide range of δ, suggesting that localization accuracy is distributed across multiple feature layers rather than dominated by a single level.

Figure 5 illustrates how different IoU criteria impact qualitative retrieval outcomes. Unlike the continuous variant, which reflects gradual changes in spatial alignment, thresholded LocScore(δ) exposes failure modes where retrieved patches partially overlap the target but fail to satisfy a strict IoU requirement. These cases highlight the sensitivity of certain models to small spatial displacements. The contrast between the two forms therefore helps to diagnose whether a model's failure stems from ranking errors (low IoU overall) or boundary precision (IoU slightly below δ).

Together, these supplementary results provide a more complete understanding of the spatial behavior of retrieval systems. They show how localization precision evolves under increasingly strict criteria, how robustness varies across feature hierarchies, and how different formulations of LocScore reveal distinct types of spatial failure.

Qualitative results of LocScore variants Figure S11a compares different localization strategies for the same query—Global, Patchify, Patchify with a sliding window (stride 0.25), and the Region Proposal variant. Improved localization accuracy leads to higher IoU, which subsequently improves both AP and LocScore. The Region Proposal method exhibits the best localization quality and obtains the highest scores.

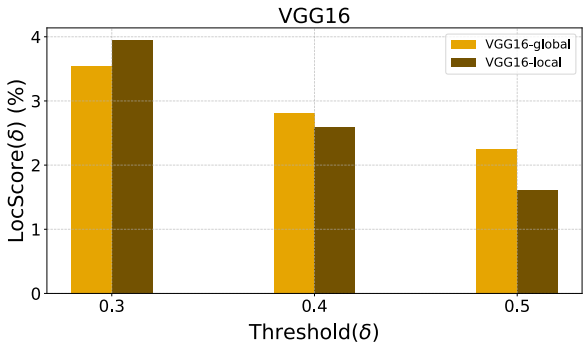
Figure S11b contrasts the continuous LocScore with its thresholded versions LocScore(δ) at different IoU thresholds. Although the continuous LocScore differs only by about 0.05 between the two rows, the thresholded scores change dramatically as δ increases, illustrating how sensitive LocScore(δ) is to the definition of acceptable localization quality.

Figure S11c mirrors the example in the main paper, highlighting the relationship between AP and LocScore. Even when AP is relatively high, LocScore can remain low if the predicted regions have poor overlap with the ground-truth boxes, showing that improving LocScore requires not only strong ranking (high AP) but also accurate localization with high IoU.

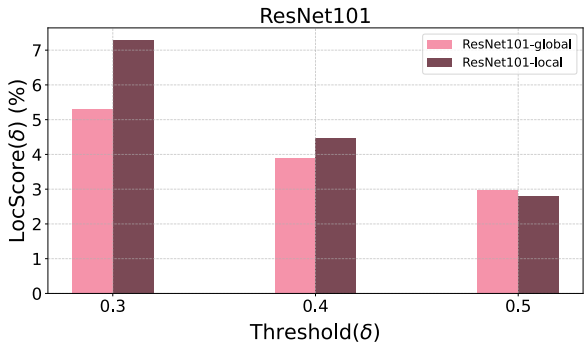
Limitations. LocScore is tightly coupled with AP because it multiplies rank-based contribution by IoU, so the score can be overly driven by the localization quality of a few top-ranked positives while good localization at lower ranks has little influence. The thresholded variant partly alleviates this by enforcing a minimum IoU, but it introduces subjectivity since the notion of "good localization" depends on a human-chosen threshold. To reduce this ambiguity, we additionally report a mean version of LocScore and present both the continuous and thresholded variants side by side. We added this discussion in the supplementary.

Model	Level	mAP@1k	LocScore (cont.)	mLocScore	LocScore (δ)
SigLIP	**L0 (Global)**	0.2041	6.59	5.60	δ=0.3 : 7.55 δ=0.4 : 5.38 δ=0.5 : 3.86
SigLIP	**L1**	0.2968	10.08	8.92	δ=0.3 : 12.56 δ=0.4 : 8.37 δ=0.5 : 5.83
SigLIP	**L2**	0.5027	18.11	16.21	δ=0.3 : 24.78 δ=0.4 : 15.43 δ=0.5 : 8.41
SigLIP	**L3**	0.2608	8.81	7.55	δ=0.3 : 10.58 δ=0.4 : 7.32 δ=0.5 : 4.76
SigLIP†	**L0 (Global)**	0.3386	9.59	7.40	δ=0.3 : 10.39 δ=0.4 : 7.11 δ=0.5 : 4.72
SigLIP†	**L1**	0.3655	11.08	8.99	δ=0.3 : 12.69 δ=0.4 : 8.50 δ=0.5 : 5.78
SigLIP†	**L2**	0.3924	12.29	9.92	δ=0.3 : 14.22 δ=0.4 : 9.46 δ=0.5 : 6.08
SigLIP†	**L3**	0.4048	12.82	10.33	δ=0.3 : 15.04 δ=0.4 : 9.77 δ=0.5 : 6.19
DINOv3	**L0 (Global)**	0.2180	8.02	7.40	δ=0.3 : 9.55 δ=0.4 : 7.26 δ=0.5 : 5.39
DINOv3	**L1**	0.3147	11.48	10.25	δ=0.3 : 14.22 δ=0.4 : 9.68 δ=0.5 : 6.86
DINOv3	**L2**	0.3954	15.71	15.31	δ=0.3 : 22.33 δ=0.4 : 14.53 δ=0.5 : 9.08
DINOv3	**L3**	0.4298	17.05	16.44	δ=0.3 : 24.72 δ=0.4 : 15.32 δ=0.5 : 9.29

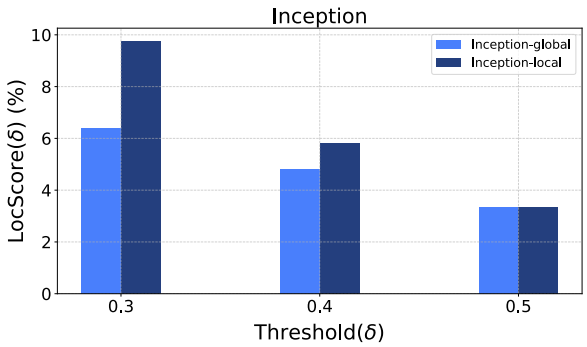
Table 8. Quantitative results on mini-ILIAS with mAP@1k and LocScore variants. A dagger (†) denotes the linear adaptation baseline from ILIAS; δ is the IoU threshold for the thresholded LocScore.



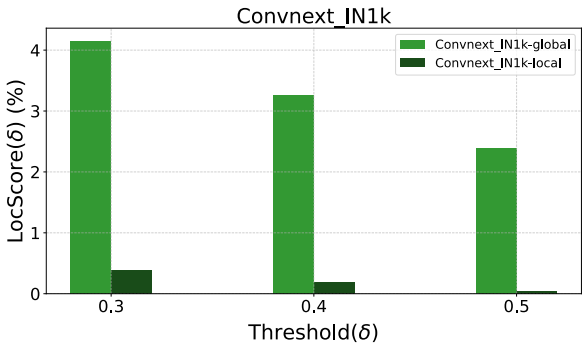
(a) VGG16



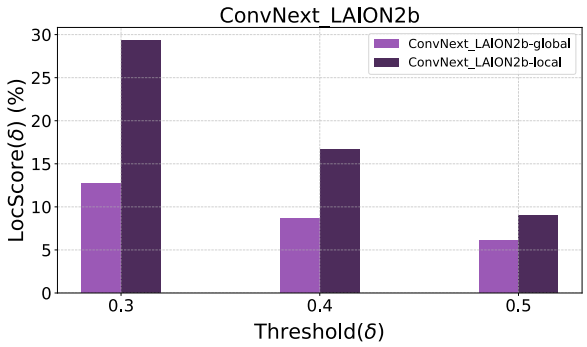
(b) ResNet



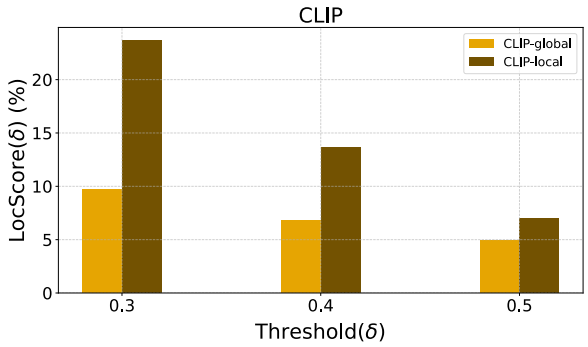
(c) Inception



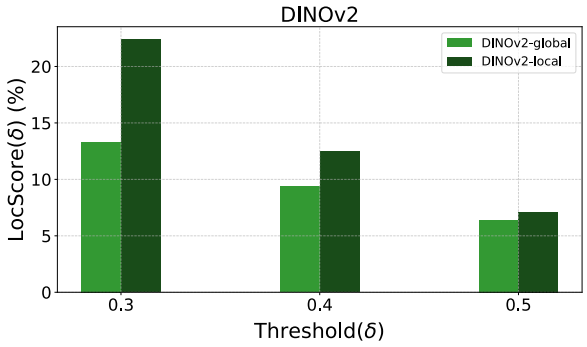
(d) ConvNext-IN1k



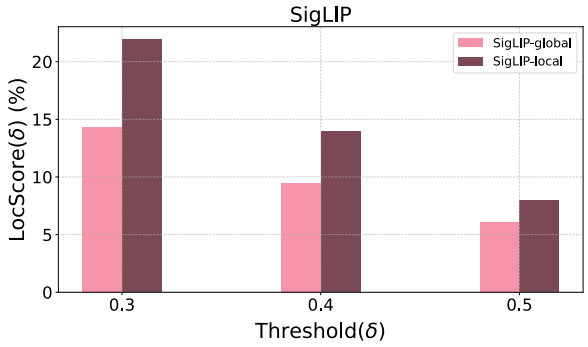
(e) ConvNext-LAION2b



(f) CLIP

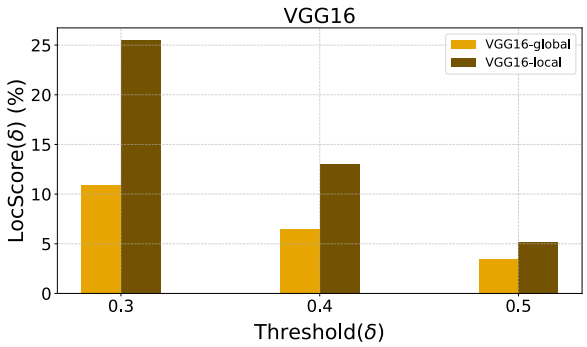


(g) DINOv2

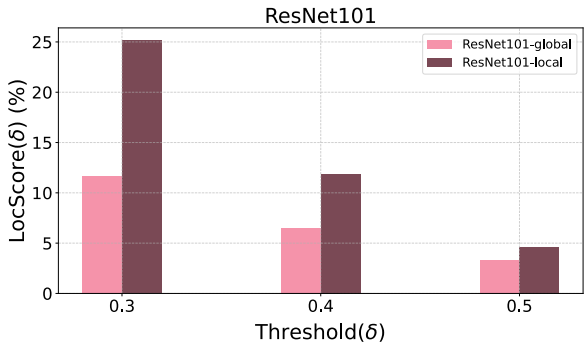


(h) SigLIP

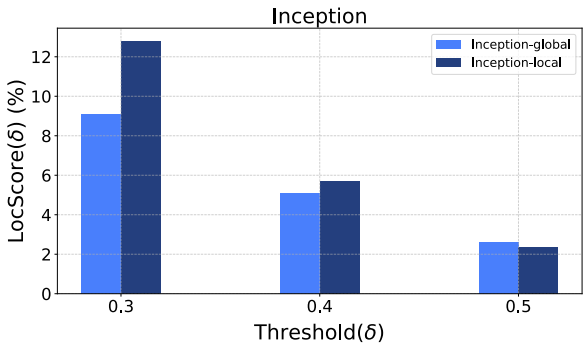
Figure S7. LocScore(δ) visualizations across threshold $\delta = 0.3, 0.4, 0.5$ of CNN-based and Transformer-based models on ILIAS.



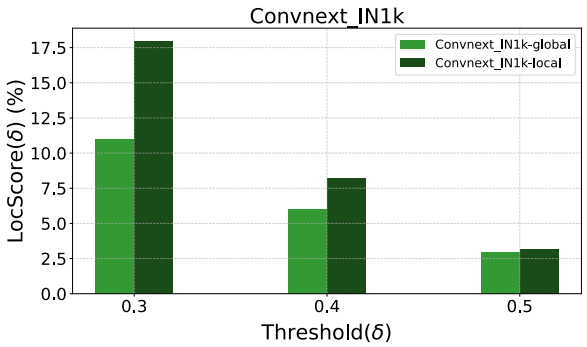
(a) VGG16



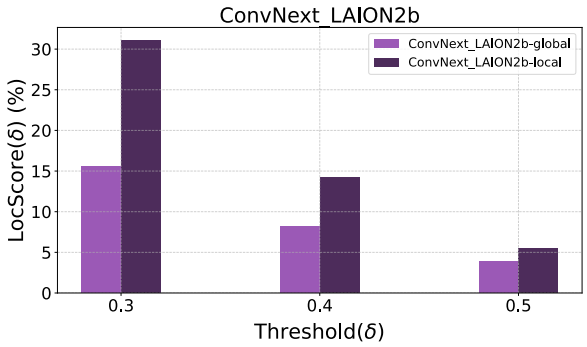
(b) ResNet



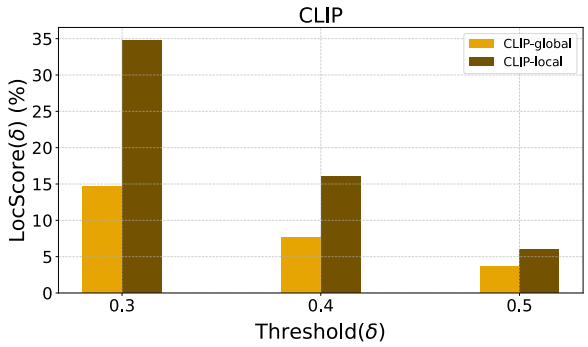
(c) Inception



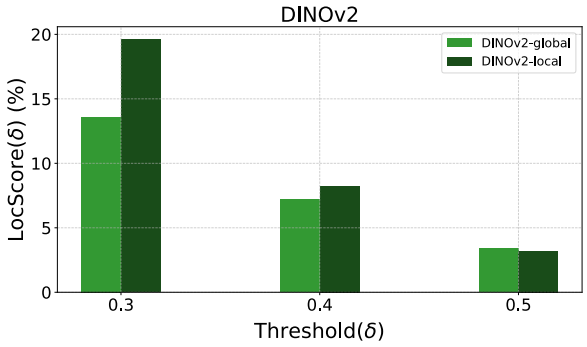
(d) ConvNext-IN1k



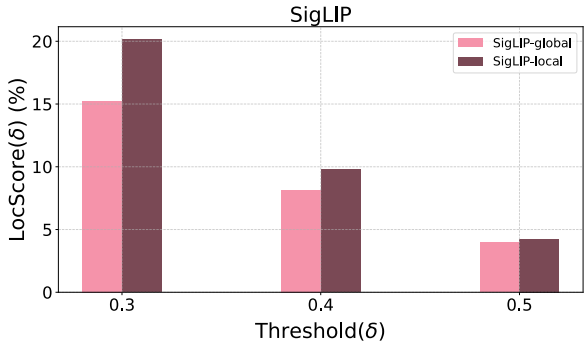
(e) ConvNext-LAION2b



(f) CLIP



(g) DINOv2



(h) SigLIP

Figure S8. LocScore(δ) visualizations across threshold $\delta = 0.3, 0.4, 0.5$ of CNN-based and Transformer-based models on INSTRE.

		INSTRE				ILIAS			
model	**Level**	**mAP**	**LocScore**	**mLocScore**	**LocScore(δ)**	**mAP**	**LocScore**	**mLocScore**	**LocScore(δ)**
					δ=0.3 : 10.928				δ=0.3 : 3.540
VGG16	**L0**	26.690	8.842	6.960	δ=0.4 : 6.526	5.864	2.625	2.866	δ=0.4 : 2.814
					δ=0.5 : 3.426				δ=0.5 : 2.245
					δ=0.3 : 16.400				δ=0.3 : 3.359
	L1	39.859	13.697	9.453	δ=0.4 : 8.295	7.399	2.718	2.468	δ=0.4 : 2.396
					δ=0.5 : 3.665				δ=0.5 : 1.649
					δ=0.3 : 24.852				δ=0.3 : 3.852
	L2	51.186	19.189	14.331	δ=0.4 : 12.865	8.485	2.978	2.728	δ=0.4 : 2.614
					δ=0.5 : 5.277				δ=0.5 : 1.719
					δ=0.3 : 25.461				δ=0.3 : 3.941
	L3	53.562	19.813	14.561	δ=0.4 : 12.999	8.764	3.030	2.716	δ=0.4 : 2.595
					δ=0.5 : 5.223				δ=0.5 : 1.613
					δ=0.3 : 11.643				δ=0.3 : 5.296
ResNet101	**L0**	36.288	10.763	7.147	δ=0.4 : 6.505	10.396	4.111	4.064	δ=0.4 : 3.908
					δ=0.5 : 3.293				δ=0.5 : 2.989
					δ=0.3 : 17.410				δ=0.3 : 6.110
	L1	48.757	15.892	9.725	δ=0.4 : 8.259	13.300	4.941	4.367	δ=0.4 : 4.158
					δ=0.5 : 3.504				δ=0.5 : 2.832
					δ=0.3 : 25.269				δ=0.3 : 7.309
	L2	58.588	21.058	14.150	δ=0.4 : 12.284	15.410	5.573	4.969	δ=0.4 : 4.640
					δ=0.5 : 4.896				δ=0.5 : 2.959
					δ=0.3 : 25.138				δ=0.3 : 7.276
	L3	61.382	21.558	13.893	δ=0.4 : 11.894	16.193	5.625	4.853	δ=0.4 : 4.467
					δ=0.5 : 4.646				δ=0.5 : 2.815
					δ=0.3 : 9.066				δ=0.3 : 6.410
Inception	**L0**	27.943	8.435	5.585	δ=0.4 : 5.105	14.624	5.260	4.859	δ=0.4 : 4.812
					δ=0.5 : 2.584				δ=0.5 : 3.355
					δ=0.3 : 11.091				δ=0.3 : 7.807
	L1	34.592	10.866	6.325	δ=0.4 : 5.464	18.026	6.418	5.545	δ=0.4 : 5.275
					δ=0.5 : 2.422				δ=0.5 : 3.552
					δ=0.3 : 13.787				δ=0.3 : 9.831
	L2	42.077	13.755	7.676	δ=0.4 : 6.513	21.385	7.484	6.557	δ=0.4 : 6.187
					δ=0.5 : 2.728				δ=0.5 : 3.652
					δ=0.3 : 12.808				δ=0.3 : 9.769
	L3	45.822	14.112	6.952	δ=0.4 : 5.698	23.631	7.757	6.310	δ=0.4 : 5.810
					δ=0.5 : 2.350				δ=0.5 : 3.350
					δ=0.3 : 10.981				δ=0.3 : 4.153
ConvNext_IN1k	**L0**	38.972	11.006	6.649	δ=0.4 : 5.993	8.606	3.253	3.266	δ=0.4 : 3.253
					δ=0.5 : 2.974				δ=0.5 : 2.394
					δ=0.3 : 13.879				δ=0.3 : 0.820
	L1	47.535	14.549	7.727	δ=0.4 : 6.535	2.517	0.705	0.505	δ=0.4 : 0.451
					δ=0.5 : 2.768				δ=0.5 : 0.243
					δ=0.3 : 18.403				δ=0.3 : 0.658
	L2	54.045	17.939	10.173	δ=0.4 : 8.663	2.232	0.549	0.355	δ=0.4 : 0.312
					δ=0.5 : 3.452				δ=0.5 : 0.096
					δ=0.3 : 17.984				δ=0.3 : 0.392
	L3	56.254	18.163	9.793	δ=0.4 : 8.203	1.902	0.421	0.206	δ=0.4 : 0.187
					δ=0.5 : 3.193				δ=0.5 : 0.041

Table 9. Quantitative results with mAP and LocScore under various backbones.

		INSTRE				ILIAS			
model	**Level**	**mAP**	**LocScore**	**mLocScore**	**LocScore(δ)**	**mAP**	**LocScore**	**mLocScore**	**LocScore(δ)**
					δ=0.3 : 15.531				δ=0.3 : 12.752
ConvNext_LAION2b	**L0**	77.949	19.005	9.171	δ=0.4 : 8.135	41.253	11.886	9.156	δ=0.4 : 8.672
					δ=0.5 : 3.847				δ=0.5 : 6.043
					δ=0.3 : 21.397				δ=0.3 : 19.025
	L1	86.710	24.872	11.977	δ=0.4 : 10.228	50.754	16.078	12.302	δ=0.4 : 11.002
					δ=0.5 : 4.306				δ=0.5 : 6.880
					δ=0.3 : 30.010				δ=0.3 : 27.039
	L2	91.524	30.159	16.453	δ=0.4 : 13.943	60.215	20.789	17.296	δ=0.4 : 16.130
					δ=0.5 : 5.407				δ=0.5 : 8.719
					δ=0.3 : 31.109				δ=0.3 : 29.363
	L3	92.870	30.936	16.953	δ=0.4 : 14.263	64.441	22.221	18.333	δ=0.4 : 16.688
					δ=0.5 : 5.488				δ=0.5 : 8.948
					δ=0.3 : 13.593				δ=0.3 : 13.332
DINOv2	**L0**	57.701	15.074	8.063	δ=0.4 : 7.185	40.557	12.181	9.698	δ=0.4 : 9.374
					δ=0.5 : 3.411				δ=0.5 : 6.386
					δ=0.3 : 16.078				δ=0.3 : 16.526
	L1	64.621	18.738	8.781	δ=0.4 : 7.285	46.825	14.932	11.476	δ=0.4 : 10.594
					δ=0.5 : 2.980				δ=0.5 : 7.309
					δ=0.3 : 20.583				δ=0.3 : 20.722
	L2	69.978	22.183	11.065	δ=0.4 : 9.056	52.918	17.371	13.460	δ=0.4 : 12.330
					δ=0.5 : 3.556				δ=0.5 : 7.328
					δ=0.3 : 19.611				δ=0.3 : 22.401
	L3	72.543	22.216	10.343	δ=0.4 : 8.233	57.515	18.492	13.999	δ=0.4 : 12.485
					δ=0.5 : 3.184				δ=0.5 : 7.110
					δ=0.3 : 14.594				δ=0.3 : 9.696
OpenCLIP	**L0**	73.837	18.107	8.638	δ=0.4 : 7.659	31.598	9.183	7.143	δ=0.4 : 6.827
					δ=0.5 : 3.660				δ=0.5 : 4.904
					δ=0.3 : 21.196				δ=0.3 : 14.501
	L1	78.812	23.034	11.454	δ=0.4 : 9.428	39.819	12.170	9.302	δ=0.4 : 8.278
					δ=0.5 : 3.737				δ=0.5 : 5.129
					δ=0.3 : 33.733				δ=0.3 : 21.976
	L2	85.594	29.556	18.428	δ=0.4 : 15.674	48.878	16.425	14.039	δ=0.4 : 13.256
					δ=0.5 : 5.877				δ=0.5 : 6.885
					δ=0.3 : 34.804				δ=0.3 : 23.706
	L3	87.565	30.370	18.936	δ=0.4 : 16.053	53.351	17.945	14.767	δ=0.4 : 13.598
					δ=0.5 : 5.953				δ=0.5 : 6.998
					δ=0.3 : 15.261				δ=0.3 : 14.345
SigLIP	**L0**	78.483	18.918	9.120	δ=0.4 : 8.135	55.029	14.305	9.969	δ=0.4 : 9.469
					δ=0.5 : 3.965				δ=0.5 : 6.095
					δ=0.3 : 17.815				δ=0.3 : 17.504
	L1	83.056	21.719	10.335	δ=0.4 : 9.037	59.449	16.631	11.911	δ=0.4 : 11.195
					δ=0.5 : 4.154				δ=0.5 : 7.033
					δ=0.3 : 20.255				δ=0.3 : 20.688
	L2	85.965	23.888	11.526	δ=0.4 : 9.963	63.319	18.842	13.888	δ=0.4 : 13.368
					δ=0.5 : 4.360				δ=0.5 : 7.607
					δ=0.3 : 20.151				δ=0.3 : 21.933
	L3	87.015	24.285	11.397	δ=0.4 : 9.777	65.165	19.745	14.613	δ=0.4 : 13.933
					δ=0.5 : 4.264				δ=0.5 : 7.974

Table 10. Quantitative results with mAP and LocScore under various backbones.

S8. Qualitative Results

Figure S9. Qualitative results of Global and Patchify on ILIAS

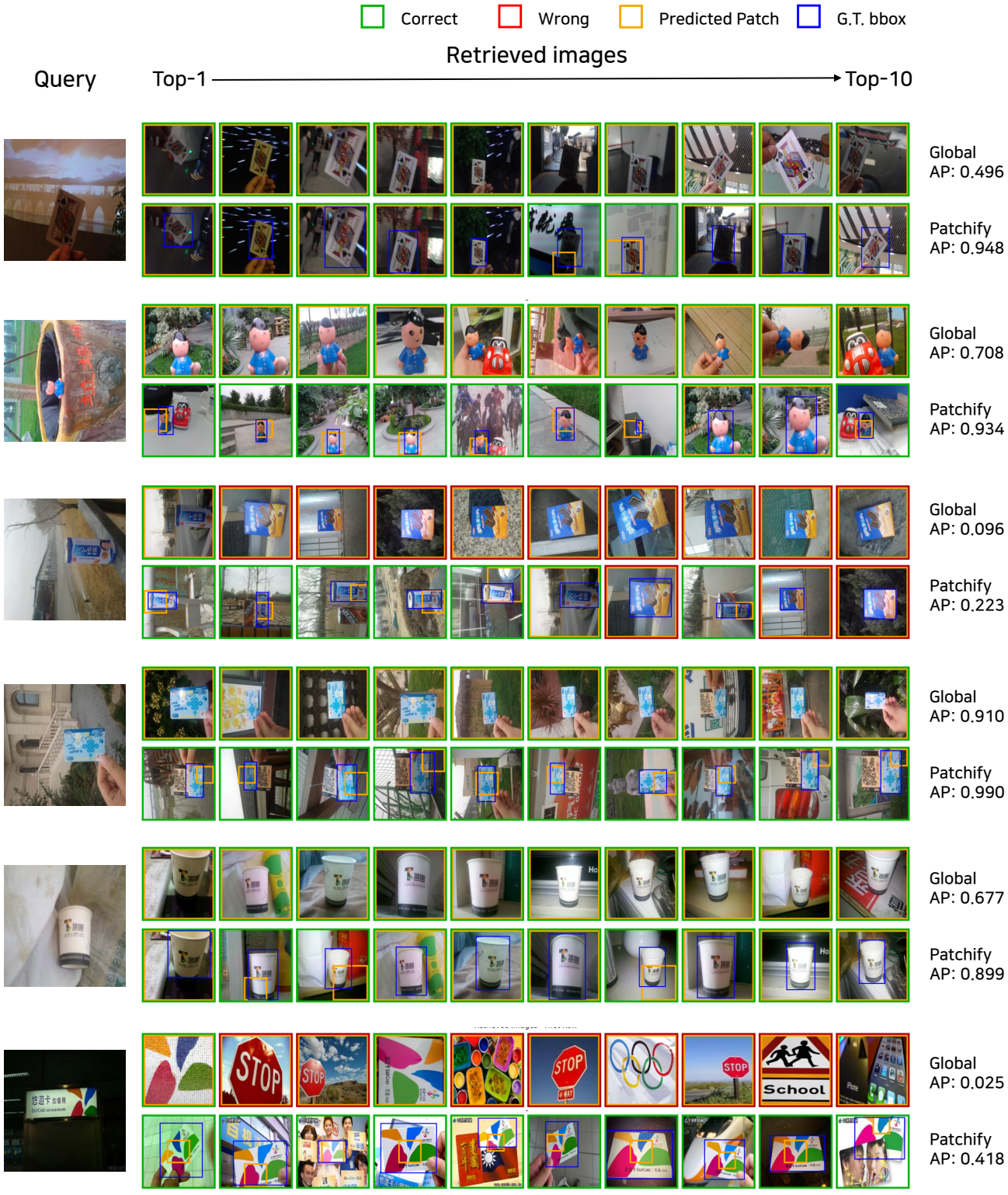
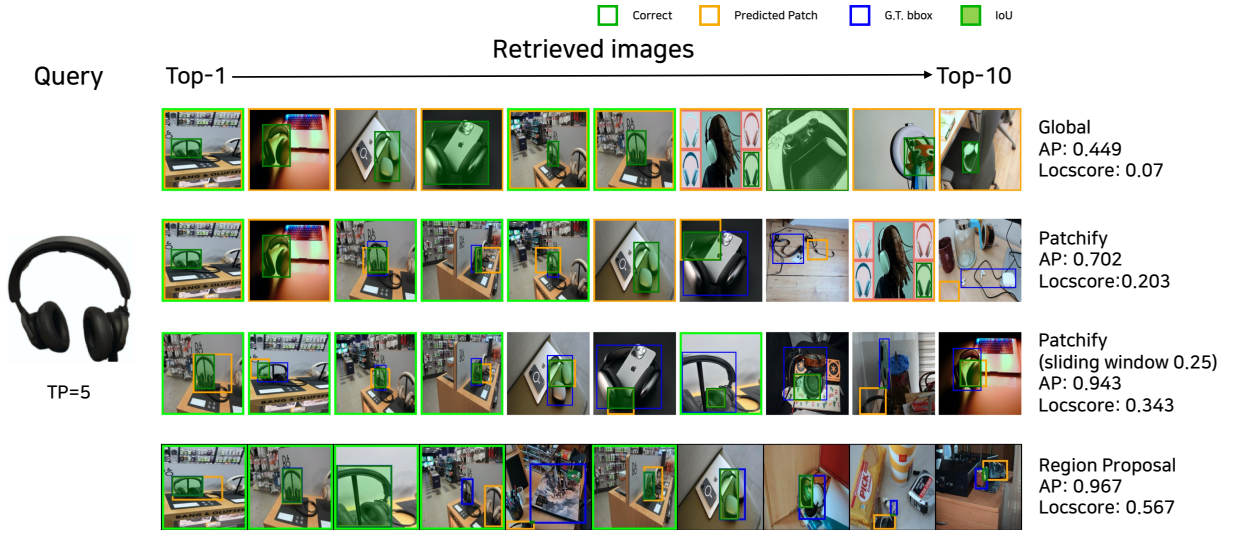
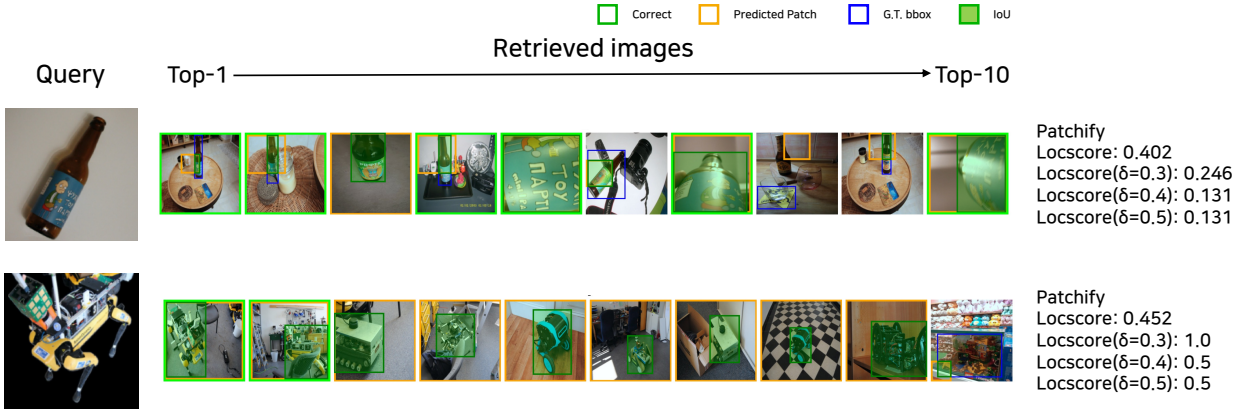


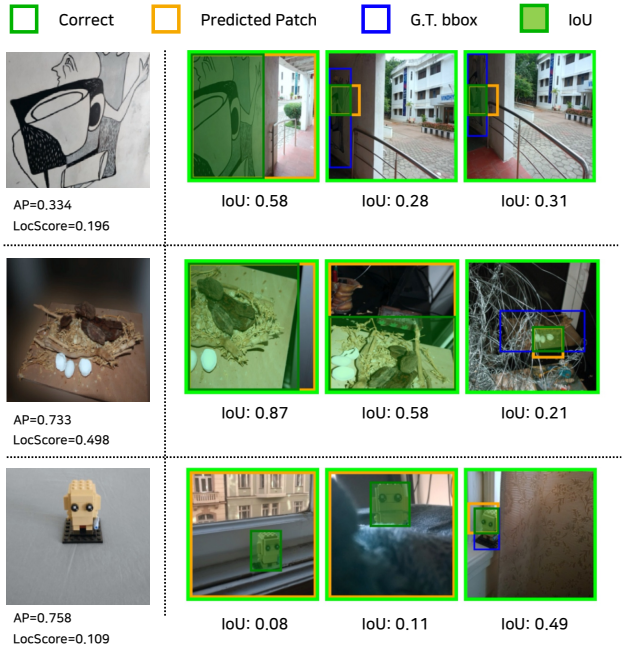
Figure S10. Qualitative results of Global and Patchify on INSTRE



(a) Comparison of LocScore by method



(b) Comparison of LocScore and LocScore(δ)



(c) Comparison of AP and LocScore

Figure S11. **Qualitative results of LocScore variants.**