Tracing Copied Pixels and Regularizing Patch Affinity in Copy Detection

Supplementary Material

A.1. More about DISC21 Data

The DISC21 competition [11] comprises four datasets: `training set`, `reference set`, `dev set` (which is evenly split into two parts: `dev set part I` and `dev set part II`) and `test set`. All data in `training set` is unlabeled. Both `dev set` and the `test set` function as queries, sharing `reference set`. According to the rules outlined on the DISC21 official website[1], the competition is divided into two phases. In the first phase, `dev set part I` and its labels are made public, serving as a validation set for the participants. Meanwhile, `dev set part II` is also made public, but its labels remain concealed for the purpose of ranking participants' performance in the first phase. Both data and labels in `test set` will not be disclosed during this phase. In the second phase, images in `test set` will be released for the final rankings. After the competition, labels for both `dev set part II` and `test set` will be accessible. For clarity, please refer to Tab. 6. It is noteworthy that the competition rules allow the fine-tuning on `dev set part I`. Since times of evaluating on `dev set part II` during the first phase are no limited, while times of evaluation on `test set` is restricted, there are more reported results on `dev set part II` from public papers than those on `test set`. Therefore, Tab 1 of this paper utilizes results on `dev set part II`, while the simulated results on `test set` are presented in Tab. 4.

A.2. More about Evaluation Protocols

The formulas for calculating mAP (mean Average Precision) and μAP (unified Average Precision) are as follows:

$$\text{mAP} = \frac{1}{N}\sum_{i=1}^{N} \text{AP}(\{s_{ij}|j = 1, 2, \dots\}), \tag{A.1}$$

$$\mu\text{AP} = \text{AP}(\{s_{ij}\}), \tag{A.2}$$

where s_{ij} denotes similarity score between query i and reference j. mAP averages Average Precision (AP) of each sample, while μAP concatenates similarity scores of all image pairs to compute AP. Consequently, under mAP formulation, the distribution of similarity scores for each query may vary significantly. While in practical scenarios, a fixed threshold is typically used to filter edited copy pairs. In cases where the distribution of similarity scores varies greatly, mAP may not accurately reflect the performance of model, as the case in Fig. 5. In contrast, μAP computes AP based on all aggregated scores, which better simulates real-world scenario.

[1] https://www.drivendata.org/competitions/group/image-similarity-challenge

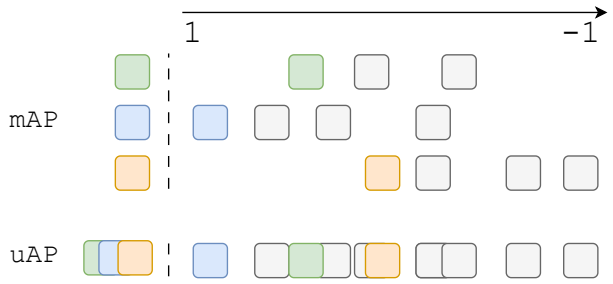

Figure 5. **mAP *v.s.* μAP.** In this case, mAP is 100%, because all positive samples are ranked at the first place in the view of each query. While no threshold could be set to achieve that perfect performance. In μAP, results are concatenated together to calculate precision of each threshold. Thus, μAP could better reflect the real performance of model. Mathematically, mAP also serves as an upper bound of μAP.

A.3. More about Post-Processing

A.3.1. Score Normalization

Score normalization (SN) is a widely used technique in data mining. SN maps data from different distributions to a common distribution. If the distributions of similarity score vary among queries, a global threshold is difficult to be set. Then SN could effectively address this problem and improve μAP significantly. We refer to SN method in SSCD [35], which is formulated as follows:

$$s_{ij} = \cos(z_i, z_j) - \alpha \frac{1}{k_{\text{end}} - k_{\text{start}} + 1} \sum_{k=k_{\text{start}}}^{k_{\text{end}}} \cos(z_i, z_{k\text{-NN}}), \tag{A.3}$$

where s_{ij} is normalized score of query x_i and reference x_j, z_i and z_j are their features extracted by descriptor. k-NN is the top-k nearest neighbor of query x_i in auxiliary data. Here we simply employ `training set` as auxiliary data and set $k_{\text{start}} = 0$, $k_{\text{end}} = 9$, $\alpha = 1$, which means that takes the closest neighbor in `training set` as bias.

A.3.2. Feature Stretching

Feature stretching a technology to stabilize scores across different queries. It is introduced by Wang *et al.* in BoT [49]. In short, query feature z_q is stretched by formula

		Training Set	Reference Set	Dev Set Part I		Dev Set Part II		Test Set	
				Data	Labels	Data	Labels	Data	Labels
Scale		1,000,000	1,000,000	25,000		25,000		50,000	
Accessible	Phase 1	✓	✓	✓	✓	✓	✗	✗	✗
	Phase 2	✓	✓	✓	✓	✓	✗	✓	✗
	After	✓	✓	✓	✓	✓	✓	✓	✓

Table 6. **Details of DISC21 datasets.** In "Accessible" part, ✓ denotes that corresponding data is available during that period of time, and ✗ denotes unavailability.

Method	μAP	R@1	Method	μAP	R@1
SSCD [35]	14.22	20.24	**ViT-S**	**27.07**	**34.68**
S-square [33]	14.51	21.05	**ViT-B**	**31.66**	**37.78**
Lyakaap [59]	13.80	18.02	ViT-S†	25.38	31.57
AnyPat. Base. [51]	16.18	20.54	ViT-B†	28.05	34.36

Table 7. **Results on AnyPattern.** All methods are evaluated with "SmallPattern" protocol . "AnyPat. Base." denotes Baseline in AnyPattern. CopyNCE results are marked in blue and † means results achieved with augmentations that aligned with Lyakaap.

	SSCD SN [35]	**ViT-S SN**	**ViT-B SN**
Descriptor μAP	64.99	70.59	**71.57**
Matching μAP	46.92	**51.32**	50.05

Table 8. **Results on VSC2022.** Results are produced by official baseline implementation of VSC2022 on its training set.

below:

$$\bar{z_q} = \beta \frac{\sum_{i=1}^{k} s_i}{k} z_q, \tag{A.4}$$

where β is stretching coefficient, $\{s_1, s_2, \ldots, s_k\}$ denotes the similarity scores (inner product) of k-NN images of query in the auxiliary dataset. After all query features are stretched, euclidean distance is utilized to ranking similarity between query and reference. In this paper, we follow the settings in BoT [49] that $\beta = 2.5$, $k = 5$ and let training set as the auxiliary set.

A.4. Experiments on more datasets

AnyPattern [51]. To evaluate the generalization of CopyNCE to unseen copy edits, we conducted experiments on AnyPattern [51] test set. Note that models are only trained on `training set` of DISC21, without any finetuning on `dev set part I`. AnyPattern encompasses rare and aggressive copy edits that are completely disjoint from the augmentation pipeline employed during CopyNCE training. As reported in Tab. 7, under the "SmallPattern" setting, CopyNCE with ViT-S achieves 27.07% μAP and 34.68% R@1, whereas the ViT-B variant attains 31.66% μAP and 37.78% R@1. Although our original augmentation is more aggressive than competing approaches, we further aligned it with the augmentation settings of Lyakaap [59] and retrained CopyNCE from scratch. Even under this reduced augmentation, CopyNCE still achieves 25.38% μAP / 31.57% R@1 (ViT-S) and 28.05% μAP / 34.36% R@1 (ViT-B). Despite the expected performance drop due to weaker augmentation, these results exceed the best competing method by at least 9.20%+ μAP and 10.52%+ R@1. This empirical evidence corroborates that CopyNCE possesses superior generalization to previously unseen infringement patterns.

Test Set	Model	Resolution	Metrics		
			mAP	μAP	RP90
`Dev II`	ViT-S	224×224	90.6	83.5	75.4
`Dev II`	ViT-B		90.5	83.5	76.7
`Dev II`	ViT-S	336×336	91.3	85.8	79.9
`Dev II`	ViT-B		91.3	85.7	80.1

Table 9. **Scaling of matcher.** All results are achieved without finetuning on `dev set part I`.

Test Set	Model	Resolution	Metrics		
			mAP	μAP	RP90
`Dev II`	ViT-S	224×224	76.5	70.5	63.6
`Dev II`	ViT-B		77.9	72.3	65.2
`Dev II`	ViT-S	336×336	75.0	69.8	63.9
`Dev II`	ViT-B		76.2	71.3	66.0

Table 10. **Scaling of descriptor.** All results are achieved without finetuning on `dev set part I`.

VSC2022 [36]. To further validate CopyNCE in the context of video copy detection, we evaluate the model trained on DISC21 by following the official baseline[1] of VSC2022 [36] and testing it on the VSC2022 training set. Under score normalization, CopyNCE surpasses SSCD by 5.6%+ Descriptor μAP and 3.1%+ Matching μAP. These results indicate that CopyNCE, when employed as a feature extractor for video copy detection, still delivers impressive performance.

A.5. Scaling Performance

Scaling performance matters in cases of subtle and complex edits. In this section, we explored how resolution and model size affect performance. For matcher, increasing resolution leads to improvements of 2.2%+ μAP / 3.4%+ RP90. However, unlike results of "Separate" shown in Tab. 1, enlarging model size for CopyNCE fails to boost performance. We hypothesize that matcher allows interaction between query and reference tokens, while CopyNCE provides direct supervision for interaction. Such guidance is sufficient enough for either ViT-S or larger ViT-B to tell whether query edits upon reference. And this will lead to performance saturation in terms of model size. This could be another evidence of effectiveness of CopyNCE.

For descriptor, scaling model size brings lifts of 1.5%+ μAP / 1.6%+ RP90 in Tab. 10, which is consistent with expectations. However, increasing resolution has negative effects. We believe the primary reason is that training descriptor requires a larger batch size. However, increasing the resolution significantly raises the memory consumption for ViT. To accommodate training at a higher resolution, it is necessary to reduce the batch size, which leads to the decline in descriptor performance. Another potential reason will be discussed in Sec. A.12.

A.6. More about Implementation Details

All our training and finetune implementation details are listed in Tab. 11 and Tab. 12. Note that due to numerical stability, we re-implemented the average and one-hot mode of CopyNCE in order to conduct experiments of $\gamma = 0$ and $\gamma = +\infty$. And in finetuning, no pixel mappings are available due to human-made edits. Thus, only baseline loss is utilized for descriptor and matcher.

A.7. More about Model Arch

The architectures of both matcher and descriptor are illustrated in Fig. 6. Matcher comprises an encoder and a fusion module, both constructed from attention blocks of same architecture. Matcher takes query and reference images as input, encoding each through encoder to obtain patch tokens. Subsequently, a learnable `[CLS]` token is concatenated with the tokens from both query and reference. And then they are passed to the fusion module for information interaction. Finally, CopyNCE supervises fused tokens, while binary cross-entropy (BCE) loss optimizes `[CLS]` token through a fully connected head. To enhance efficiency and reduce computational load, matcher is constructed based on a default ViT-S, which has 12 layers of attention blocks. The first eight layers form the encoder, while the last four layers constitute the fusion.

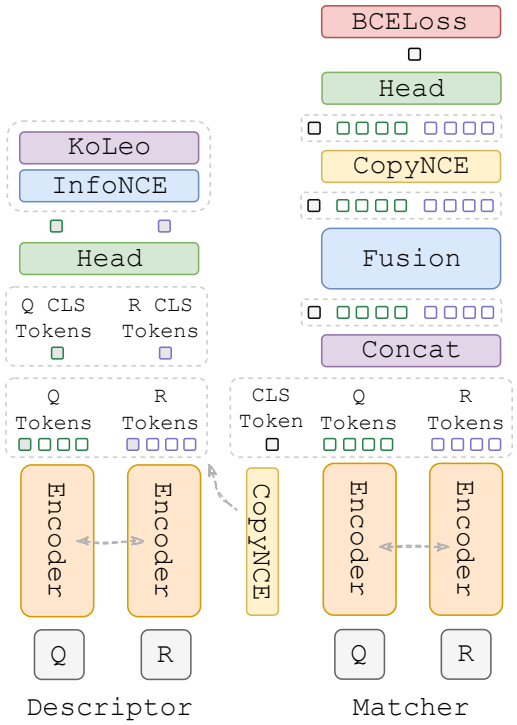

Figure 6. **Detailed architecture of matcher and descriptor.**

In contrast, descriptor is solely based on ViT. Upon receiving query and reference, descriptor extracts their `[CLS]` tokens and patch tokens. In baseline scenario, `[CLS]` tokens of both images are trained with InfoNCE [45] and KoLeo [39] loss. Within our framework, CopyNCE regularizes patch tokens in the last layer. Similarly, descriptor defaults to using ViT-S for simplification.

Additionally, the property of both matcher and descriptor are listed in Tab. 13.

A.8. More about Tricks

A.8.1. Global Hard Negative Mining

During training of descriptor, although hard negative mining (HNM) is performed in both KoLeo and InfoNCE, the mining process is constrained within batch, which is limited by batch size. Thus, for a training set consisting of millions of images, the probability of identifying hard samples in k-NNs is minimal. To address this issue, we utilize global hard negative mining (GHNM) as Algo. A.8.1. This method ensures that every sample in the mini-batch

[1] https://github.com/facebookresearch/vsc2022/blob/main/docs/baseline.md
[2] https://github.com/facebookresearch/dino

Settings		Parameters	
		Matching	Descriptor
Input	Resolution	224×224 / 336×336	
	Augmentation	color jitter, random grayscale, random blur, overlay text, overlay emoji, random flipping, affine, perspective, random resized-crop, overlay image, random erasing, etc.	
	Positive Rate	0.3	-
	Hard Negative Mining	p=0.5, k-NN (k=128)	p=1.0, k-NN (k=8)
Model	Arch	ViT-S/16 ViT-B/16 [9] 8 Encoder Layers + 4 Fusion Layers	ViT-S/16 ViT-B/16 [9]
	Linear Head	-	512 dims
Optim	Pretraining	DINO [3] ViT-S/16 ViT-B/16[2]	DINO [3] ViT-S/16 ViT-B/16[2]
	Batch Size	8 GPUs $\times$ 32	8 GPUs $\times$ 96
	Epoch	30 (base lr $= 1e-3$) + 30 (base lr $= 2e-4$)	30 (base lr $= 6e-4$)
	Optimizer	AdamW (β=[0.9, 0.999])	
	Weight Decay	0.04	
	Learning Rate	$0.001 \times \sqrt{\text{bs}/1024}$	$0.0006 \times \sqrt{\text{bs}/1024}$
	Scheduler	Cosine Scheduler	
	Min lr	$2.0e-06$	
	Warmup Epoch	1	
	Clip Grad	3.0	
Loss	Baseline Loss	BCELoss $\times$ 1	InfoNCE $\times$ 1 + KeLeoLoss $\times$ 5
	CopyNCE	Refer to Sec. 4	

Table 11. **All detailed settings in our training pipeline.** For the aspects of data augmentation that have been omitted, please refer to our code.

Settings		Parameters	
		Matching	Descriptor
Input	Resolution	224×224 / 336×336	
	Augmentation	resize	
	Positive Rate	0.3	-
	Hard Negative Mining	Off	Off
Model	Linear Head	-	256 dims
Optim	Pretraining	CopyNCE	CopyNCE
	Batch Size	8 GPU $\times$ 32	1 GPU $\times$ 96
	Epoch	20 (base lr $= 2 \times 10^{-4}$)	30 (base lr $= 1 \times 10^{-4}$)
	Learning Rate	$2 \times 10^{-4} \times \sqrt{\text{bs}/1024}$	$1 \times 10^{-4} \times \sqrt{\text{bs}/1024}$
Loss	Baseline Loss	BCELoss $\times$ 1	InfoNCE $\times$ 1 + KeLeoLoss $\times$ 5
	CopyNCE	Off	

Table 12. **All detailed settings in our finetune pipeline.** If certain settings are not listed in this table, they are set to be the same settings used in training by default.

can at least find a k-NN-level negative sample, significantly improving the lower bound for hard negative mining. We conduct ablation studies on GHNM and the results are reported in Tab. 14. Compared to baseline w/o GHNM, which employs standard hard negative mining (HNM) within the batch, GHNM leads to substantial 7.1% μAP / 30.6% RP90 enhancement for descriptor, clearly demonstrating the necessity of GHNM.

	Encoder	Fusion	Extensive	Task
Descriptor	Att. Block	Cosine	1 *v.s.* N	Coarse Retrieval
Matcher	Att. Block	Att. Block	1 *v.s.* 1	Fine Matching

Table 13. **Property of descriptor and matcher.** "Att. Block" denotes Attention block for short. Descriptor extends well because it retrieves images through vector similarity and can yield N results with a single matrix multiplication. While matcher has to perform classification pair by pair.

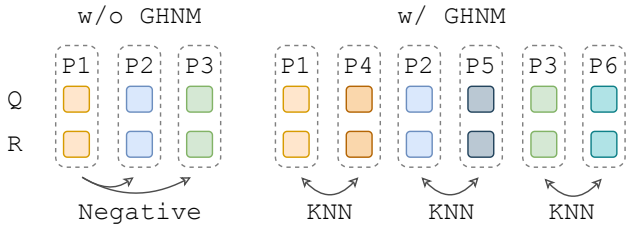

Figure 7. **Demonstration of GHNM.**

Method	Parameter	Metrics		
		mAP	μAP	RP90
Baseline		76.4	68.9	60.6
Baseline	w/o GHNM	77.1	57.7	30.0

Table 14. **Ablation studies of GHNM** on **descriptor**.

Algorithm 1 Global Hard Negative Mining (GHNM)

Input: Batch Size N.
Input: KNN constructed by DINO [3] features.
Output: Mini-batch $\{[x, x']_i | i = 1, \ldots, N\}$.

Init mini-batch $B = \emptyset$.
for $i = 1, \ldots, \frac{N}{2}$ **do**
  Sample random image x.
  Generate positive pair $[x, x']$ of x.
  Randomly select global hard negative x_{hn} of x from KNN.
  Generate positive pair $[x_{\text{hn}}, x'_{\text{hn}}]$ of x_{hn}.
  Update mini-batch $B = B \cup \{[x, x'], [x_{\text{hn}}, x'_{\text{hn}}]\}$
end for

A.8.2. Local Crops Ensembling

In copy detection, there are numerous edited copy instances of small regions, which present significant challenges to algorithms. In response to this issue, a straightforward approach is local crops ensembling, abbreviated as LCE. The primary concept of LCE is to crop query q and reference r according to a fixed rule, followed by pairwise comparisons between the cropped queries and references. Ultimately, LCE takes the maximum score as the copy score of query q and reference r. For matcher, LCE takes the highest copy probability, while for descriptor, it takes the maximum cosine similarity.

As posted in Tab. 1, LCE significantly enhances matcher performance (2.9%+ μAP / 4.0%+ RP90). However, the downside of LCE is also evident: it requires substantial computational resources. In our implementation, we cropped 26 regions (25 local + 1 global) from query and 10 regions (9 local + 1 global) from reference, leading to a corresponding computation cost of 260x. These regions include rotation and different ratios of crops. Please refer to our code for more details of LCE. It is important to emphasize that we design this complexity mainly because some approaches in Tab. 1 adopt sophisticated rules for the best performance and we follow them for fair comparison.

A.9. More about Candidates List for Matcher

As illustrated in Tab. 13, matcher can only perform classification pair by pair. According to Tab. 6, `reference set` has 1M images and `dev set` or `test set` has over 25k images. If we force matcher to classify all possible query and reference pairs, it will incur exceptionally high computational cost. To solve this problem, descriptor is first utilized to recall as many edited copy cases as possible to generate a candidate list. Next, we apply matcher to classify all pairs within this candidate list. When recalling with descriptor, LCE mentioned in Sec. A.8.2 is employed.

Specifically, for each crop of query, we identify k-NN in reference set and each query recalls 390 candidates. We ensemble the candidate lists from "Baseline" (ViT-B, 224×224) and "CopyNCE" (ViT-B, 224×224), recalling a total of 780 candidates. And then duplicated candidates for each query will be removed and the top 400 candidates form a multi-model fused candidate list. Finally, we use matcher "CopyNCE" (ViT-S, 224×224) in conjunction with LCE to select the top 10 highest-scoring candidates to generate the final candidate list. All matcher will subsequently utilize this candidate list for inference. Recall of different steps is listed in Tab. 15. Note that recall could be viewed as the upper bound of mAP and μAP, *i.e.,* our reported μAP of matcher in Tab. 1 can no longer be greater than 93.4%. More details could be found in our code.

A.10. Solutions of DISC21 Phase 2

DISC21 Phase 2 used `test set` to measure performance, which is more challenging compared to `dev set part II`. To achieve better results on `test set`, we follow the official rules and finetune both descriptor and matcher on `dev set part I`, with the tuning parameters detailed in Tab. 12. It is important to note that, according to DISC21 rules, "Descriptor Track" only allows features with 256 dim. Therefore, we reduced the default linear head dimensions from 512 to 256.

For "Descriptor Track", we finetune "CopyNCE" de-

Step	Model				Tricks	# Candidates	Recall
	Type	Method	Backbone	Resolution			
0	-				-	1,000,000	100.0%
1	Descriptor	Baseline CopyNCE	ViT-B	224 × 224	LCE	400	94.3%
2	Matcher	CopyNCE	ViT-S	224 × 224	LCE	10	93.4%

Table 15. **Recall of dev set part II in different candidates retrieval steps.**

Step	Model					Tricks	# Candidates	Recall
	Type	Method	Pre-train	Backbone	Resolution			
0	-					-	1,000,000	100.0%
1	Descriptor	Finetune	Baseline CopyNCE	ViT-B	224 × 224	LCE	400	91.3%
2	Matcher	Finetune	CopyNCE	ViT-S	224 × 224	LCE	10	90.1%

Table 16. **Recall of test set in different candidates retrieval steps.**

scriptor (ViT-B, 336 × 336) to obtain the final descriptor and perform inference at 336 × 336. The result presented in Tab. 4 is achieved with feature stretching (as described in Sec. A.3.2). To obtain the candidate list for matcher, we repeat the process outlined in Sec. A.9 with the distinction that descriptors used for retrieval are finetuned on `dev set part II`. The recall after different steps are listed in Tab. 16. Finally, we utilized finetuned matcher (ViT-S, 336 × 336) to achieve the result shown in Tab. 4.

A.11. More about Reverse Operation of Table

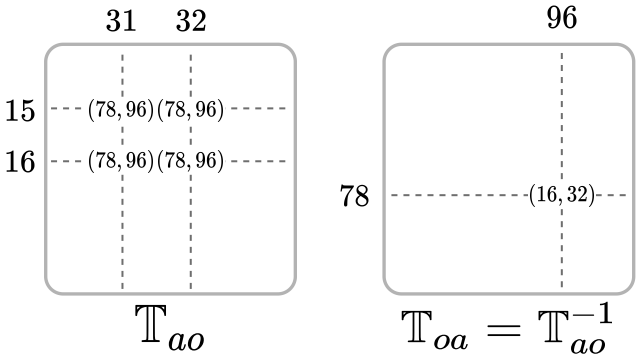

Figure 8. **Reverse issue of table $\mathbb{T}$.**

When performing the reverse operation on the coordinate table $\mathbb{T}$, the keys and values are often not uniquely corresponding, as multiple keys may correspond to the same value. If table $\mathbb{T}$ is reversed, a single key may correspond to multiple values. In such cases, we apply a row-by-row reversal rule, where the subsequently reversed value overrides the previous one. For instance, in the case shown in the diagram, if image I_o is magnified to twice its size and overlaid on image I_a, the coordinates (15, 31), (15, 32), (16, 31), and (16, 32) in image I_a are tracked to the pixel at coordinate (78, 96) in image I_o. Therefore, if table $\mathbb{T}_{ao}$ is reversed, key coordinate (78, 96) will correspond to multiple value coordinates. Following the row-by-row reversal process, the later value will override the earlier one, making (78, 96) correspond to (16, 32).

A.12. More about ResNet experiments

For a 224×224 input image, ResNet-50 yields $7 \times 7 = 49$ regional features, whereas ViT with the default patch size of 16×16 produces $14 \times 14 = 196$ patch tokens. To align the spatial granularity of ResNet-50 with that of ViT, we upsample the input to 448×448 in our ResNet-50 experiments. Without this alignment, CopyNCE fails to confer performance gain. We attribute this phenomenon to the fact that, for descriptor, the fraction of image area occupied by each regional feature critically modulates the efficacy of CopyNCE, and 14×14 appears to constitute an optimal trade-off. Tab. 10 also corroborates this hypothesis that increasing the input resolution from 224×224 to 336×336 does not improve performance.

A.13. More Visualization

To better visualize the effectiveness of CopyNCE series, we provide additional cases that include both success (Fig. 9 for matcher and Fig. 11 for descriptor) and failure (Fig. 10 for matcher and Fig. 12 for descriptor) examples of matcher

and descriptor. In success cases, the copied images are correctly identified with the highest scores. Some of these cases even applied a large number of image transformations. This indicates that CopyNCE series is capable of handling most common edited copy cases. However, it remains evident that when copy area is relatively small, the descriptor is less effective compared to the matcher, as scores of descriptor decline significantly when the proportion of copy regions decreases. In failure cases, edited images generally employed highly exaggerated transformations that fused the model, which still remained a challenge to our models.

A.14. Potential Limitations

Unlike image-level supervision, CopyNCE requires a coordinate table as their supervisory signal. For an edited copy pair under the default settings, a table containing 224×224 key-value pairs needs to be generated. Therefore, training process may encounter CPU bottlenecks, resulting in longer training time for CopyNCE compared to that for image-level supervised pipelines.

Besides, to ensure a fair comparison with other SOTA methods, CopyNCE utilizes the LCE trick when obtaining matcher results, which significantly enhances performance. However, in practical applications, the high computational cost of LCE makes it nearly impossible to be leveraged in real-life applications. Regarding this limitation, we argue that even without LCE, CopyNCE still is still capable to yield competitive results, as shown in some cases in Tab. 1.

Finally, since copy detection emphasizes features that contain rich texture information, such features often lack semantic understanding compared to models of SSL or other image retrieval tasks. Consequently, they tend to focus on detailed texture information between two images while overlooking semantic foregrounds.

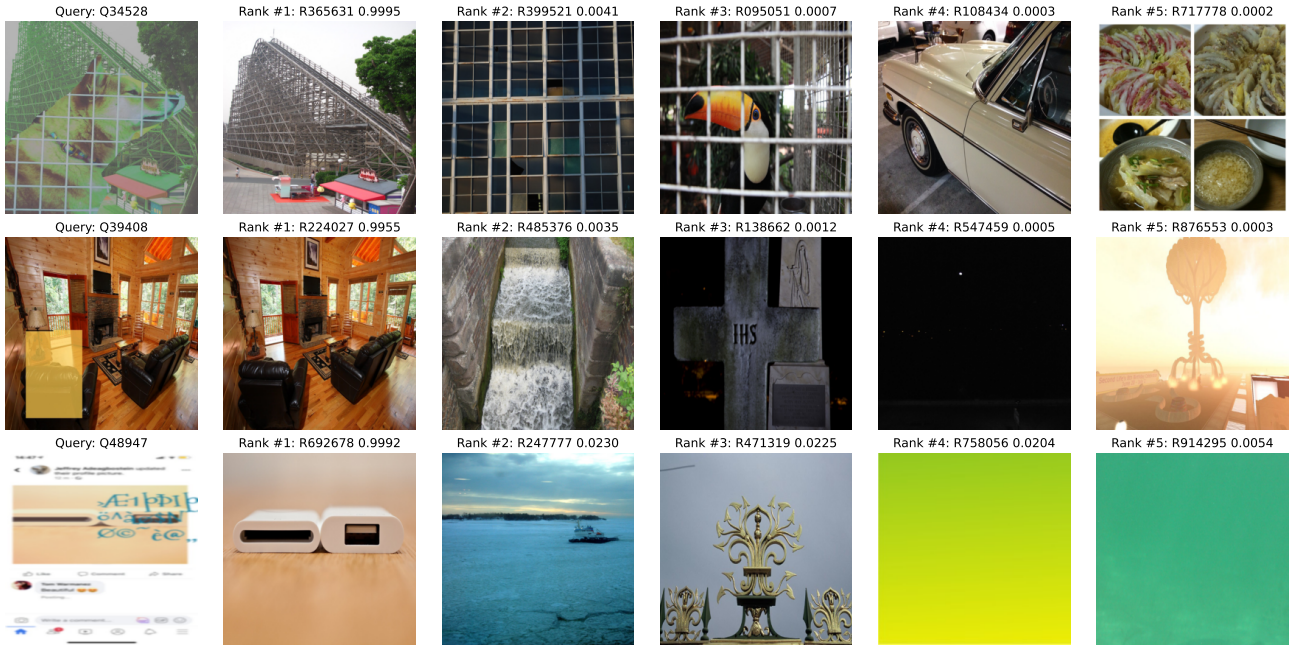

Figure 9. **Success cases of matcher.** In each row, the query image is placed in the first column, followed by five recalled reference images sorted by score in descending order. Each image is titled with its rank, ID and copy score. Notably, the ground truth is always ranked first.

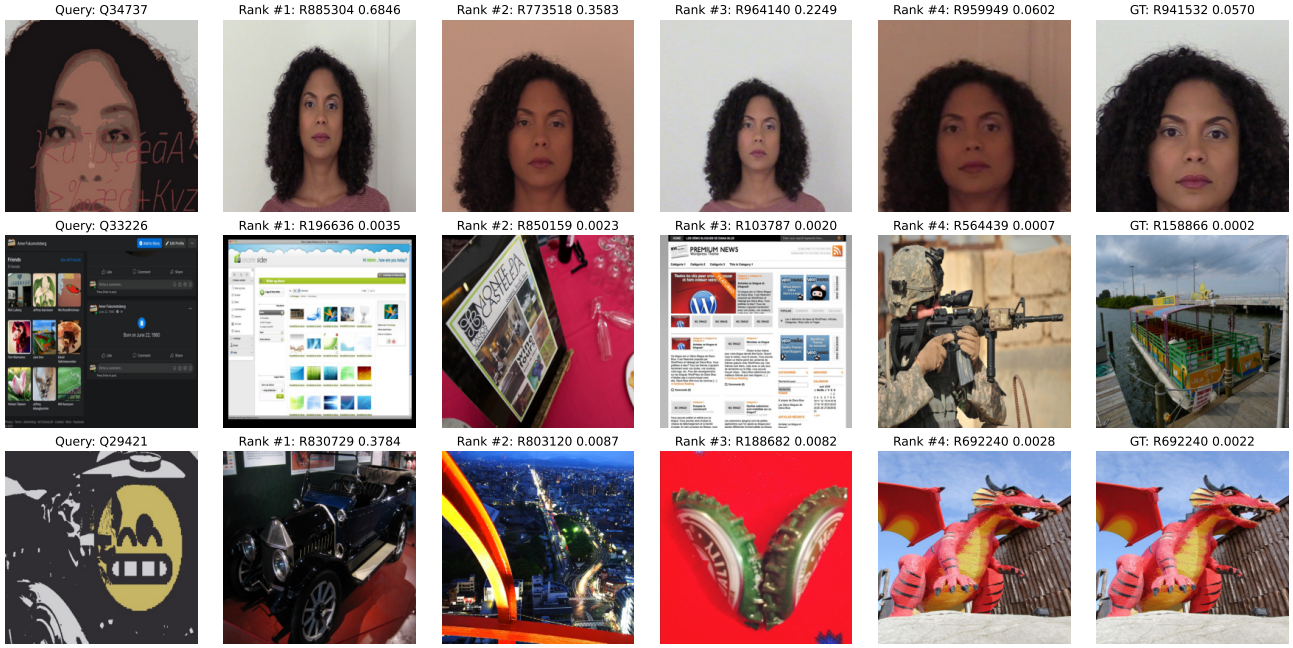

Figure 10. **Failure cases of matcher.** In each row, the query image is placed in the first column, followed by four recalled reference images sorted by score in descending order. Each image is labeled with its rank, ID and copy score. In these cases, the ground truth copied image is placed last.

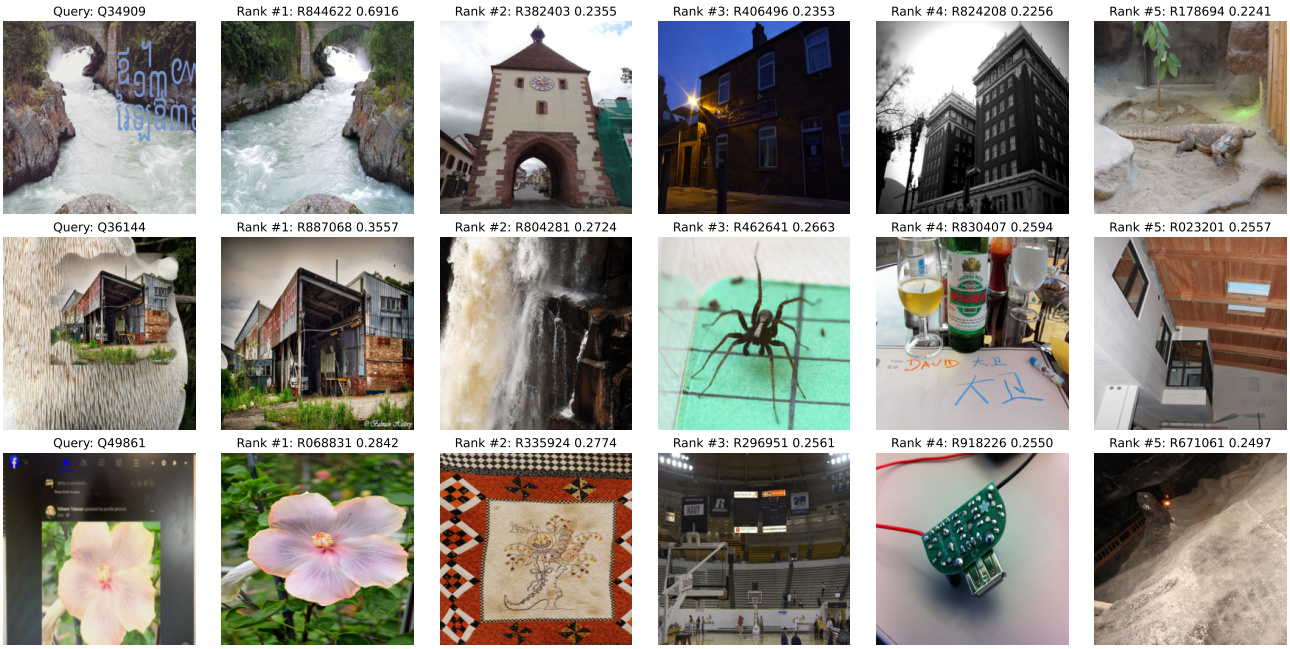

Figure 11. **Success cases of descriptor.** In each row, the query image is placed in the first column, followed by five recalled reference images sorted by score in descending order. Each image is titled with its rank, ID and copy score. Notably, the ground truth is always ranked first.

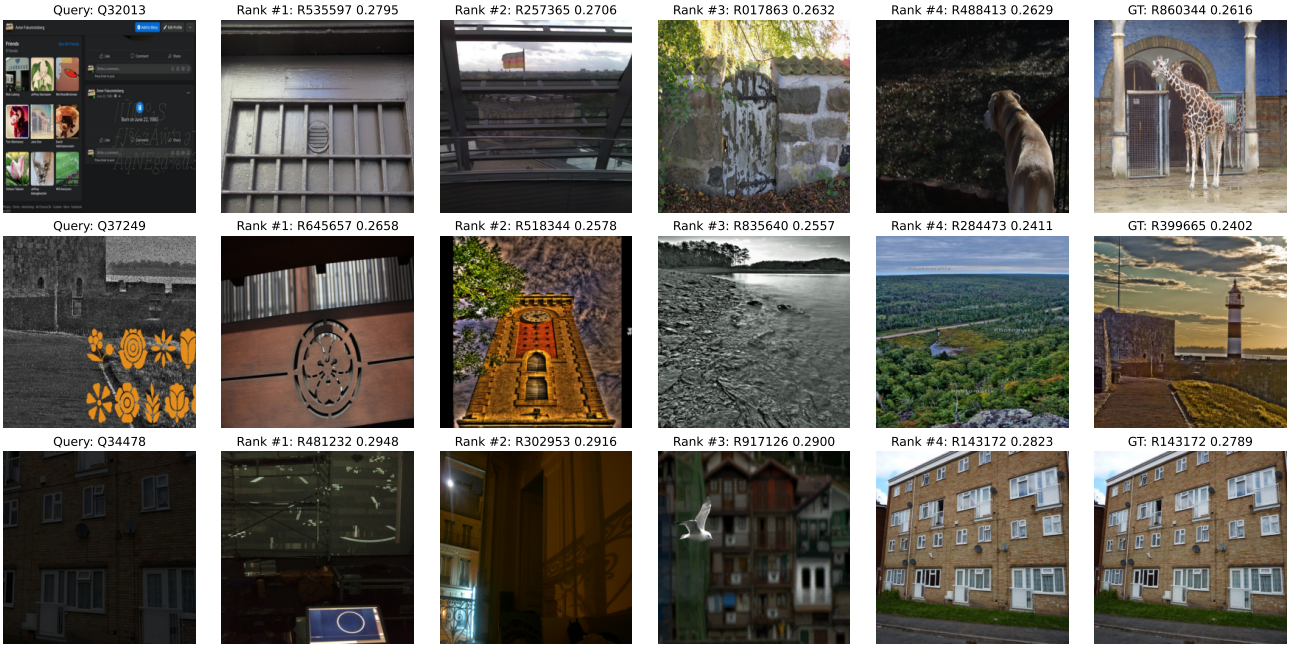

Figure 12. **Failure cases of descriptor.** In each row, the query image is placed in the first column, followed by four recalled reference images sorted by score in descending order. Each image is labeled with its rank, ID and copy score. In these cases, the ground truth copied image is placed last.