Patch-wise Retrieval: An Interpretable Instance-Level Image Matching

Supplementary Material

S1. Overview

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

- Section S2 describes the trends in related research, explains how our work is positioned among them, and highlights the differences from prior approaches.
- Section S3 presents an analysis of the effectiveness of global and local features in instance retrieval with respect to three factors: instance size, location, and brightness. The results provide insights into why local features are more robust than global features under various conditions.
- Section S4 analyzes the impact of model architecture and pretraining data scale on retrieval performance.
- Section S5 provides a comprehensive view of how different region-level sampling strategies affect retrieval outcomes, and examines whether the simple Patchify method remains competitive compared to strategies with stronger spatial alignment.
- Section S6 explains the implementation details of the experiments from Section 3.2.2 that were not described in the main paper.
- Section S7 investigates how to achieve scalability while using patch-wise features, analyzing database size, performance, and the role of informative features. We show that the choice of training features for PQ has a significant impact on performance; in particular, features well-aligned with target objects lead to better compression and retrieval accuracy, underscoring the importance of informative patch selection during index construction.
- Section S8 provides an extended version of the conclusion, containing discussions that could not be included in the main paper due to space limitations.
- Section S9 covers the thresholded version of LocScore and the mean LocScore that were not included in the main paper, and analyzes the corresponding results.
- Section S10 presents qualitative results across a wide variety of instances.

Together, these sections offer a deeper and more comprehensive view of our framework, clarifying design choices, providing diagnostic analyses, and demonstrating both the scalability and interpretability benefits of Patchify in instance-level image retrieval.

S2. Related work

Instance-Level Image Retrieval Methods Early studies in instance-level image retrieval relied on hand-crafted descriptors such as SIFT [19] and SURF [1], which extract local keypoints and compute image similarity via Bag-of-Words models. With the advent of deep learning, the field shifted toward learning feature representations using deep convolutional backbones like VGG, Inception, and ResNet. Global image descriptors extracted from these networks, particularly with pooling schemes such as GeM [24], became the dominant approach.

Recent advances in large-scale data-driven learning have further led to the development of highly transferable visual representations. Models trained on vast and diverse datasets, such as CLIP [12, 25] and DINO [2], are capable of extracting general-purpose image features that perform well across a wide range of downstream tasks, including classification [10, 40], segmentation [16, 26], and image retrieval [21, 23]. These self-supervised and vision-language models provide strong global representations that enable scalable and efficient retrieval systems. However, these global descriptors often struggle with fine-grained instance matching, especially when the target object is small, occluded, or appears off-center. This limitation arises from their lack of spatial precision, which becomes critical in instance-level retrieval scenarios where spatial alignment between query and retrieved content is essential.

To overcome the spatial limitations of global descriptors, recent instance retrieval pipelines have adopted a two-stage reranking strategy [15, 30]. In these methods, an initial ranking is produced using global descriptors, followed by a reranking stage that compares sets of local features between top-ranked images. Local features are typically extracted as hundreds of dense patches or region proposals per image, and fine-grained similarity is computed across all pairs. While this improves localization and robustness, it introduces significant computational overhead. Each query requires comparing thousands of patch-level embeddings, and storing such dense local features across a database is memory-intensive.

Our method, Patchify, addresses the above limitations through a simple yet effective patch-based retrieval framework. We divide each image into a small number of non-overlapping grid patches and embed them independently using frozen visual encoders. Compared to reranking-based approaches, Patchify requires only a handful of patch features per image, drastically reducing both computation and memory. To further improve scalability, we apply Product

Quantization (PQ) [14] to compress patch embeddings to a level comparable with global descriptors. Despite its simplicity, Patchify achieves performance on par with state-of-the-art global and reranking methods, and can further benefit from reranking extensions. This demonstrates that careful patch-level design choices can lead to highly efficient and interpretable retrieval without compromising accuracy.

Explainability via Localization Cues As deep learning systems grow in capability and complexity, interpretability has emerged as a critical component for building trustworthy and transparent AI. The field of explainable AI (XAI) has gained increasing attention, particularly in computer vision, where spatial interpretability enables users to understand where a model focuses during prediction. Foundational approaches such as Class Activation Maps (CAM) [39] and Weakly Supervised Object Localization (WSOL) [3, 4] have laid the groundwork, and recent efforts have expanded toward analyzing the internal structures of large-scale models [6, 8, 9].

In contrast to prior work, our method provides interpretability *by design* in the context of instance-level retrieval. Our method, Patchify, allows us to trace which part of the image contributes most to the retrieval, unlike traditional global descriptors that lack spatial grounding. To quantify this interpretability, we propose LocScore, a localization-aware metric that jointly considers retrieval accuracy and spatial alignment with the ground truth. This provides a unified measure of retrieval quality and interpretability, allowing for diagnostic analysis of model behavior.

S3. Impact of Image characteristics

S3.1. 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.

S3.2. Implementation 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.

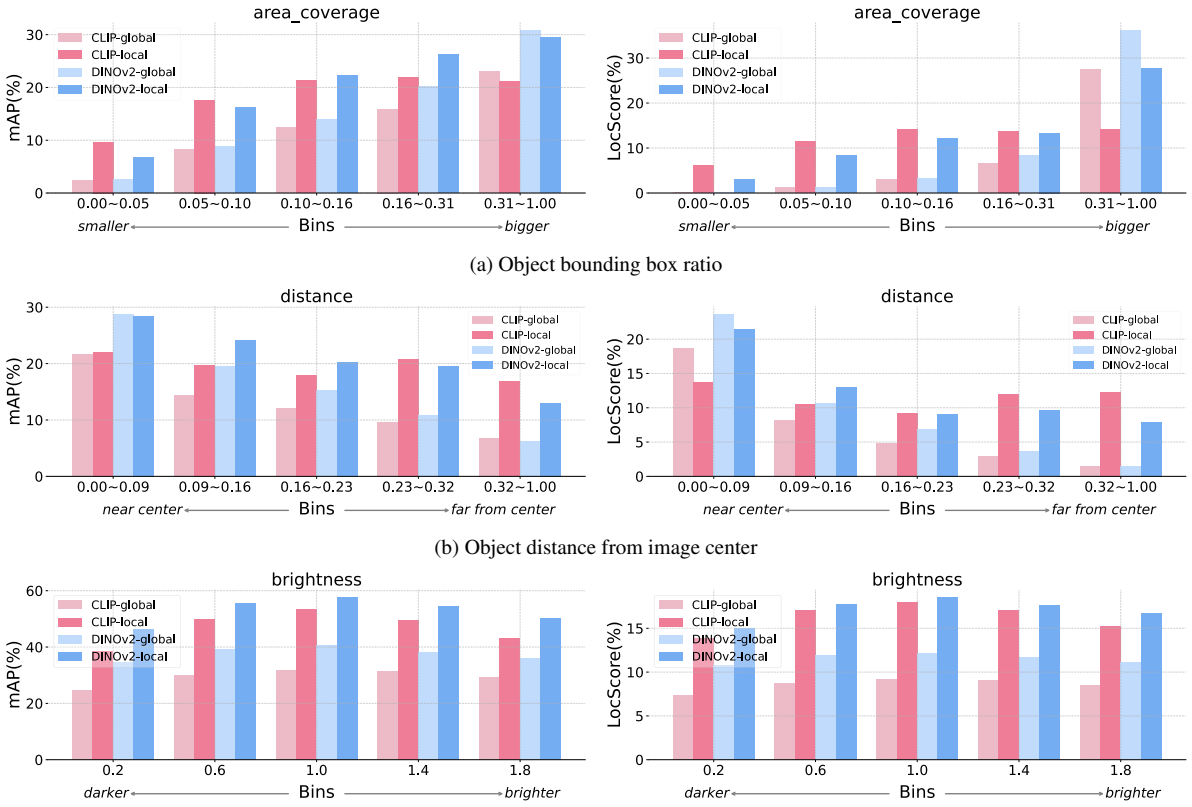
Object bounding box ratio We sort database images with object size by computing the object bounding box area. Then we divide the total object size values equally into 5 bins. For the evaluation, queries with no true positive images in a specific bin are excluded. Global feature works well in big objects 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.

S4. Investigation of visual backbone

To better understand the impact of model architecture and pretraining data scale on retrieval performance, we conduct a comparative analysis across various visual encoders. Our results indicate that Transformer-based models such as DINOv2, CLIP, and SigLIP consistently outperform CNN-based counterparts like ResNet and Inception, demonstrating the effectiveness of modern architectural designs in both retrieval accuracy and localization quality. Notably, ConvNeXt pretrained on LAION-2B, despite being a CNN, achieves performance on par with Transformer-based models. This observation underscores the importance of large-scale pretraining, which significantly enhances the generalization ability of learned features across architectures.



(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.

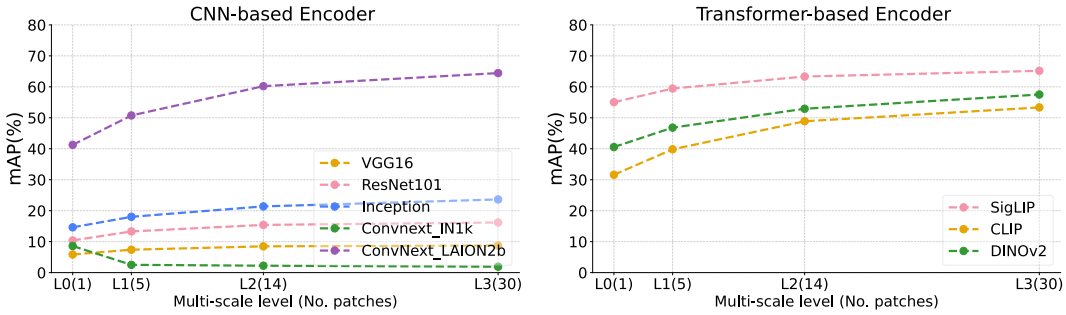
Recently, many visual encoders have been proposed, and we have to choose which visual encoder we use. In this section, we further explore which visual encoders offer the most effective representations for instance retrieval. We analyze a diverse set of models, including both CNN-based (e.g., VGG [28], ResNet [11], Inception [31], ConvNext [18]) and Transformer-based (e.g., DINOv2, CLIP, SigLIP [37]) 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.

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 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.

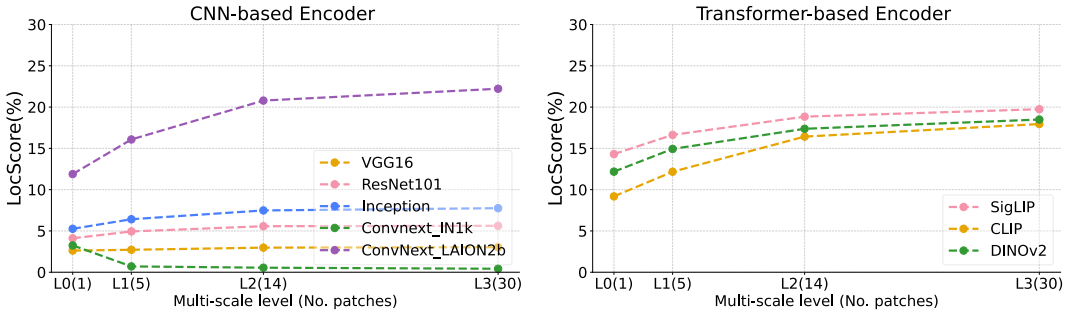
Impact of pretraining data size of encoder In terms of the exception case of ConvNeXt pretrained on LAION-2B, we hypothesize that the size of the dataset used during pretraining plays a significant role. As shown in Table 4 and Figure S3, model performance consistently improves with the scale of training 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 [36, 41], and further confirms that this relationship holds for instance retrieval task as well that such an effect also extends to instance retrieval tasks.

S5. Different region selection strategies

A simple and structured grid-based extraction strategy, as used in our Patchify method, proved highly effective, es-



(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 4. Impact of pretraining data size on instance retrieval performance of encoders.

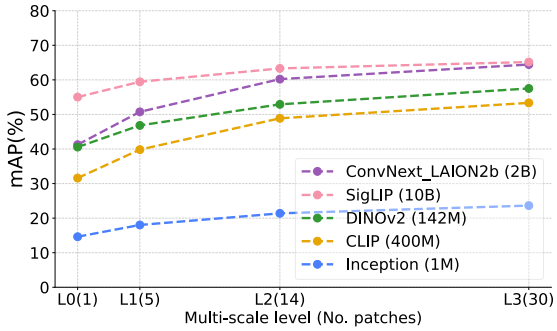
Dataset (Size)	Encoder (Feat. Dim.)	Type	mAP	
			INSTRE	ILIAS
ImageNet-1K (1M)	VGG16 (4096)	global	26.90	5.86
		local	53.56	8.76
	ResNet101 (2048)	global	36.29	10.40
		local	61.38	16.19
	Inception (1536)	global	27.94	14.62
		local	45.82	23.63
	ConvNext (1024)	global	38.97	8.61
		local	56.25	1.90
LVD-142M (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

pecially when combined with strong vision transformer encoders such as SigLIP. This raises an important question: how does such a simple approach compare to more elaborate region selection methods?

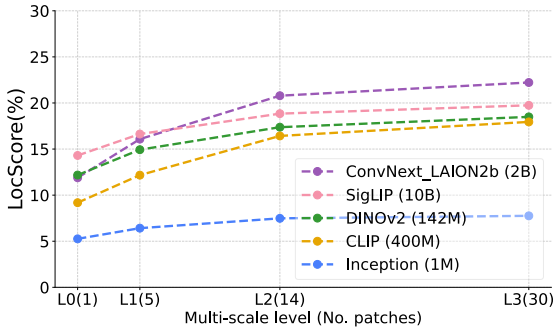
To answer this, we design a series of experiments comparing Patchify with alternative region sampling strategies, restricting our analysis to the SigLIP encoder, which showed strong performance in earlier evaluations. All methods are evaluated on both the INSTRE and ILIAS datasets, and a visual illustration is provided in Figure S4. We consider the following strategies:

Global The standard approach where the entire image is passed through the encoder to produce a single feature vector.

Patchify Our proposed method. The image is divided into non-overlapping grid patches according to a predefined multi-scale configuration (e.g., L0, L1, L2, L3). L0 corresponds to a single 1×1 patch (the global setting), L1 includes both 1×1 and 2×2 patches, L2 adds 3×3 patches, and L3 further includes 4×4 patches. Each patch is independently passed through the encoder to extract local descriptors. This cumulative multi-scale design enables spatial interpretability, as the most similar patch can be traced



(a) mAP



(b) LocScore

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

to a specific image region.

Sliding windows A denser sampling variant of Patchify with overlapping patches, using strides of 0.5 and 0.25 relative to patch size instead of a stride of 1. This increases spatial coverage and the likelihood of capturing well-localized patches, allowing us to test whether finer sampling improves retrieval accuracy and spatial alignment.

Region proposals A semantically guided method using Grounding DINO [17] to generate up to 20 high-quality object-level bounding boxes per image. These regions are cropped and encoded, offering an upper bound on performance when informative and well-localized patches are available.

Our comparison in Table 2 shows that sliding window methods outperform Patchify in both mAP and LocScore, with the 0.25 stride variant achieving the best results. This suggests that reducing the stride improves the likelihood of capturing accurate patches. Region proposal methods, while potentially stronger in spatial alignment, require additional computational cost and detection models. Across strategies, higher mAP generally coincides with higher LocScore, indicating that models retrieving correct images also tend to localize target objects more precisely. This positive correlation holds for both CNN and Transformer architectures, where increasing patch granularity consistently enhances performance.

S6. Comparison with SOTA Methods

In this section, we describe the implementation details for the experiments conducted in Section 3.2.2. All methods are evaluated using SigLIP as the visual encoder, and retrieval performance is measured on both INSTRE [33] and ILIAS [15] benchmarks.

AMES For AMES [30], 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 [15]. 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.

S7. Applying Compression to Local Features

A well-known limitation of local feature-based retrieval methods is their high memory usage, as they require storing a large number of descriptors per image. Unlike conventional approaches, Patchify uses only a small number of structured patch features per image, significantly reducing the memory and computational burden. To further enhance

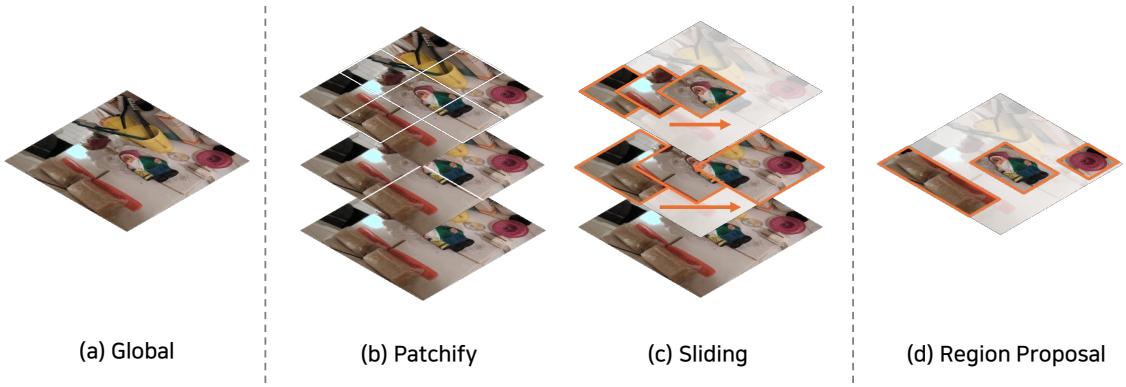


Figure S4. Methods used in analysis. In Global, we use only one embedding. In Patchify, the local features are extracted from the patches of an image. Sliding is similar to Patchify, but the patches have overlap. In Region Proposal, the detector is used to find the informative object in an image and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

Table 5. Comparison of mAP (%) performance under different multi-scale levels and PQ usage.

Level	Patchify	PQ	INSTRE		ILIAS	
			mAP	LocScore	mAP	LocScore
L0	✗	✗	78.48	18.92	55.03	14.31
	✗	✓	81.99	19.48	54.97	14.30
L1	✓	✗	83.06	21.72	59.45	16.63
	✓	✓	86.20	22.92	57.37	15.65
L2	✓	✗	85.97	23.89	63.32	18.84
	✓	✓	88.20	25.00	59.28	17.00
L3	✓	✗	87.01	24.29	65.16	19.75
	✓	✓	88.52	25.08	59.96	17.33

scalability for large-scale retrieval, we apply Inverted File with Product Quantization (IVFPQ) [14].

Database Size Table 7 shows the compression ratio when applying IVFPQ to the database. As we increase the level of patches, the memory consumption grows rapidly because the required memory is proportional to the square of the patch level. However, when applying IVFPQ to the database, the increased amount of memory is alleviated. For example, in the INSTRE dataset, the difference in memory between L0 and L3 before IVFPQ is about 29.3 times, but after IVFPQ, the difference is 3.6 times.

Performance comparison Table 5 presents the retrieval performance before and after applying PQ. As expected, compressing feature representations with PQ can lead to a slight performance drop due to quantization. This trend is observed on the ILIAS benchmark, where the data scale and difficulty are substantially higher. Interestingly, on the smaller and less challenging INSTRE benchmark, PQ even yields performance improvements, potentially due to noise reduction or enhanced centroid generalization. Despite compression, local features with PQ still outperform global features in both benchmarks. Together with the memory statistics in Table 7, these results indicate that local characteristics and PQ offer complementary benefits: delivering strong retrieval accuracy with substantially reduced storage.

Table 6. Comparison of mAP (%) performance using different training features for PQ.

Training Feature	L0 (1)	L1 (5)	L2 (14)	L3 (30)	G.T. (1)
mAP (%)	60.19	55.80	55.53	53.13	**65.07**

Training with Informative Features Product Quantization (PQ) requires a training phase to learn cluster centroids, and the choice of training features can significantly impact retrieval performance. To explore this, we compare PQ trained with features at different patch levels (L0, L1, L2, L3) and with ground-truth-aligned features cropped from instance bounding boxes. As summarized in Table 6, the highest performance is achieved when PQ is trained on ground-truth features, emphasizing the value of using semantically informative representations. Interestingly, features from intermediate patch levels (L1 and L2) perform worse than L0, likely due to the inclusion of background noise or irrelevant content. These results suggest that tighter spatial alignment during PQ training plays a crucial role, and highlight the importance of effective region selection strategies. This highlights a previously underexplored yet critical aspect of PQ training and may inform more robust design choices in future PQ-based retrieval pipelines.

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. 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 6, 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.

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)

S8. Extended version of Conclusion

In this work, we introduced Patchify, a simple yet effective patch-wise retrieval framework that achieves strong performance while maintaining both scalability and interpretability. By decomposing each database image into a small set of structured, multi-scale patches, Patchify enables spatially-aware matching without requiring fine-tuning or region proposals. Our experiments demonstrate that Patchify consistently outperforms global feature baselines, adapts well across a variety of backbones, and integrates seamlessly into existing reranking pipelines. To evaluate the spatial quality of retrievals, we proposed LocScore, a localization-aware metric that captures not only retrieval accuracy but also the alignment between retrieved regions and target objects. This metric provides valuable diagnostic insight into how retrieval decisions are made and helps bridge perfor-

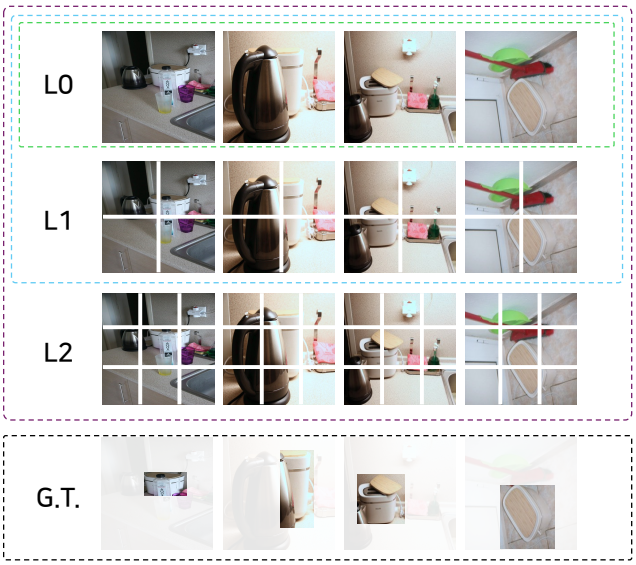


Figure S5. Images used to train PQ indices in each Patchify level and G.T. We can observe that features from G.T. are related to the instance, directly.

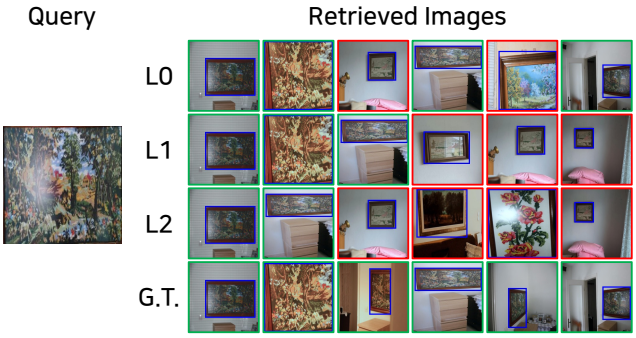


Figure S6. Qualitative retrieval results using different PQ training features at L3. PQ trained on G.T. finds the correct instance compared to others. Thus, we also consider the informative features when training PQ.

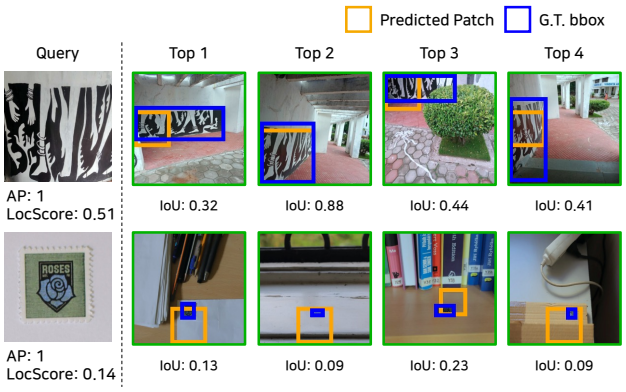


Figure S7. **Comparison of AP and LocScore for evaluating retrieval quality.** Both query examples achieve perfect AP (*i.e.*, all ground-truth images are retrieved), yet their LocScores differ significantly due to localization accuracy. This highlights how LocScore complements AP by incorporating spatial alignment quality.

mance evaluation with model interpretability. Furthermore, we showed that Patchify is highly scalable through the use of Product Quantization (PQ). Our analysis revealed that the choice of training features for PQ has a significant impact on performance. In particular, features that are well-aligned with target objects lead to better compression and retrieval accuracy, highlighting the importance of informative patch selection during index construction. Overall, Patchify offers a practical and interpretable design for instance-level image retrieval, enabling efficient matching with minimal computation and memory overhead. Our findings point toward promising future directions for retrieval systems that combine spatial precision, scalability, and transparency.

S9. LocScore

In this section, we present additional analyses of the LocScore metric that could not be included in the main paper due to space constraints. Specifically, we report results on the thresholded variants of LocScore as illustrated in Figure S8 and S9, the corresponding averaged metric across thresholds, as shown in Table 8 and 9. These results provide a more detailed understanding of how localization quality varies under different criteria and offer further insight into the spatial precision of retrieval systems.

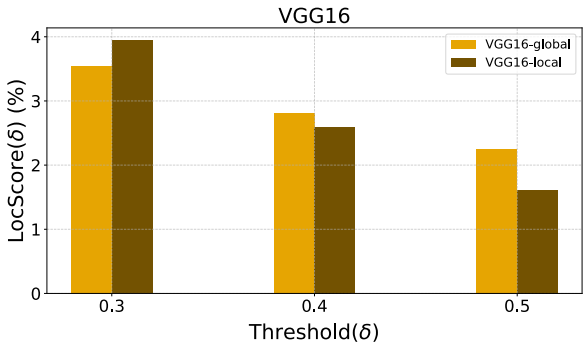
As illustrated in Figure 3, the thresholded variant of LocScore differs from the original continuous formulation by enforcing a binary success criterion based on a fixed IoU threshold δ. While the original LocScore weights retrievals proportionally to their IoU values, the thresholded version considers a retrieved patch correct only if its IoU with the ground-truth box exceeds δ. For example, in Figure 3, Top-3 is counted as correct for $\delta = 0.3$ but not for $\delta = 0.5$, leading to a decrease in LocScore as the threshold increases. This binary formulation provides a more stringent evaluation of spatial alignment, complementing the continuous score by emphasizing higher-precision localization.

Thresholded vs. Continuous LocScore Our proposed LocScore metric comes in two variants: the thresholded LocScore(δ) and the continuous version, mLocScore. These two forms offer different perspectives on evaluating spatial alignment in retrieval results.

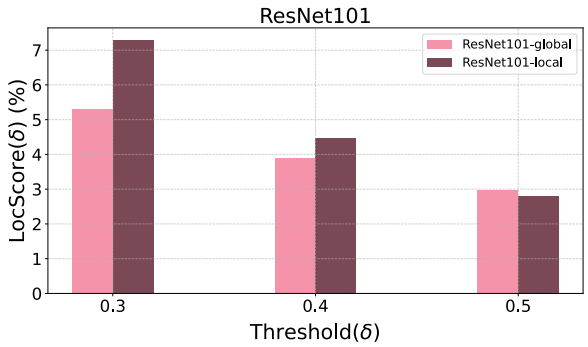
Thresholded LocScore(δ) evaluates whether the predicted patch sufficiently overlaps with the ground-truth bounding box based on a predefined IoU threshold δ. Only retrieved results with IoU $\geq \delta$ are counted as correct, making the score binary and stricter. For example, as the threshold increases from 0.3 to 0.5, fewer retrievals are considered valid, leading to a lower overall LocScore. This allows for clear pass/fail decisions but may ignore partial matches that are still spatially relevant.

In contrast, mLocScore provides a continuous assessment by averaging the IoU values of the retrieved patches. This formulation captures fine-grained differences in spatial alignment and offers a more nuanced view of localization quality. Unlike the binary variant, mLocScore reflects varying degrees of correctness rather than a hard cutoff.

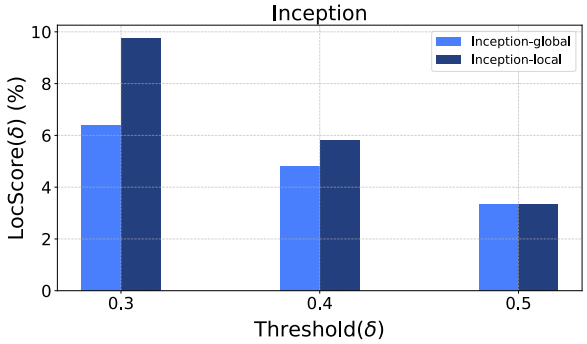
The difference in behavior is evident in the experimental results. Increasing δ in LocScore(δ) leads to a sharp drop in scores, particularly at $\delta = 0.5$, while mLocScore exhibits smoother variations and preserves relative differences between models and feature levels. Therefore, thresholded LocScore is more appropriate when a strict spatial correctness criterion is needed, while mLocScore is preferable for capturing overall localization trends.



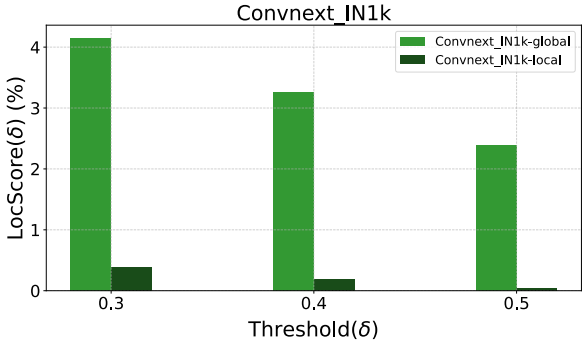
(a) VGG16



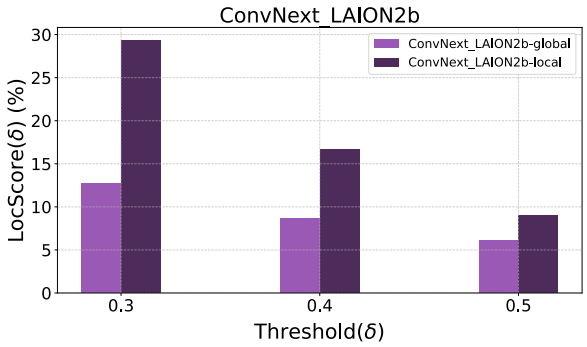
(b) ResNet



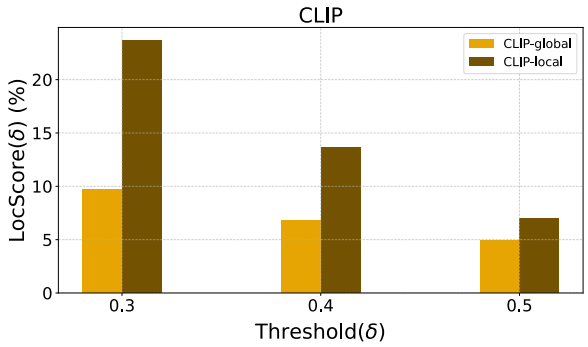
(c) Inception



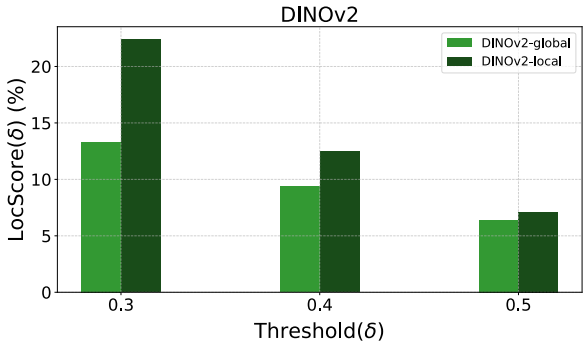
(d) ConvNext-IN1k



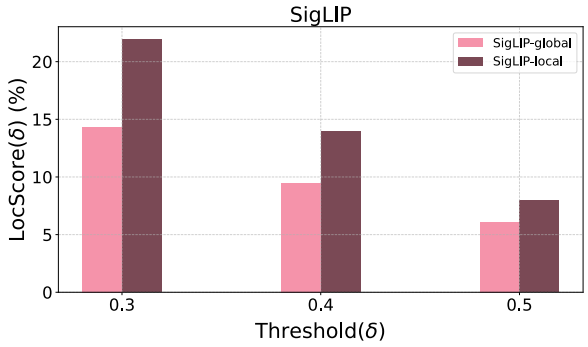
(e) ConvNext-LAION2b



(f) CLIP

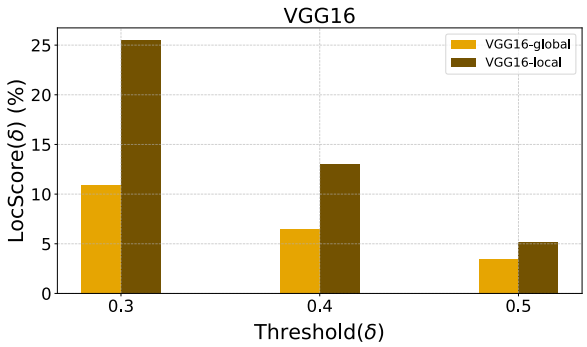


(g) DINOv2

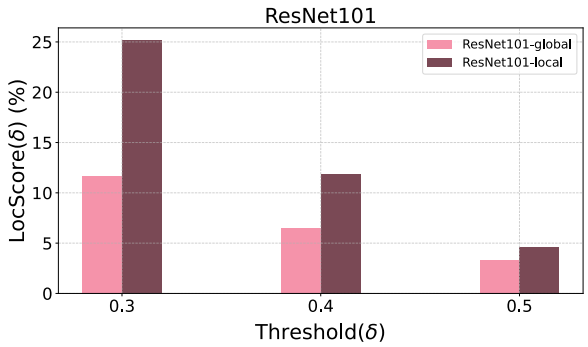


(h) SigLIP

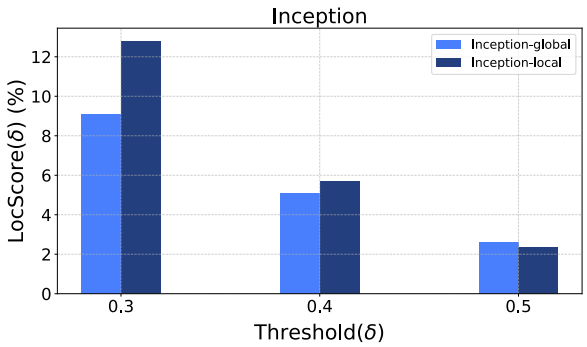
Figure S8. LocScore(δ) visualizations across threshold $\delta = 0.3, 0.4, 0.5$ of CNN-based and Transformer-based models on ILIAS.



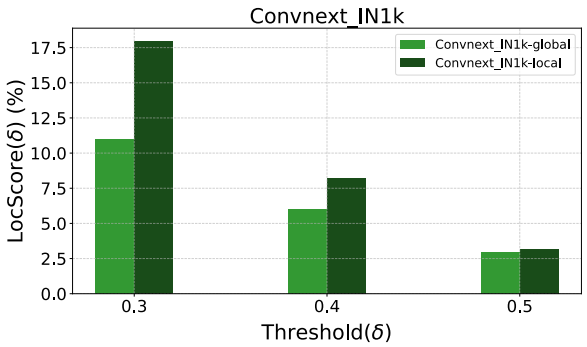
(a) VGG16



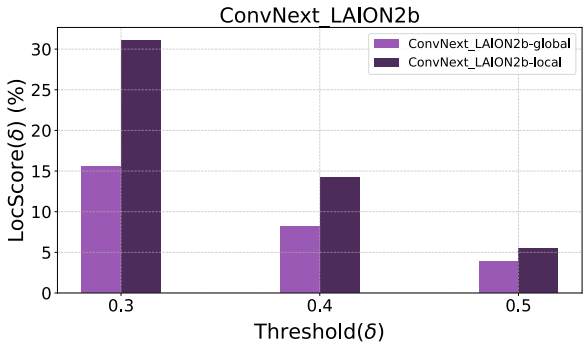
(b) ResNet



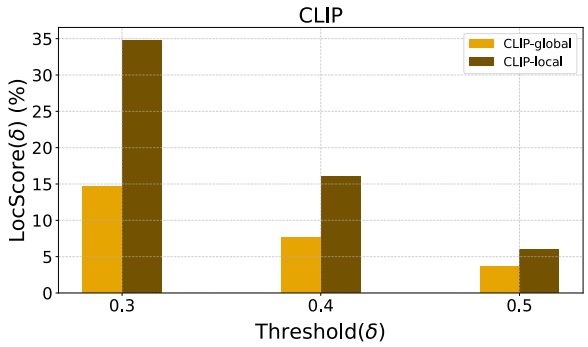
(c) Inception



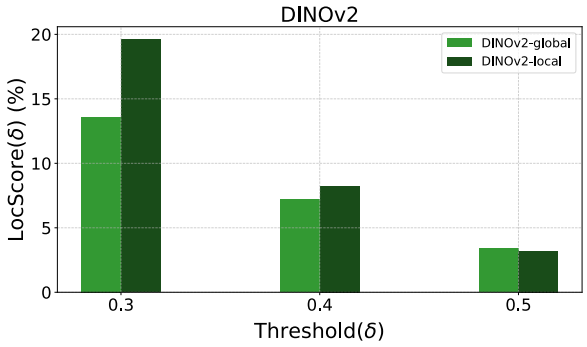
(d) ConvNext-IN1k



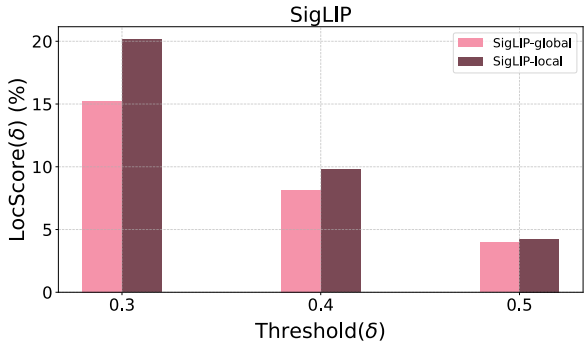
(e) ConvNext-LAION2b



(f) CLIP



(g) DINOv2



(h) SigLIP

Figure S9. 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 8. 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 9. Quantitative results with mAP and LocScore under various backbones.

S10. Qualitative Results

Figure S10. Qualitative results of Global and Patchify on ILIAS

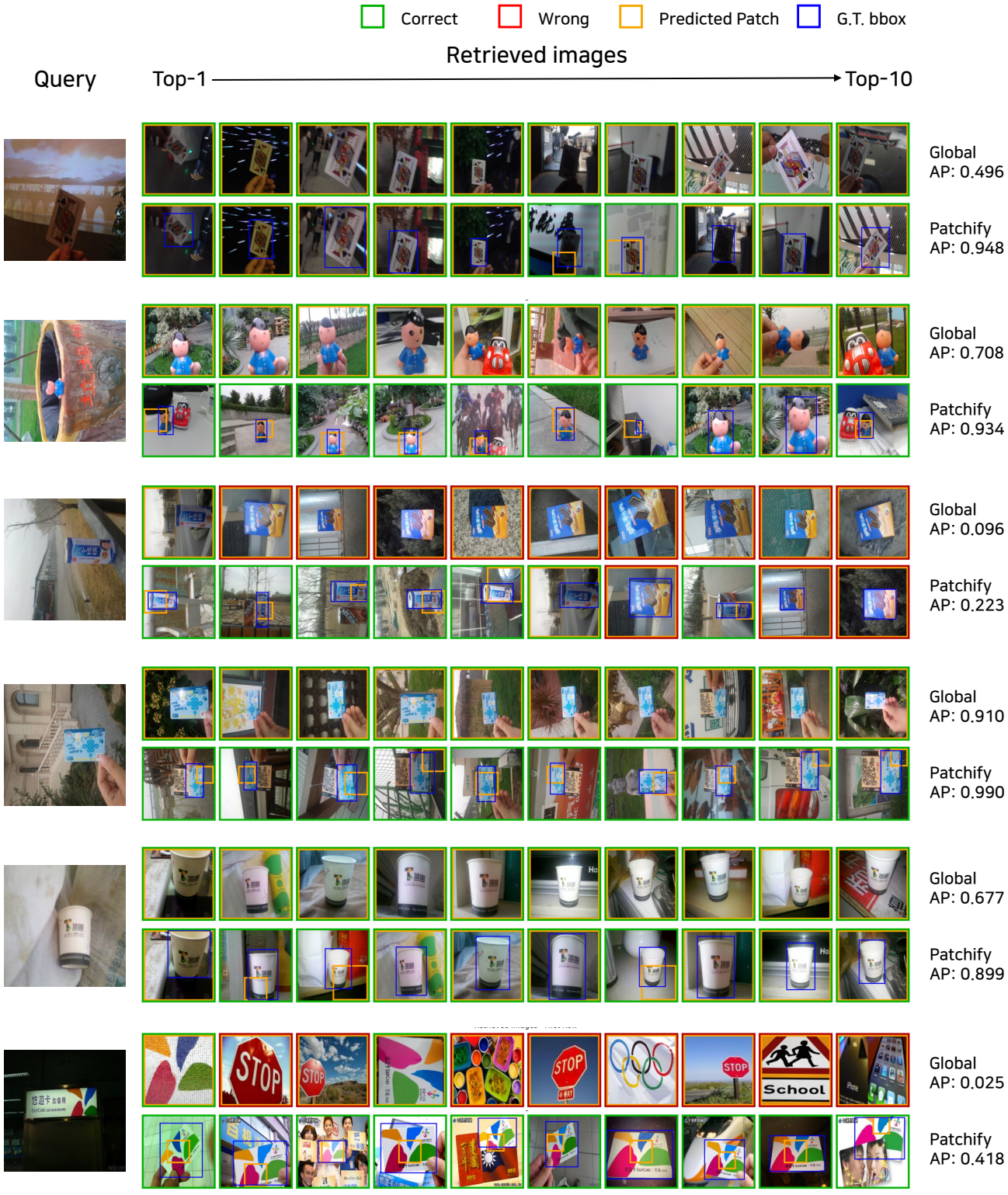


Figure S11. Qualitative results of Global and Patchify on INSTRE