ILIAS: Instance-Level Image retrieval At Scale –supplementary material–

Giorgos Kordopatis-Zilos Vladan Stojnić Anna Manko Pavel Šuma
Nikolaos-Antonios Ypsilantis Nikos Efthymiadis Zakaria Laskar
Jiří Matas Ondřej Chum Giorgos Tolias

A. Implementation details

Collection process. Queries and positives are created/collected by a group of 16 collectors who are well-informed about the task objectives. Most of the images consist of photographs taken by the collectors for the purpose of this work, while a smaller part is downloaded from online repositories with a permissive license. All collected images are manually filtered and curated by the authors. Regarding the selection of the objects, the collectors are advised to opt for objects with distinct, uncommon features–such as unique shapes, colors, or textures–that set them apart within their category, *i.e.* prioritize items with rare modifications. As mentioned in the main paper, objects that are created or share parts with other objects created before 2014 do not qualify as query objects. Fig. A illustrates some of the objects rejected during the selection process. Fig. Aa is the Kuggen building, whose construction finished in 2011. Fig. Ab is a newly bought coaster that displays a well-known van Gogh painting. Fig. Ac is a newly bought cutlery holder whose design is rather generic with no distinctive detail; hence, very similar (close to identical) objects may exist in YFCC100M. Furthermore, the collectors are provided with older camera models used in YFCC100M. This simulates similar camera distribution for the query and positive images with the distractors. Tab. A shows the distribution of the most used cameras. Older-generation cameras are used for the majority of the collected images. The collectors are instructed to avoid using the same camera for both the query and the positives of an object to avoid any possible shortcuts learned by pre-trained models.

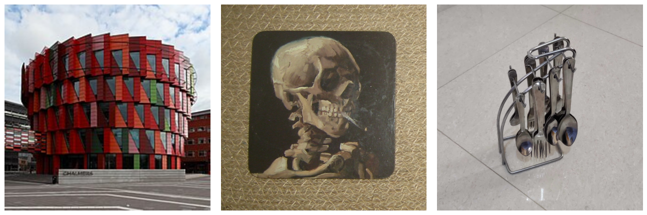

(a) (b) (c)

Figure A. **Rejected objects.** Example of objects that are disregarded during the selection process.

model	year	type	images
Canon EOS 450D	2008	DSLR	770
NIKON D3000	2009	DSLR	585
NIKON 3100	2010	DSLR	443
DiMAGE X1	2005	camera	286
Xiaomi Poco X5 Pro	2023	phone	275
iPhone 14	2022	phone	237
Xiaomi Redmi Note 11 Pro	2022	phone	210
Canon EOS 6D Mark II	2017	DSLR	208
iPhone SE (3rd generation)	2022	phone	195
NIKON 5300	2013	DSLR	144
Canon EOS 50D	2008	DSLR	141
iPhone 14 Pro	2022	phone	122
Canon PowerShot S5 IS	2007	DSLR	118
ONEPLUS A6003	2018	phone	110
Canon EOS REBEL T2i	2010	DSLR	102

Table A. **Most frequently used camera models in ILIAS.** Cameras used for more than 100 images are displayed. Information about release date and type of camera is provided.

Downloading and storing images. To acquire the YFCC100M [42], we download images based on the Flickr URLs provided by the original authors. Approximately ∼82M images are downloaded. The remaining images are downloaded from the AWS S3 data bucket provided by the authors. We opt for downloading the images from Flickr to ensure that identical preprocessing has been applied to the distractor dataset and the collected query and positive sets in ILIAS. The collected images in ILIAS are also uploaded to and downloaded from Flickr. We use the "medium" option to download all images, which resizes images to 500px based on their larger side. All images are stored with 90 JPEG compression quality with 4:4:4 chroma subsampling. Following [1, 12], white balancing is applied on all images. All personal details (*e.g.* human faces, license plates) that are displayed in the collected images of ILIAS are either blurred or cropped.

a close-up photo of the ⋆.
a good photo of the ⋆.
a photo of a cool ⋆.
a low resolution photo of the ⋆.
a bad photo of the ⋆.
a cropped photo of the ⋆.
a photo of a hard to see ⋆.
a bright photo of a ⋆.
a photo of a clean ⋆.
a photo of a dirty ⋆.
a dark photo of the ⋆.
a photo of my ⋆.
a photo of the cool ⋆.
a close-up photo of a ⋆.
a bright photo of the ⋆.

Table B. **Examples of templates** used for the text query generation for the creation of *mini*-ILIAS. The ⋆ symbol is replaced with a taxonomy term.

```
You are a system generating descriptions of
objects shown in an image.
Provided with an image and a category in which
the item shown in the image belongs to, you will
describe the main item that you see in the image,
giving enough details to unambiguously describe
the object.
You can describe unambiguously what the item is
and its material, color, and style if clearly
identifiable.
Please do not describe anything about the
background.
```

Figure B. **Prompt** used for the initial generation of text queries.

***mini*-ILIAS composition.** We consider the 88 text category labels from ILIAS taxonomy to generate text queries, manually expanded with 132 terms that are synonyms or fine-grained descriptions of the original labels. The collected labels are combined with 43 templates used in the original CLIP [32] to generate a list of 9,976 text queries. Examples of the templates used are in Tab. B. We do not consider domain-specific templates. We use the large model variants of SigLIP, OpenCLIP, and EVA-CLIP to compute the ensemble text-image similarities between the text queries and each image of YFCC100M.

Text query generation. We generate text queries using the GPT-4o [26]. The prompt displayed in Fig. B is first provided to the LLM. Then, a query image of one of the objects in ILIAS and its corresponding category is provided to the model to generate a textual description. For object category, we use mid level category from taxonomy. If it is not available, we use the coarser level category. The generated text queries are manually edited by the authors to fix errors, insufficient descriptions, or nuances of the model.

Global representations. For the implementation of global representation models, we rely on public resources available on PyTorch [28]. We use the timm[1] and torchvision[2] libraries that provide relevant code and weights for the majority of the models. For the models not included there, we use the relevant code from the official github repositories provided by the authors, *i.e.* [27][3], [5][4], [4][5], [15][6], [29][7], [19][8], [23][9], [35][10], [2][11], [34][12], [50][13], [51][14]. Model weights that are not publicly available are provided to us by the original authors. For t2i, we use the image encoders from timm and the text encoders from huggingface[15] and OpenCLIP[16]. We include only base and large model variants in our benchmark. Tab. G and H contain more information, including model checkpoints. Regarding image preprocessing, following instance-level retrieval literature [23, 31, 35], the images are resized based on their largest side respecting their aspect ratio, *i.e.* isotropic rescaling. Image resolution is dictated by each model's specifications together with the rule setting resolution one level higher than those used during training. This rule is empirically created based on experiments presented in Sec. B.1. We normalize the image tensors with the mean and standard deviation statistics according to model specifications. For all ViT-based models, bicubic interpolation of the position embeddings is performed. Unicom [2] requires fixed-size tensors in the backbone output, which goes through a projection head; hence, we use adaptive average pooling to fix the spatial dimensions of the output feature tensors. For UDON [51] and USCRR [50] models, we use the representation before projection due to the low dimensionality of the latter. For AlexNet [22] and VGG [36] models, we extract descriptors based on the feature maps of the last convolutional layer by applying GeM pooling [31]. For the rest of the models, the extraction process used in the original methods is employed. All global descriptors are ℓ_2 normalized.

[1] github.com/rwightman/pytorch-image-models
[2] github.com/pytorch/vision
[3] github.com/facebookresearch/dinov2
[4] github.com/facebookresearch/dino
[5] github.com/facebookresearch/swav
[6] github.com/facebookresearch/moco-v3
[7] github.com/yash0307/recallatk_surrogate
[8] github.com/tjddus9597/hier-cvpr23
[9] github.com/sungonce/cvnet
[10] github.com/shihaoshao-gh/superglobal
[11] github.com/deepglint/unicom
[12] github.com/naver/unic
[13] github.com/nikosips/universal-image-embeddings
[14] github.com/nikosips/udon
[15] huggingface.co
[16] github.com/mlfoundations/open_clip

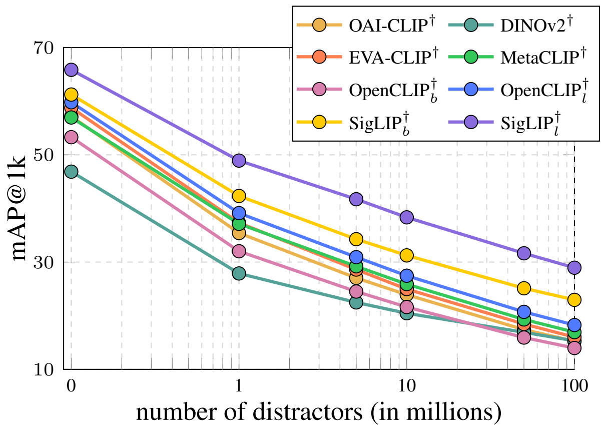


Figure C. **Impact of the number of distractors.** mAP@1k of five models for varying db size. † indicates results with the linear adaptation. b and l: base and large model variants.

model	train res	224	384	512	724
EVA-CLIP [10, 39]	224	5.0	7.7	5.8	3.1
MetaCLIP [48]	224	5.1	8.8	6.5	3.8
OpenCLIP [17, 24]	224	8.2	10.7	6.1	2.5
DINOv2 [27]	518	6.4	12.2	14.0	14.3
SigLIP [53]	224	9.1	14.1	10.2	6.1
SigLIP [53]	512	0.1	8.9	18.9	20.2
EVA-CLIP [10, 39]	336	4.7	13.1	13.4	9.5
MetaCLIP [48]	224	10.3	14.4	11.0	7.4
OpenCLIP [17, 24]	320	10.3	16.5	12.7	7.6
DINOv2 [27]	518	9.7	16.0	17.7	18.5
SigLIP [53]	256	8.3	18.8	15.4	10.8
SigLIP [53]	384	1.8	21.6	24.1	20.6

Table C. **Impact of resolution.** Performance (mAP@1k) by testing at different resolutions. The underline indicates the resolution selected for each model based on our rule. Linear adaptation is not used. Top: base models. Bottom: large models.

Linear adaptation. The single linear adaptation layer is trained on a 1M random subset of UnED [50]. The training follows the UJCDS [50] method that learns a linear classifier on all classes in the UnED subset (191,513 classes). The classifier gets the ℓ_2 normalized features output from the linear adaptation layer. During training, the Normalized Softmax loss [52] is minimized, and no balancing across UnED domains is performed. The linear layer and classifier are trained for 2 epochs with 128 batch size. We use Adam [20] optimizer with 10^{-3} learning rate and 10^{-6} weight decay. The scale of the Normalized Softmax loss is 16.

Local representations. Following AMES [38], local descriptors are extracted based on the base variant of DINOv2 with registers [8, 27]. Local descriptors are selected based on their weights estimated by a feature detector [3]. We use the pre-trained network trained on the corresponding descriptors. The local descriptor extraction, the pre-trained models, and inference configurations are publicly-available[17]. To ensure a fair comparison between re-ranking methods, we use the same local descriptors for other methods but with different binarization. AMES consists of a binarization layer initialized with ITQ [11, 21] and fine-tuned during model training. Hence, for Chamfer Similarity (CS) [33] and Spatial verification (SP) [30], the descriptors are binarized with the same ITQ weights. For SP, we follow the standard practice in retrieval with fast spatial matching [3] and use single correspondence hypotheses, which is translation in our case, and LO-RANSAC [7] for affine-transformation. Due to the single scale local descriptors, departing from the single correspondence hypothesis and sampling correspondence pairs or triplets can potentially provide better results despite being slower. Tentative inlier correspondences are extracted based on the nearest neighbor of each query local descriptor, using a threshold of 32 Hamming distance. Local similarity for re-ranking is estimated based on the number of inliers detected by RANSAC, with a minimum threshold of 5 inliers. Also, the final AMES similarity is an ensemble of local and global similarity. For a fair comparison, the same ensembling scheme is also used for CS and SP, following the same validation process. The ensemble hyper-parameters are tuned on the public split of the GLDv2 dataset. In the default settings, *i.e.* 100 binarized local descriptors for db images, the total memory requirements for storing local descriptors is ∼149GB, which is to be compared with ∼95GB needed for 512D global descriptors stored in half-precision. Note that we do not consider compression techniques for the global descriptors, which can decrease the memory footprint by an order of magnitude with an insignificant performance loss [13, 38].

[17] `github.com/pavelsuma/ames`

B. Additional experiments

Similar to the main paper, unless stated otherwise, we use the large ViT model variants with the largest resolution available, *e.g.* we use SigLIP ViT-L trained with 384 resolution. In the case of various architectures for the same method, we use the best-performing one, *e.g.* we use the large variant of ConvNext architecture for OpenCLIP.

B.1. Additional analysis

Impact of number of distractors. Fig. C presents the performance of five models under varying numbers of distractors. Performance declines as more distractors are added; however, significantly increasing the dataset's difficulty requires an exponential growth in the number of distractors. Notably, the ranking of models changes considerably when comparing performance with no distractors to that with 100M distractors. For example, DINOv2 demonstrates strong robustness to distractor increases, ranking last with no distractors but surpassing two models at 100M distractors and reaching others. Also, several crossings between models are observed. Therefore, evaluation at a large scale, provided by ILIAS, is important.

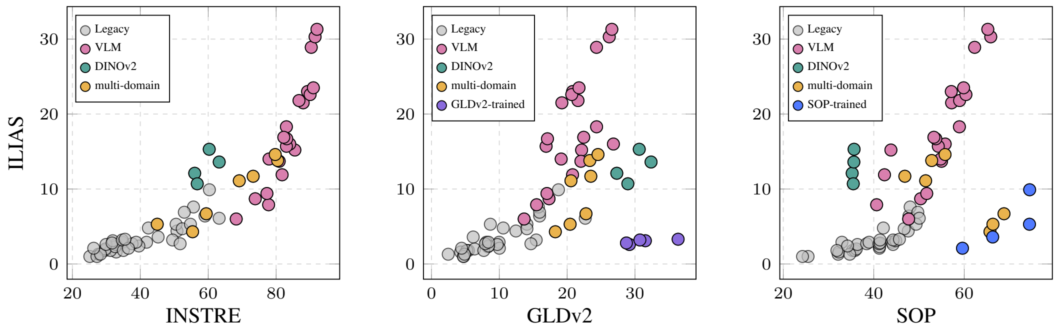


Figure D. **Comparison with other instance-level retrieval datasets** via reporting mAP@1k. Results with linear adaptation. INSTRE: 27.3K db size, multi-domain. GLDv2: 762K db size, single-domain. SOP: 60.5K db size, single-domain. Different network types are color-coded. For GLDv2 and SOP, models fine-tuned on these domains with the corresponding training sets are highlighted.

Impact of image resolution. In Tab. C, we investigate the impact of resolution and validate the rule of using as test resolution one up from the training one. Linear adaptation is not used in this experiment. It is clear that the vast majority of models achieve the best performance following the imposed rule; test at a resolution one level larger than the training resolution. Interestingly, SigLIP collapses when used with a resolution much lower than the training one.

Impact of background clutter. To quantify the impact of background clutter, we experiment with masking out areas outside object bounding boxes in the positives during descriptor extraction. This approach improves SigLIP† performance from 28.9 to 62.4. Fig. Ea also presents the impact of clutter, *i.e*. number of segments detected by SAM in a positive image outside of the object bounding box, on the ranking of this positive. This experiment provides insight about the type of positives, according to clutter, that populate the top and bottom ranks. Positives with less clutter, *i.e*. low number of segments, are the most common in the higher ranks; while, positives with more clutter, *i.e*. high number of segments, are the most common in the lower ranks.

Impact of object scale. Flowing the same strategy as above, but object bounding boxes are cropped and rescaled instead of masking, performance further improves to 69.4. However, although this does not reflect solely the impact of scale changes due to potential partial views and drastic viewpoint changes, it still gives a good insight into the limitations of the current models regarding scale changes. Fig. Eb presents the impact of relative scale, *i.e*. percentage of the bounding box area within the image area, on the ranking of positive. This experiment provides insight into the type of positives, according to relative image coverage, that populate the corresponding rank ranges. Positives where the object covers a large area are the most common in the higher ranks; while, positives with a small area coverage are the most common in lower.

Multi-scale and multi-rotation extraction. A common approach to address scale variation is multi-scale feature extraction, as widely adopted in the literature [31, 35]. Applying multi-scale extraction asymmetrically, *i.e*. only on the queries, yields an average 0.4 performance improvement across benchmarked models. SigLIP is marginally improved by 0.1. Multi-rotation is also tested in a similar manner, which, however, leads to an average drop of 0.3. Yet, SigLIP is marginally improved by 0.2.

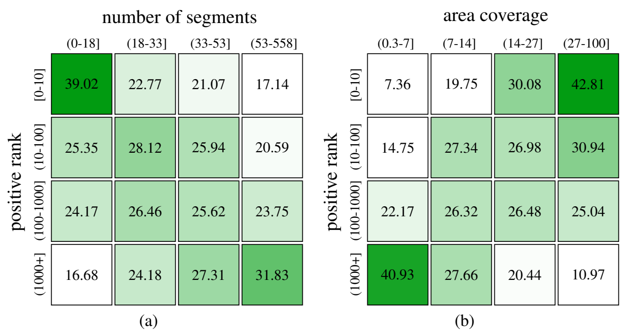


Figure E. **Impact of clutter and area coverage.** Percentage of images per ranking range based on SigLIP† and grouped based on (a) clutter, *i.e*. number of segments detected by SAM, (b) scale, *i.e*. area of object bounding box in images. Column bins contain the same number of positives. Normalization per row is applied.

Comparison with other datasets using linear adaptation. Fig. D presents the performance of global representation models with linear adaptation. Similar conclusions derive as in the case without adaptation. Only SigLIP achieves a competitive performance in SOP datasets out of the models not trained in-domain.

Performance per domain. Fig. F shows the average performance of objects grouped based on coarser taxonomy level.

Qualitative examples. Fig. L and M show examples of retrieved images based on i2i and t2i retrieval, respectively.

B.2. Linear adaptation

Comparison with other approaches. Tab. D compares the proposed linear adaptation with other linear projection methods trained on UnED for three models. All methods project the off-the-shelf descriptors to 512D ones. The unsupervised PCA whitening (PCA$_w$) [18] and the supervised learnable whitening (L$_w$) [31] approaches are evaluated. The proposed linear adaptation scheme achieves the best performance, typically with a large margin. It is the only one that does not drop off-the-shelf DINOv2 performance.

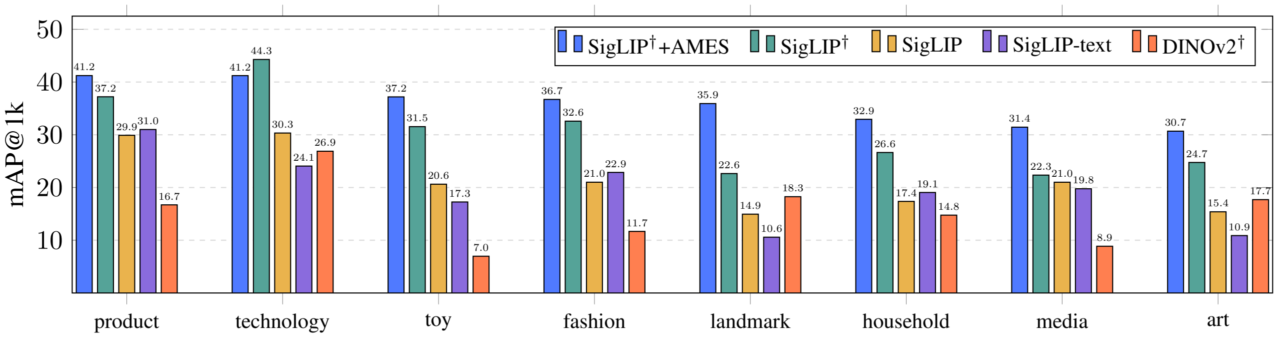


Figure F. **Performance comparison per primary category.** mAP@1k averaged over objects in the same primary-level category size, sorted by SigLIP†+AMES performance. Comparison between SigLIP with and without adaptation, SigLIP combined with AMES reranking, SigLIP t2i, and DINOv2. † indicates results with the linear adaptation.

model	labels	DINOv2	OpenCLIP	SigLIP
no adaptation	-	15.3	9.6	19.6
PCA$_w$ [18]	✗	14.8	12.5	22.2
L$_w$ [31]	✓	14.0	9.1	15.1
ours	✓	**15.3**	**18.3**	**28.9**

Table D. **Performance comparison for linear adaptation via mAP@1k.** Label requirement is indicated. Performance before adaptation is provided for reference.

dataset	domain	DINOv2	OpenCLIP	SigLIP
no adaptation	-	15.3	9.6	19.6
GLDv2 [47]	landmarks	14.6	14.2	25.6
Food2k [25]	food	12.6	13.6	22.6
Met [49]	artworks	14.7	5.1	7.6
iNaturalist [45]	natural world	14.2	16.3	26.4
UnED [50]	multi-domain	**15.3**	**18.3**	**28.9**

Table E. **Performance comparison of single- and multi-domain linear adaptation.** mAP@1k of models with linear adaptation trained on different dataset setups based on UnED. Performance before adaptation is provided for reference.

Impact of multi-domain linear adaptation. Tab. E illustrates the performance of several models with linear adaptation trained on the four largest single-domain datasets of UnED, as well as the entire UnED. Training on a single domain typically increases the performance of VLMs, except in the case of Met, where performance drops dramatically. DINOv2 performance decreases consistently with single-domain training. Nevertheless, the margin with multi-domain training is significant, indicating that multi-domain training on the whole UnED is best suited for ILIAS.

Impact of descriptor dimensionality. Fig. G illustrates the performance of five models linearly adapted on UnED with varying descriptor dimensionalities. For all models, performance saturates at a descriptor dimensionality of 256D, with only marginal improvements observed for most models beyond this point.

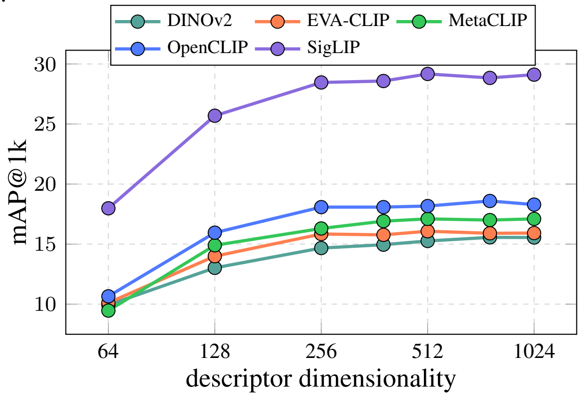


Figure G. **Impact of descriptor dimensionality.** mAP@1k of five models with the linear adaptation of various dimensionalities.

Robustness. We conduct three independent runs using different random seeds to evaluate the robustness of the linear adaptation. Across five global descriptors, the proposed scheme exhibits strong robustness, with a maximum standard deviation of 0.2 and a minimum of 0 across runs.

B.3. Re-ranking with local representations

Impact of top-M re-ranked images and number of local descriptors. Fig. H illustrates the performance of SigLIP with re-ranking when an increasing number of re-ranked images and local descriptors, translated to memory per image, are used. Performance increases as both variables increase. In the default scenario of top-1k and 100 descriptors, the performance is 35.6, which requires 0.6sec per query and approximately 150GB of memory. In an unconstrained scenario, the top performance is 38.8, requiring 20sec and almost 900GB.

Combination with various global representations. Tab. F presents the performance with and without re-ranking on ILIAS and *mini*-ILIAS using various models for global representation. mAP@1k is improved by more than 6 when re-ranking is applied for all models and datasets.

Qualitative examples. Fig. I presents some queries with the largest AP improvement from re-ranking with AMES. Several cases of severe clutter, scale changes, and partial views are successfully retrieved with re-ranking.

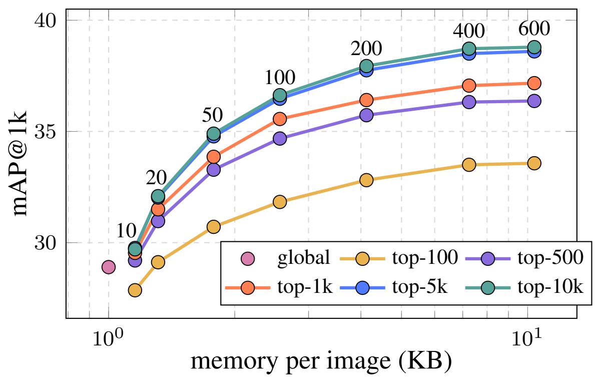


Figure H. **Impact of the re-ranking shortlist size and required memory for local descriptors.** Text above each point denotes the number of local descriptors per DB image. The shortlist size is indicated in the legend. Results are with the linear adaptation.

model	ILIAS		*mini*-ILIAS	
	mAP@1k	**oracle**	**mAP@1k**	**oracle**
DINOv2†	15.3	34.0	18.8	41.8
+ AMES	21.8	34.0	26.5	41.8
OpenCLIP†	18.3	48.0	22.9	56.3
+ AMES	27.1	48.0	32.9	56.3
SigLIP†	28.9	56.0	34.3	63.9
+ AMES	35.6	56.0	41.4	63.9

Table F. **Re-ranking on top of different global representations.** mAP@1k and oracle re-ranking on ILIAS and *mini*-ILIAS. + indicates re-ranking with AMES. † indicates results with the linear adaptation.

C. Dataset extras

Spatial location of objects in positives. Fig. J illustrates the spatial location of the object in the positives. Center bias in ILIAS is much less prominent in comparison with INSTRE [46] dataset.

Taxonomy. Fig. K illustrate the defined categories for the three taxonomy levels.

Query and positive examples. Fig. N provides visual examples of the collected queries and positives of several query objects.

Benchmarked models. Tab. G and H provide details and performance on ILIAS and *mini*-ILIAS of all models.

D. Dataset hosting, sharing and license

ILIAS is hosted in our servers in its entirety (*i.e.* collected images and the downloaded YFCC100M) to assert its long-term availability to the broader public. All collected images are shared under the permissive CC-BY 4.0 license. The downloaded images are distributed under their original license. All collectors have signed a consent form for the distribution of their images under this license.

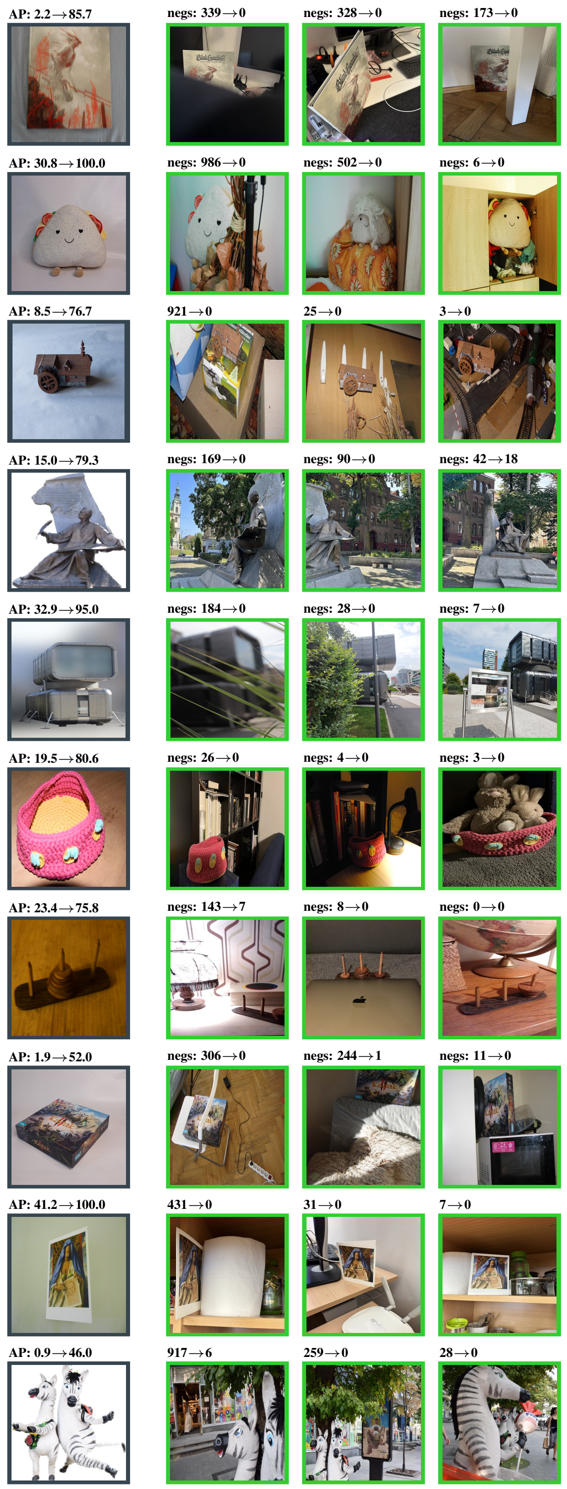


Figure I. **Re-ranking with AMES.** Queries with the most significant AP increase from re-ranking. The number of negatives ranked above positives is reported on top, as before → after re-ranking.

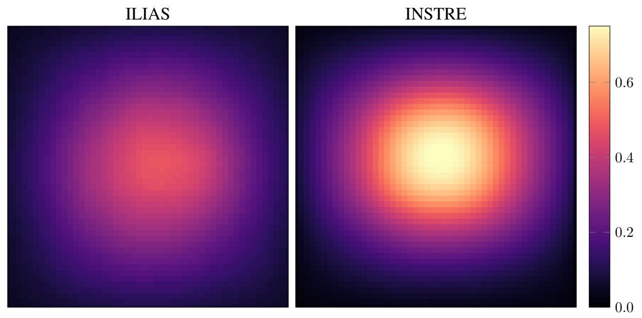


Figure J. **Distribution of object bounding boxes in positives.**

References

[1] Mahmoud Afifi and Michael S Brown. What else can fool deep learning? addressing color constancy errors on deep neural network performance. In *ICCV*, 2019. 1

[2] Xiang An, Jiankang Deng, Kaicheng Yang, Jaiwei Li, Ziyong Feng, Jia Guo, Jing Yang, and Tongliang Liu. Unicom: Universal and compact representation learning for image retrieval. In *ICLR*, 2023. 2, 9

[3] Bingyi Cao, André Araujo, and Jack Sim. Unifying deep local and global features for image search. In *ECCV*, 2020. 3

[4] Mathilde Caron, Ishan Misra, Julien Mairal, Priya Goyal, Piotr Bojanowski, and Armand Joulin. Unsupervised learning of visual features by contrasting cluster assignments. In *NeurIPS*, 2020. 2, 9

[5] Mathilde Caron, Hugo Touvron, Ishan Misra, Hervé Jégou, Julien Mairal, Piotr Bojanowski, and Armand Joulin. Emerging properties in self-supervised vision transformers. In *ICCV*, 2021. 2, 9

[6] Mehdi Cherti, Romain Beaumont, Ross Wightman, Mitchell Wortsman, Gabriel Ilharco, Cade Gordon, Christoph Schuhmann, Ludwig Schmidt, and Jenia Jitsev. Reproducible scaling laws for contrastive language-image learning. In *CVPR*, 2023. 9, 10

[7] Ondřej Chum, Jiří Matas, and Josef Kittler. Locally optimized ransac. In *GCPR*, 2003. 3

[8] Timothée Darcet, Maxime Oquab, Julien Mairal, and Piotr Bojanowski. Vision transformers need registers. In *ICLR*, 2024. 3, 9

[9] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby. An image is worth 16x16 words: Transformers for image recognition at scale. In *ICML*, 2021. 9

[10] Yuxin Fang, Wen Wang, Binhui Xie, Quan Sun, Ledell Wu, Xinggang Wang, Tiejun Huang, Xinlong Wang, and Yue Cao. EVA: Exploring the limits of masked visual representation learning at scale. In *CVPR*, 2023. 3, 9, 10

[11] Yunchao Gong, Svetlana Lazebnik, Albert Gordo, and Florent Perronnin. Iterative quantization: A procrustean approach to learning binary codes for large-scale image retrieval. *PAMI*, 2012. 3

[12] Rafael C Gonzalez. *Digital image processing*. 2009. 1

[13] Albert Gordo, Jon Almazan, Jerome Revaud, and Diane Larlus. End-to-end learning of deep visual representations for image retrieval. *IJCV*, 2017. 3

[14] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *CVPR*, 2016. 9

[15] Kaiming He, Haoqi Fan, Yuxin Wu, Saining Xie, and Ross Girshick. Momentum contrast for unsupervised visual representation learning. In *CVPR*, 2020. 2, 9

[16] Gao Huang, Zhuang Liu, Laurens Van Der Maaten, and Kilian Q Weinberger. Densely connected convolutional networks. In *CVPR*, 2017. 9

[17] Gabriel Ilharco, Mitchell Wortsman, Ross Wightman, Cade Gordon, Nicholas Carlini, Rohan Taori, Achal Dave, Vaishaal Shankar, Hongseok Namkoong, John Miller, Hannaneh Hajishirzi, Ali Farhadi, and Ludwig Schmidt. Openclip, 2021. 3, 9, 10

[18] Hervé Jégou and Ondřej Chum. Negative evidences and cooccurences in image retrieval: The benefit of pca and whitening. In *ECCV*, 2012. 4, 5

[19] Sungyeon Kim, Boseung Jeong, and Suha Kwak. HIER: Metric learning beyond class labels via hierarchical regularization. In *CVPR*, 2023. 2, 9

[20] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *ICLR*, 2015. 3

[21] Giorgos Kordopatis-Zilos, Christos Tzelepis, Symeon Papadopoulos, Ioannis Kompatsiaris, and Ioannis Patras. DnS: Distill-and-select for efficient and accurate video indexing and retrieval. *IJCV*, 2022. 3

[22] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *NeurIPS*, 2012. 2, 9

[23] Seongwon Lee, Hongje Seong, Suhyeon Lee, and Euntai Kim. Correlation verification for image retrieval. In *CVPR*, 2022. 2, 9

[24] Zhuang Liu, Hanzi Mao, Chao-Yuan Wu, Christoph Feichtenhofer, Trevor Darrell, and Saining Xie. A convnet for the 2020s. *CVPR*, 2022. 3, 9, 10

[25] Weiqing Min, Zhiling Wang, Yuxin Liu, Mengjiang Luo, Liping Kang, Xiaoming Wei, Xiaolin Wei, and Shuqiang Jiang. Large scale visual food recognition. *PAMI*, 2023. 5

[26] OpenAI. Gpt-4o system card. In *arXiv*, 2024. 2

[27] Maxime Oquab, Timothée Darcet, Théo Moutakanni, Huy Vo, Marc Szafraniec, Vasil Khalidov, Pierre Fernandez, Daniel Haziza, Francisco Massa, Alaaeldin El-Nouby, Mahmoud Assran, Nicolas Ballas, Wojciech Galuba, Russell Howes, Po-Yao Huang, Shang-Wen Li, Ishan Misra, Michael Rabbat, Vasu Sharma, Gabriel Synnaeve, Hu Xu, Hervé Jegou, Julien Mairal, Patrick Labatut, Armand Joulin, and Piotr Bojanowski. DINOv2: Learning robust visual features without supervision. *TMLR*, 2024. 2, 3, 9

[28] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. Pytorch: An imperative style, high-performance deep learning library. In *NeurIPS*, 2019. 2

[29] Yash Patel, Giorgos Tolias, and Jiří Matas. Recall@k surrogate loss with large batches and similarity mixup. In *CVPR*, 2022. 2, 9

[30] James Philbin, Ondřej Chum, Michael Isard, Josef Sivic, and Andrew Zisserman. Object retrieval with large vocabularies and fast spatial matching. In *CVPR*, 2007. 3

[31] Filip Radenović, Giorgos Tolias, and Ondřej Chum. Fine-tuning cnn image retrieval with no human annotation. *PAMI*, 2019. 2, 4, 5

[32] Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, Gretchen Krueger, and Ilya Sutskever. Learning transferable visual models from natural language supervision. In *ICML*, 2021. 2, 9, 10

[33] Ali S Razavian, Josephine Sullivan, Stefan Carlsson, and Atsuto Maki. Visual instance retrieval with deep convolutional networks. *ITE Trans. on Media Technology and Applications*, 2016. 3

[34] Mert Bulent Sariyildiz, Philippe Weinzaepfel, Thomas Lucas, Diane Larlus, and Yannis Kalantidis. UNIC: Universal classification models via multi-teacher distillation. In *ECCV*, 2024. 2, 9

[35] Shihao Shao, Kaifeng Chen, Arjun Karpur, Qinghua Cui, André Araujo, and Bingyi Cao. Global features are all you need for image retrieval and reranking. In *ICCV*, 2023. 2, 4, 9

[36] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. In *ICLR*, 2015. 2, 9

[37] Andreas Steiner, Alexander Kolesnikov, Xiaohua Zhai, Ross Wightman, Jakob Uszkoreit, and Lucas Beyer. How to train your vit? data, augmentation, and regularization in vision transformers. *TMLR*, 2021. 9

[38] Pavel Suma, Giorgos Kordopatis-Zilos, Ahmet Iscen, and Giorgos Tolias. AMES: Asymmetric and memory-efficient similarity estimation for instance-level retrieval. In *ECCV*, 2024. 3

[39] Quan Sun, Yuxin Fang, Ledell Wu, Xinlong Wang, and Yue Cao. EVA-CLIP: Improved training techniques for clip at scale. In *arXiv*, 2023. 3, 9, 10

[40] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going deeper with convolutions. In *CVPR*, 2015. 9

[41] Mingxing Tan and Quoc Le. Efficientnet: Rethinking model scaling for convolutional neural networks. In *ICML*, 2019. 9

[42] Bart Thomee, David A Shamma, Gerald Friedland, Benjamin Elizalde, Karl Ni, Douglas Poland, Damian Borth, and Li-Jia Li. YFCC100M: The new data in multimedia research. *Communications of the ACM*, 2016. 1

[43] Hugo Touvron, Matthieu Cord, Matthijs Douze, Francisco Massa, Alexandre Sablayrolles, and Hervé Jégou. Training data-efficient image transformers & distillation through attention. In *ICML*, 2021. 9

[44] Michael Tschannen, Alexey Gritsenko, Xiao Wang, Muhammad Ferjad Naeem, Ibrahim Alabdulmohsin, Nikhil Parthasarathy, Talfan Evans, Lucas Beyer, Ye Xia, Basil Mustafa, et al. SigLIP 2: Multilingual vision-language encoders with improved semantic understanding, localization, and dense features. In *arXiv*, 2025. 9, 10

[45] Grant Van Horn, Oisin Mac Aodha, Yang Song, Yin Cui, Chen Sun, Alex Shepard, Hartwig Adam, Pietro Perona, and Serge Belongie. The inaturalist species classification and detection dataset. In *CVPR*, 2018. 5

[46] Shuang Wang and Shuqiang Jiang. INSTRE: A new benchmark for instance-level object retrieval and recognition. *TOMM*, 2015. 6

[47] Tobias Weyand, André Araujo, Bingyi Cao, and Jack Sim. Google Landmarks Dataset v2 – A large-scale benchmark for instance-level recognition and retrieval. In *CVPR*, 2020. 5

[48] Hu Xu, Saining Xie, Xiaoqing Ellen Tan, Po-Yao Huang, Russell Howes, Vasu Sharma, Shang-Wen Li, Gargi Ghosh, Luke Zettlemoyer, and Christoph Feichtenhofer. Demystifying clip data. In *ICLR*, 2024. 3, 9, 10

[49] Nikolaos-Antonios Ypsilantis, Noa Garcia, Guangxing Han, Sarah Ibrahimi, Nanne Van Noord, and Giorgos Tolias. The MET dataset: Instance-level recognition for artworks. In *NeurIPS*, 2021. 5

[50] Nikolaos-Antonios Ypsilantis, Kaifeng Chen, Bingyi Cao, Mário Lipovský, Pelin Dogan-Schönberger, Grzegorz Makosa, Boris Bluntschli, Mojtaba Seyedhosseini, Ondřej Chum, and André Araujo. Towards universal image embeddings: A large-scale dataset and challenge for generic image representations. In *ICCV*, 2023. 2, 3, 5, 9

[51] Nikolaos-Antonios Ypsilantis, Kaifeng Chen, André Araujo, and Ondřej Chum. UDON: Universal dynamic online distillation for generic image representations. In *NeurIPS*, 2024. 2, 9

[52] Andrew Zhai and Hao-Yu Wu. Classification is a strong baseline for deep metric learning. In *BMVC*, 2018. 3

[53] Xiaohua Zhai, Basil Mustafa, Alexander Kolesnikov, and Lucas Beyer. Sigmoid loss for language image pre-training. In *ICCV*, 2023. 3, 9, 10

[54] Barret Zoph, Vijay Vasudevan, Jonathon Shlens, and Quoc V Le. Learning transferable architectures for scalable image recognition. In *CVPR*, 2018. 9

checkpoint	year	cite	repo	arch	train	dims	dataset	data size	train res	test res	5M	100M	5M†	100M†
alexnet.tv_in1k	2012	[22]	torchvision	CNN	sup	256	`in1k`	1M	224	384	2.0	1.5	1.9	1.3
vgg16.tv_in1k	2014	[36]	torchvision	CNN	sup	512	`in1k`	1M	224	384	3.0	2.3	2.3	1.6
resnet50.tv_in1k	2015	[14]	torchvision	R50	sup	2048	`in1k`	1M	224	384	2.3	1.7	2.5	1.8
resnet101.tv_in1k	2015	[14]	torchvision	R101	sup	2048	`in1k`	1M	224	384	2.7	1.9	2.7	1.8
densenet169.tv_in1k	2016	[16]	torchvision	CNN	sup	2048	`in1k`	1M	224	384	3.2	2.4	2.9	2.0
inception_v4.tf_in1k	2017	[40]	torchvision	CNN	sup	1536	`in1k`	1M	299	512	1.7	1.1	1.5	1.0
nasnetalarge.tf_in1k	2018	[54]	torchvision	CNN	sup	4032	`in1k`	1M	331	512	1.7	1.0	1.6	1.0
tf_efficientnet_b4.ns_jft_in1k	2019	[41]	timm	CNN	sup+dist	1792	`in1k`	1M	380	512	3.8	2.6	4.3	2.9
vit_base_patch16_224.augreg_in1k	2020	[9, 37]	timm	ViT-B	sup	768	`in1k`	1M	224	384	1.4	1.0	1.9	1.3
vit_base_patch16_224.augreg_in21k	2020	[9, 37]	timm	ViT-B	sup	768	`in21k`	14M	224	384	4.2	3.0	6.2	4.4
vit_large_patch16_224.augreg_in21k	2020	[9, 37]	timm	ViT-L	sup	1024	`in21k`	14M	224	384	6.0	4.6	7.3	5.3
vit_large_patch16_224.augreg_in21k_ft_in1k	2020	[9, 37]	timm	ViT-L	sup	1024	`in1k`	1M	224	384	5.1	3.6	6.6	4.7
vit_large_patch16_384.augreg_in21k_ft_in1k	2020	[9, 37]	timm	ViT-L	sup	1024	`in1k`	1M	384	512	7.2	5.3	8.7	6.4
deit3_base_patch16_224.fb_in1k	2021	[43]	timm	ViT-B	sup+dist	768	`in1k`	1M	224	384	3.8	1.2	2.7	1.8
deit3_large_patch16_224.fb_in1k	2021	[43]	timm	ViT-L	sup+dist	1024	`in1k`	1M	224	384	5.0	1.5	3.3	2.4
RN50.openai	2021	[32]	github	R50	vla	1024	`opanai`	400M	224	384	2.2	3.2	8.5	6.0
vit_base_patch16_clip_224.openai	2021	[32]	timm	ViT-B	vla	512	`opanai`	400M	224	384	3.3	4.2	10.7	7.9
vit_large_patch14_clip_224.openai	2021	[32]	timm	ViT-L	vla	768	`opanai`	400M	224	384	2.5	7.0	15.8	11.9
vit_large_patch14_clip_336.openai	2021	[32]	timm	ViT-L	vla	768	`opanai`	400M	336	512	4.4	9.4	19.9	15.2
vit_large_patch14_clip_224.laion2b	2021	[6, 17]	timm	ViT-L	vla	768	`laion2b`	2B	224	384	5.9	9.4	17.5	13.7
swav_resnet50	2021	[4]	github	R50	ssl	2048	`in1k`	1M	224	384	9.0	1.7	2.9	2.1
dino_resnet50	2021	[5]	github	R50	ssl	2048	`in1k`	1M	224	384	12.1	2.9	4.1	2.9
dino_vitb16	2021	[5]	github	ViT-B	ssl	768	`in1k`	1M	224	384	11.8	3.7	6.6	4.8
moco_v3_resnet50	2021	[15]	github	R50	ssl	2048	`in1k`	1M	224	384	1.9	2.6	3.4	2.6
moco_v3_vitb	2021	[15]	github	ViT-B	ssl	768	`in1k`	1M	224	384	2.0	1.9	3.2	2.3
convnext_base.fb_in1k	2022	[24]	timm	CN-B	sup	1024	`in1k`	1M	288	384	2.8	2.0	3.9	2.7
convnext_base.fb_in22k	2022	[24]	timm	CN-B	sup	1536	`in22k`	14M	224	384	3.2	6.4	9.9	7.6
convnext_large.fb_in1k	2022	[24]	timm	CN-L	sup	1024	`in1k`	1M	288	384	8.9	2.2	4.2	2.9
convnext_large.fb_in22k	2022	[24]	timm	CN-L	sup	1536	`in22k`	14M	288	384	8.6	6.6	9.1	6.9
convnext_base.clip_laion2b_augreg	2022	[17, 24]	timm	CN-B	vla	640	`laion2b`	2B	256	384	10.7	7.9	18.1	14.0
convnext_large_mlp.clip_laion2b_ft_soup_320	2022	[17, 24]	timm	CN-L	vla	768	`laion2b`	2B	320	512	12.7	9.6	22.9	18.3
recall_512-resnet50	2022	[29]	github*	R50	sup	512	`sop`	60k	224	384	3.7	1.6	3.1	2.1
recall_512-vit_base_patch16_224_in21k	2022	[29]	github*	ViT-B	sup	512	`sop`	60k	224	384	3.9	5.0	7.3	5.3
cvnet_resnet50	2022	[23]	github	R50	sup	2048	`gldv2`	1M	512	724	2.3	2.9	3.5	2.6
cvnet_resnet101	2022	[23]	github	R101	sup	2048	`gldv2`	1M	512	724	6.8	3.0	4.2	3.1
superglobal_resnet50	2023	[35]	github	R50	sup	2048	`gldv2`	1M	512	724	3.1	3.4	3.8	2.8
superglobal_resnet101	2023	[35]	github	R101	sup	2048	`gldv2`	1M	512	724	2.5	3.4	4.5	3.2
hier_dino_vits16_sop	2023	[19]	github*	ViT-S	sup	384	`sop`	60k	224	384	6.7	3.3	5.1	3.6
eva02_base_patch14_224.mim_in22k	2023	[10]	timm	ViT-B	ssl	768	`in22k`	14M	224	384	7.8	2.1	4.7	3.2
eva02_large_patch14_224.mim_in22k	2023	[10]	timm	ViT-L	ssl	1024	`in22k`	14M	224	384	13.6	1.5	3.9	2.7
eva02_large_patch14_224.mim_m38m	2023	[10]	timm	ViT-L	ssl	1024	`merged38m`	38M	224	384	4.3	4.7	8.8	6.1
eva02_base_patch16_clip_224.merged2b	2023	[10, 39]	timm	ViT-B	vla	512	`merged2b`	2B	224	384	4.5	5.9	11.7	8.7
eva02_large_patch14_clip_336.merged2b	2023	[10, 39]	timm	ViT-L	vla	768	`merged2b`	2B	336	512	13.8	10.9	20.9	16.0
unicom_vit_base_patch16_224	2023	[2]	github	ViT-B	dist	768	`laion400m`	400M	224	384	18.0	11.0	13.8	11.1
unicom_vit_large_patch14_224	2023	[2]	github	ViT-L	dist	768	`laion400m`	400M	224	384	17.8	13.8	17.7	13.8
unicom_vit_large_patch14_336	2023	[2]	github	ViT-L	dist	768	`laion400m`	400M	336	512	12.2	13.9	18.6	14.6
unicom_vit_base_patch16_gldv2	2023	[2]	github*	ViT-B	sup	768	`gldv2`	400M	512	724	3.7	3.0	4.1	3.3
unicom_vit_base_patch16_sop	2023	[2]	github*	ViT-B	sup	768	`sop`	400M	224	384	14.3	9.1	12.8	9.9
uscrr_64-vit_base_patch16_clip_224.openai	2023	[50]	github	ViT-B	sup	768	`uned`	2.8M	224	384	18.5	3.8	6.4	4.3
dinov2_vitb14	2023	[27]	github	ViT-B	ssl	768	`lvd142m`	142M	518	724	4.6	11.5	15.0	12.1
dinov2_vitl14	2023	[27]	github	ViT-L	ssl	1024	`lvd142m`	142M	518	724	14.1	15.3	18.8	15.3
vit_base_patch16_siglip_224.webli	2023	[53]	timm	ViT-B	vla	768	`webli`	10B	224	384	14.6	11.2	19.4	15.7
vit_base_patch16_siglip_256.webli	2023	[53]	timm	ViT-B	vla	768	`webli`	10B	256	384	19.3	11.5	20.6	16.7
vit_base_patch16_siglip_384.webli	2023	[53]	timm	ViT-B	vla	768	`webli`	10B	384	512	20.1	15.6	26.2	21.5
vit_base_patch16_siglip_512.webli	2023	[53]	timm	ViT-B	vla	768	`webli`	10B	512	724	18.8	16.6	27.5	23.0
vit_large_patch16_siglip_256.webli	2023	[53]	timm	ViT-L	vla	1024	`webli`	10B	256	384	24.2	15.2	26.3	21.8
vit_large_patch16_siglip_384.webli	2023	[53]	timm	ViT-L	vla	1024	`webli`	10B	384	512	5.7	19.6	34.3	28.9
vit_base_patch16_clip_224.metaclip_2pt5b	2024	[48]	timm	ViT-B	vla	768	`2pt5b`	2.5B	224	384	8.8	6.6	12.7	9.4
vit_large_patch14_clip_224.metaclip_2pt5b	2024	[48]	timm	ViT-L	vla	1024	`2pt5b`	2.5B	224	384	14.4	11.7	21.7	16.9
dinov2_vitb14_reg	2024	[8, 27]	github	ViT-B	ssl	768	`lvd142m`	142M	518	724	11.8	9.4	13.5	10.7
dinov2_vitl14_reg	2024	[8, 27]	github	ViT-L	ssl	1024	`lvd142m`	142M	518	724	15.9	12.7	17.1	13.6
unic_l	2024	[34]	github	ViT-L	dist	1024	`in1k`	1M	518	512	11.4	8.9	15.3	11.7
udon_64-vitb_in21k_ft_in1k	2024	[51]	github*	ViT-B	sup	768	`uned`	2.8M	224	384	7.5	5.5	7.3	5.3
udon_64-vitb_clip_openai	2024	[51]	github*	ViT-B	sup	768	`uned`	2.8M	224	384	8.3	5.9	9.2	6.7
vit_base_patch16_siglip_384.v2_webli	2025	[44]	timm	ViT-B	vla	768	`webli`	10B	384	512	18.4	15.0	27.5	22.6
vit_base_patch16_siglip_512.v2_webli	2025	[44]	timm	ViT-B	vla	768	`webli`	10B	512	724	18.6	15.4	28.6	23.5
vit_large_patch16_siglip_384.v2_webli	2025	[44]	timm	ViT-L	vla	1024	`webli`	10B	384	512	24.6	19.9	36.3	30.3
vit_large_patch16_siglip_512.v2_webli	2025	[44]	timm	ViT-L	vla	1024	`webli`	10B	512	724	25.3	20.8	37.3	31.3

Table G. **Benchmarked model details and mAP@1k on ILIAS and *mini*-ILIAS for global representation models for i2i**. Model details include the year of publication, repository used, architecture (arch), model descriptor dimensions (dims), training scheme (train), training data, and train/test resolution. 5M and 100M correspond to the mini and full versions of the dataset, respectively. For fine-tuned models, only the fine-tuning dataset is considered. Repo indicates the framework used to acquire model weights, *i.e.* torchvision, timm, or official github. $*$ indicates non-publicly available models provided by the original author. † indicates results with the linear adaptation. sup, ssl, dist, vla: supervised learning, self-supervised learning, distillation, vision-language alignment. R50, R101, CN: ResNet50, ResNet101 and ConvNext.

checkpoint	year	cite	repo	arch	dims	dataset	data size	train res	test res	5M	100M
RN50.openai	2021	[32]	oc	R50	1024	`opanai`	400M	224	384	2.3	1.5
vit_base_patch16_clip_224.openai	2021	[32]	timm+oc	ViT-B	512	`opanai`	400M	224	384	2.7	1.6
vit_large_patch14_clip_224.openai	2021	[32]	timm+oc	ViT-L	768	`opanai`	400M	224	384	6.7	4.6
vit_large_patch14_clip_336.openai	2021	[32]	timm+oc	ViT-L	768	`opanai`	400M	336	512	8.4	5.8
vit_large_patch14_clip_224.laion2b	2021	[6, 17]	timm+oc	ViT-L	768	`laion2b`	2B	224	384	9.4	7.0
convnext_base.clip_laion2b_augreg	2022	[17, 24]	timm+oc	CN-B	640	`laion2b`	2B	256	384	7.0	4.6
convnext_large_mlp.clip_laion2b_ft_soup_320	2022	[17, 24]	timm+oc	CN-L	768	`laion2b`	2B	320	512	11.5	8.1
eva02_base_patch16_clip_224.merged2b	2023	[10, 39]	timm+oc	ViT-B	512	`merged2b`	2B	224	384	4.4	2.5
eva02_large_patch14_clip_336.merged2b	2023	[10, 39]	timm+oc	ViT-L	768	`merged2b`	2B	336	512	10.6	7.2
vit_base_patch16_siglip_224.webli	2023	[53]	timm+hf	ViT-B	768	`webli`	10B	224	384	10.1	7.1
vit_base_patch16_siglip_256.webli	2023	[53]	timm+hf	ViT-B	768	`webli`	10B	224	384	10.3	7.5
vit_base_patch16_siglip_384.webli	2023	[53]	timm+hf	ViT-B	768	`webli`	10B	384	512	14.4	11.0
vit_base_patch16_siglip_512.webli	2023	[53]	timm+hf	ViT-B	768	`webli`	10B	512	724	14.6	11.1
vit_large_patch16_siglip_256.webli	2023	[53]	timm+hf	ViT-L	1024	`webli`	10B	256	384	16.4	12.8
vit_large_patch16_siglip_384.webli	2023	[53]	timm+hf	ViT-L	1024	`webli`	10B	384	512	22.2	18.1
vit_base_patch16_clip_224.metaclip_2pt5b	2024	[48]	timm+oc	ViT-B	768	`2pt5b`	2.5B	224	384	7.6	4.9
vit_large_patch14_clip_224.metaclip_2pt5b	2024	[48]	timm+oc	ViT-L	1024	`2pt5b`	2.5B	224	384	13.1	9.2
vit_base_patch16_siglip_384.v2_webli	2025	[44]	timm+hf	ViT-B	768	`webli`	10B	384	512	15.1	11.1
vit_base_patch16_siglip_512.v2_webli	2025	[44]	timm+hf	ViT-B	768	`webli`	10B	512	724	14.6	10.4
vit_large_patch16_siglip_384.v2_webli	2025	[44]	timm+hf	ViT-L	1024	`webli`	10B	384	512	23.7	18.6
vit_large_patch16_siglip_512.v2_webli	2025	[44]	timm+hf	ViT-L	1024	`webli`	10B	512	724	24.7	19.8

Table H. **Benchmarked model details and mAP@1k on ILIAS and *mini*-ILIAS for global representation models for t2i**. Model details include the year of publication, repository used, architecture (arch), model descriptor dimensions (dims), training data, and train/test resolution. 5M and 100M correspond to the mini and full versions of the dataset, respectively. Repo indicates the framework used to acquire model weights, *i.e*. timm for the image encoders and huggingface (hf) or OpenCLIP (oc) for the text encoders. R50, CN: ResNet50 and ConvNext.

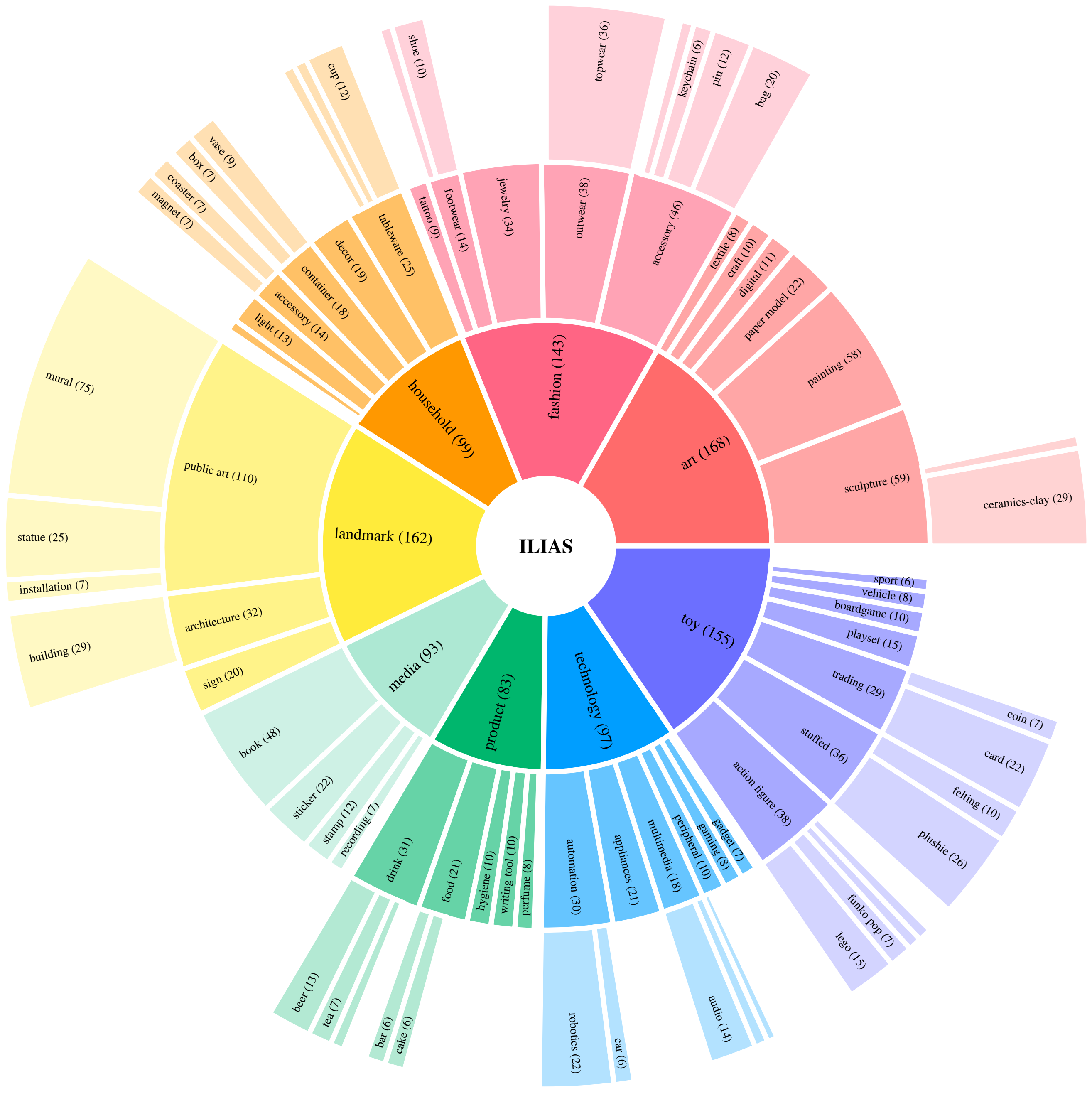


Figure K. **The ILIAS taxonomy** with a 3 level hierarchy. The number of objects is displayed for categories with more than 5 objects. The taxonomy is used to summarize the objects' diversity and distribution and to report performance per category without affecting the ground truth, which is defined at the instance level.

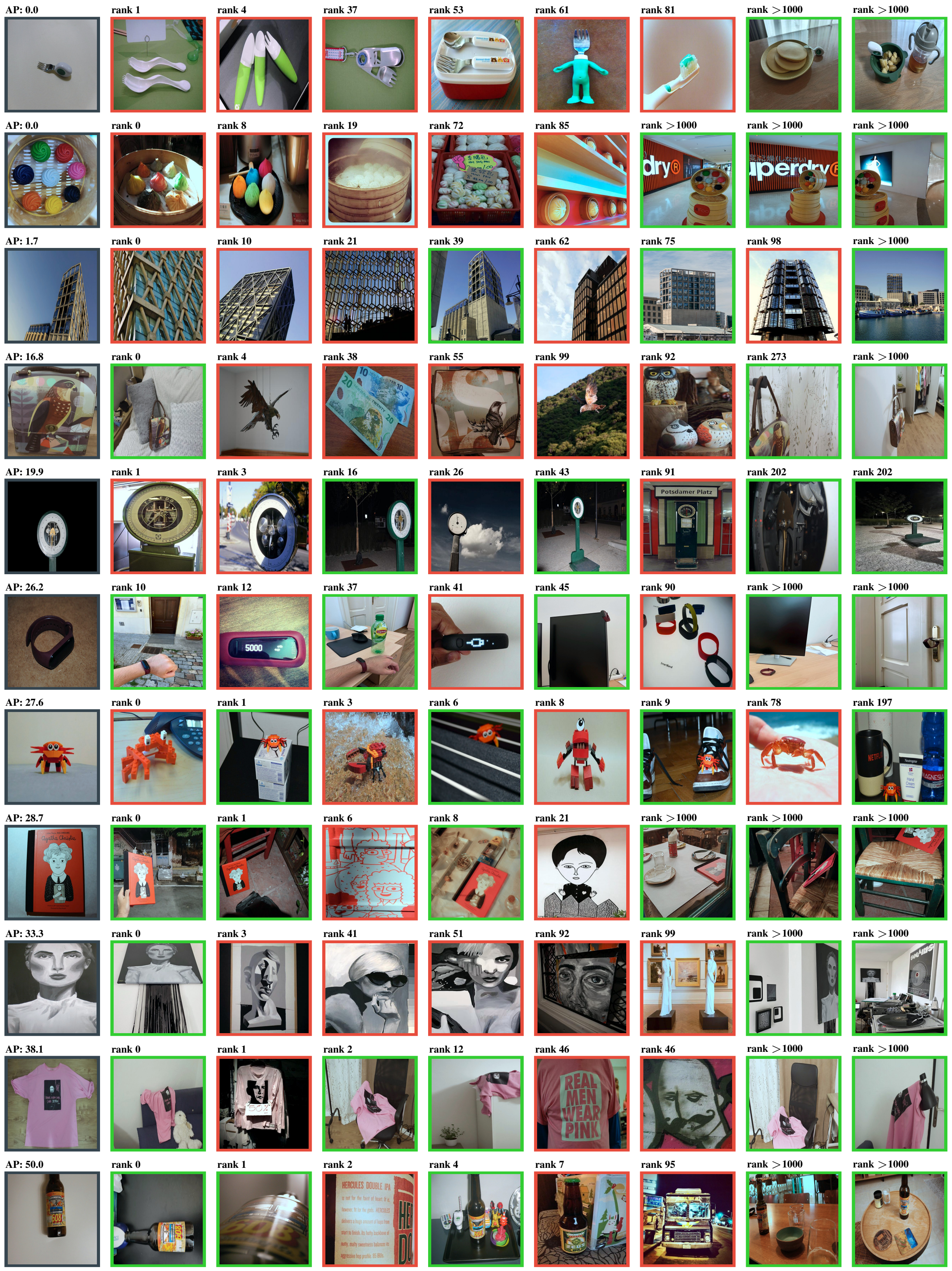
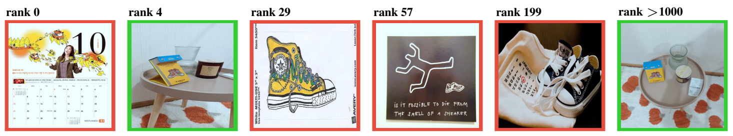

AP: 0.0 | rank 1 | rank 4 | rank 37 | rank 53 | rank 61 | rank 81 | rank >1000 | rank >1000
AP: 0.0 | rank 0 | rank 8 | rank 19 | rank 72 | rank 85 | rank >1000 | rank >1000 | rank >1000
AP: 1.7 | rank 0 | rank 10 | rank 21 | rank 39 | rank 62 | rank 75 | rank 98 | rank >1000
AP: 16.8 | rank 0 | rank 4 | rank 38 | rank 55 | rank 99 | rank 92 | rank 273 | rank >1000
AP: 19.9 | rank 1 | rank 3 | rank 16 | rank 26 | rank 43 | rank 91 | rank 202 | rank 202
AP: 26.2 | rank 10 | rank 12 | rank 37 | rank 41 | rank 45 | rank 90 | rank >1000 | rank >1000
AP: 27.6 | rank 0 | rank 1 | rank 3 | rank 6 | rank 8 | rank 9 | rank 78 | rank 197
AP: 28.7 | rank 0 | rank 1 | rank 6 | rank 8 | rank 21 | rank >1000 | rank >1000 | rank >1000
AP: 33.3 | rank 0 | rank 3 | rank 41 | rank 51 | rank 92 | rank 99 | rank >1000 | rank >1000
AP: 38.1 | rank 0 | rank 1 | rank 2 | rank 12 | rank 46 | rank 46 | rank >1000 | rank >1000
AP: 50.0 | rank 0 | rank 1 | rank 2 | rank 4 | rank 7 | rank 95 | rank >1000 | rank >1000

Figure L. **Additional examples of queries, positives, and hard negatives within the distractor set based on i2i retrieval.** Average Precision per query and rank of the negatives and positives are reported using SigLIP[†]. **Gray:** queries. **Green:** positives. **Red:** distractors.

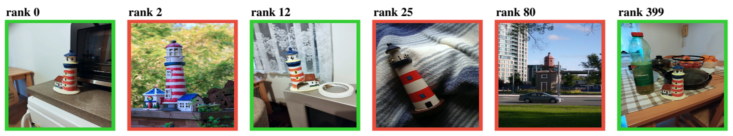
AP: 10.0

The image shows a page from a tear-off calendar. The page is yellow and features an illustration of a pair of orange sneakers with white laces. The date "16 veresnia" (September 16) is printed at the bottom in black text. The calendar is bound at the top with a blue cover that has metal fasteners.



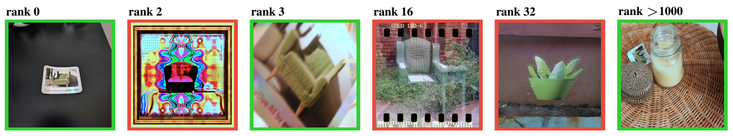
AP: 45.8

The image shows a small ceramic sculpture of a lighthouse. The lighthouse features red and white horizontal stripes and a blue top. Attached to the lighthouse is a small building with a brown roof. The sculpture is set on a light-colored base.



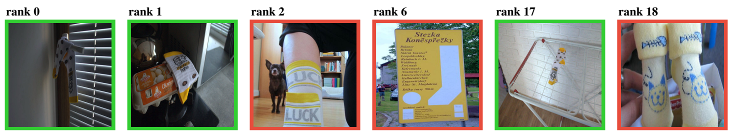
AP: 50.0

The sticker features an image of a green, textured armchair with wooden legs. The chair is positioned next to a tall cactus in a pot. The sticker has a holographic border with the text "Generative AI by gettyimages" at the bottom.



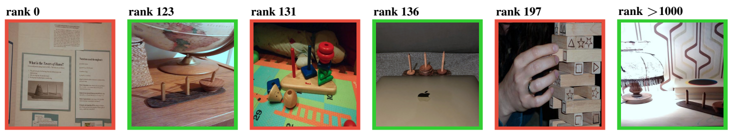
AP: 6.1

This is a white sock with yellow accents at the cuff, heel, and toe. It features a pattern of small dog images along the entire length. At the top of the sock, there is a logo of a cat and a dog, along with the text in Ukrainian "Home for Rescued Animals".



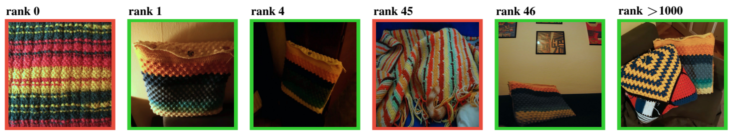
AP: 0.8

The image shows a wooden Tower of Hanoi puzzle. It consists of three vertical pegs mounted on a rectangular base. The central peg has a series of wooden discs stacked in decreasing size from bottom to top. The puzzle is typically used to demonstrate recursive problem-solving techniques.



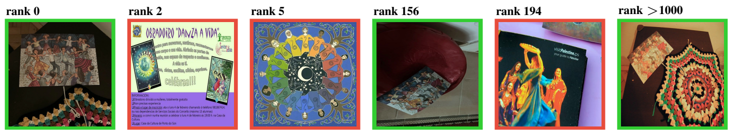
AP: 19.3

The image shows a crocheted textile with a textured pattern. It features horizontal stripes in various colors, including yellow, orange, dark blue, gray, green, and light yellow. The texture appears to be a bobble or popcorn stitch, giving it a raised, bumpy appearance.



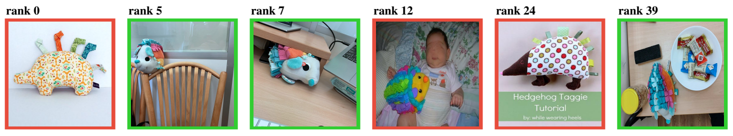
AP: 25.3

The image shows a colorful postcard featuring an illustration of people dancing in a circle. The figures are depicted in vibrant clothing, with a mosaic-like pattern on the ground and a starry night sky in the background. The scene conveys a sense of joy and celebration.



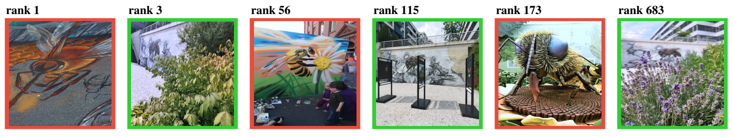
AP: 13.9

This is a stuffed toy resembling a hedgehog. It has a soft, beige body with a teal nose and ears. The toy features colorful fabric spikes in shades of teal, pink, and orange, adding a playful and vibrant touch. The eyes are black and round, giving it a cute appearance.



AP: 6.8

This is a mural depicting a large, detailed bee with realistic features, including its fuzzy body and translucent wings. The bee is set against a background of abstract, geometric shapes and flowers in grayscale, creating a striking contrast with the bee's natural colors. The artwork combines realism...



AP: 45.8

The image shows a graphic card case designed to look like a treasure chest with a monstrous theme. It features a skull with red eyes on top and sharp teeth lining the opening. Inside, a graphics card is visible. The case has a dark, weathered wood appearance with metal accents and a small skull...

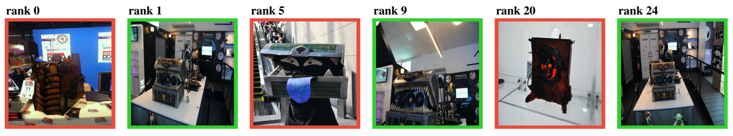


Figure M. **Examples of text queries, positives, and hard negatives within the distractor set based on t2i retrieval.** Average Precision per text query, and rank of the negatives and positives is reported using SigLIP. **Gray:** text queries. **Green:** positives. **Red:** distractors.

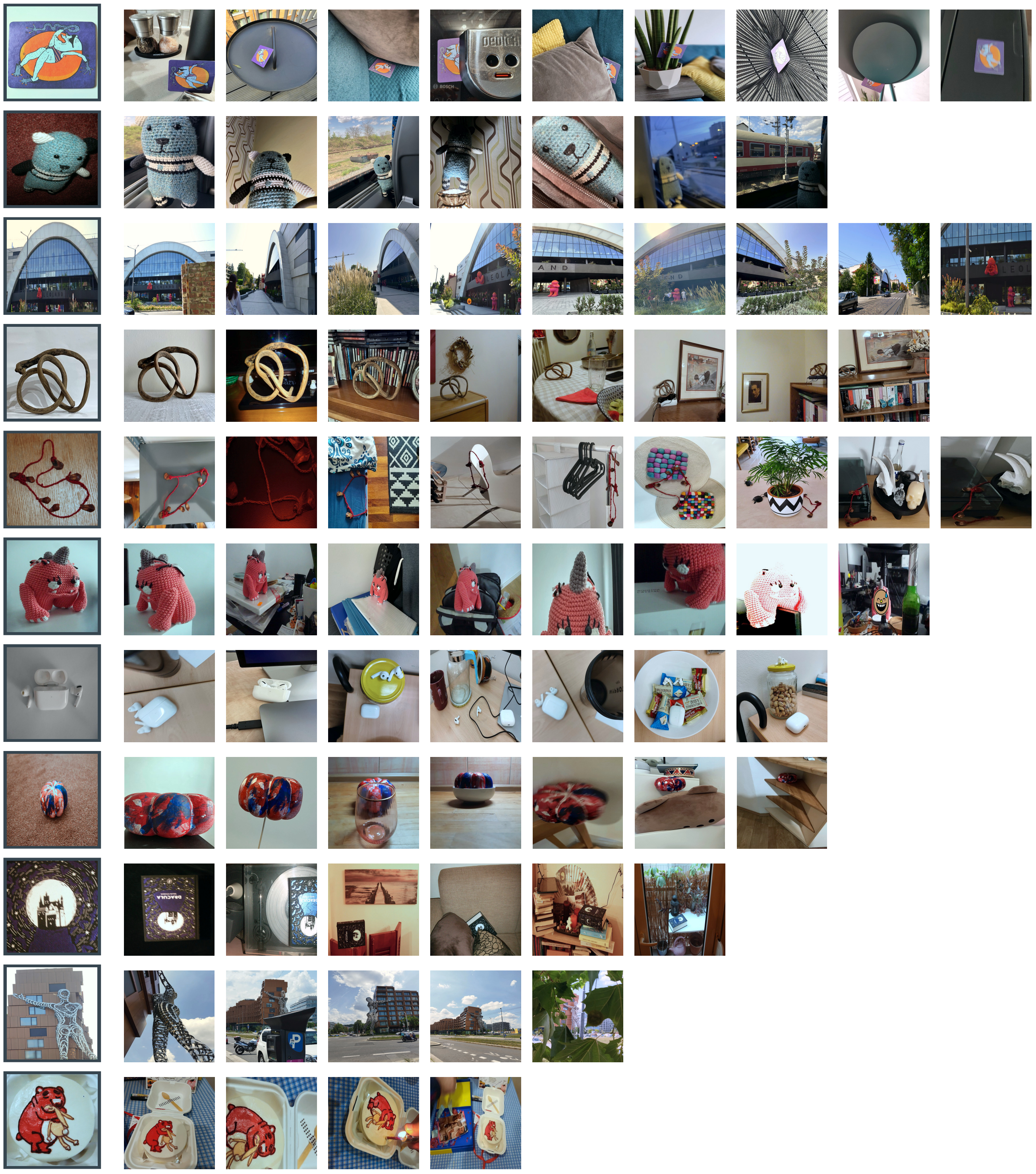


Figure N. **Examples of collected query objects.** Queries and multiple positives are displayed. **Gray:** queries.