Deep Learning for Instance Retrieval: A Survey

Wei Chen, Yu Liu, Weiping Wang, Erwin M. Bakker, Theodoros Georgiou,
Paul Fieguth, Li Liu, and Michael S. Lew

Abstract—In recent years a vast amount of visual content has been generated and shared from many fields, such as social media platforms, medical imaging, and robotics. This abundance of content creation and sharing has introduced new challenges, particularly that of searching databases for similar content — Content Based Image Retrieval (CBIR) — a long-established research area in which improved efficiency and accuracy are needed for real-time retrieval. Artificial intelligence has made progress in CBIR and has significantly facilitated the process of instance search. In this survey we review recent instance retrieval works that are developed based on deep learning algorithms and techniques, with the survey organized by deep network architecture types, deep features, feature embedding and aggregation methods, and network fine-tuning strategies. Our survey considers a wide variety of recent methods, whereby we identify milestone work, reveal connections among various methods and present the commonly used benchmarks, evaluation results, common challenges, and propose promising future directions.

Index Terms—Content based image retrieval, Instance retrieval, Deep learning, Convolutional neural networks, Literature survey

1 Introduction

Content Based Image Retrieval (CBIR) is the problem of searching for relevant images in an image gallery by analyzing the visual content (colors, textures, shapes, objects *etc.*), given a query image [1], [2]. CBIR has been a longstanding research topic in the fields of computer vision and multimedia [1], [2]. With the exponential growth of image data, the development of appropriate information systems that efficiently manage such large image collections is of utmost importance, with image searching being one of the most indispensable techniques. Thus there is a nearly endless potential for applications of CBIR, such as person/vehicle reidentification [3], [4], landmark retrieval [5], remote sensing [6], medical image search [7], online product searching [8], among many others.

Generally, CBIR methods can be grouped into two different tasks [9], [10]: Category level Image Retrieval (CIR) and Instance level Image Retrieval (IIR). The goal of CIR is to find an arbitrary image representative of the same category as the query (*e.g.*, dogs, cars) [11], [12]. By contrast, in the IIR task, a query image of a particular instance (*e.g.*, the Eiffel Tower, my neighbor's dog) is given and the goal is to find images containing the same instance that may be captured under different conditions like different imaging distances, viewing angles, backgrounds, illuminations, and weather conditions (reidentifying exemplars of the same instance) [13], [14]. The focus of this survey is the IIR task[1].

In many real world applications such as landmark retrieval and product searching, IIR is uaually a large-scale searching task, *i.e.*, finding a desired image requiring a search among thousands, millions, or even billions of images. Hence, searching efficiently is as critical as searching accurately, to which continued efforts have been devoted [13], [15], [16]. To enable accurate and efficient retrieval over a large-scale image collection, *developing compact yet discriminative feature representations* is at the core of IIR.

During the past two decades, startling progress has been witnessed in image representation which mainly consists of two important periods, *i.e.*, feature engineering and deep learning. In the feature engineering era, the field was dominated by various milestone handcrafted image representations like SIFT [24] and Bag of visual Words (BoW) [25]. The deep learning era was reignited in 2012 when a Deep Convolutional Neural Network (DCNN) referred as "AlexNet" [26] won the first place in the ImageNet classification contest with a breakthrough reduction in classification error rate. Since then, the dominating role of SIFT like local descriptors has been replaced by data driven Deep Neural Networks (DNNs) which can learn powerful feature representations with multiple levels of abstraction directly from data. During the past decade, DNNs have set the state of the art in various classical computer vision tasks, including image classification [26], [27], object detection [28], semantic segmentation [29], and image retrieval [18].

Given this period of rapid evolution, the goal of this paper is to provide a comprehensive survey of the recent achievements in IIR. In comparison with existing excellent surveys on traditional image retrieval [14], [19], [20], [21], as contrasted in Table 1, our focus in this paper is reviewing deep learning based methods for IIR, particularly on issues of retrieval accuracy and efficiency.

1.1 Summary of Progress since 2012

After the highly successful image classification implementation of AlexNet [26], significant exploration of DCNNs for retrieval tasks was undertaken, divided into three domains:

1) **Off the shelf models,** based on DCNNs with fixed parameters [30], [31], [32], as shown in Figure 1;
2) **Fine-tuned models,** based on DCNNs in which parameters are updated [30], also in Figure 1; and
3) **Effective feature use,** for which researchers have proposed embedding and aggregation methods, such as RMAC [33], CroW [16], and SPoC [13].

Building on the many methods shown in Figure 1, more recent progress for IIR can be categorized into the following

W. Chen is with College of Systems Engineering, NUDT, China and also with Leiden Institute of Advanced Computer Science, Leiden University, the Netherlands.
Y. Liu is with DUTRU International School of Information Science and Engineering, Dalian University of Technology, China.
W. Wang is with College of Systems Engineering, NUDT, China.
E. Bakker, T. Georgiou, and M. Lew are with Leiden Institute of Advanced Computer Science, Leiden University, the Netherlands.
P. Fieguth is with the Department of Systems Design Engineering, University of Waterloo, Canada.
L. Liu is with College of Systems Engineering, NUDT, China, and also with Center for Machine Vision and Signal Analysis, University of Oulu, Finland.
Corresponding author: Li Liu, li.liu@oulu.fi

1. If not further specified, "image retrieval", "IIR", and "instance retrieval" are considered equivalent and will be used interchangeably.

TABLE 1: A summary and comparison of the primary surveys in the field of image retrieval.

Title	Year	Published in	Main Content
Content-Based Image Retrieval at the End of the Early Years [1]	2000	TPAMI	This paper discusses the steps for image retrieval systems, including image processing, feature extraction, user interaction, and similarity evaluation.
Image Search from Thousands to Billions in 20 Years [17]	2013	TOMM	This paper gives a good presentation of image search achievements from 1970 to 2013, but the methods are not deep learning-based.
Deep Learning for Content-Based Image Retrieval: A Comprehensive Study [18]	2014	ACM MM	This paper introduces supervised metric learning methods for fine-tuning AlexNet. Details of instance-based image retrieval are limited.
Semantic Content-based Image Retrieval: A Comprehensive Study [19]	2015	JVCI	This paper presents a comprehensive study about CBIR using traditional methods; deep learning is introduced as a section with limited details.
Socializing the Semantic Gap: A Comparative Survey on Image Tag Assignment, Refinement, and Retrieval [20]	2016	CSUR	A taxonomy is introduced to structure the growing literature of image retrieval. Deep learning methods for feature learning is introduced as future work.
Recent Advance in Content based Image Retrieval: A Literature Survey [21]	2017	arXiv	This survey presents image retrieval from 2003 to 2016. Neural networks are introduced in a section and mainly discussed as a future direction.
Information Fusion in Content-based Image Retrieval: A Comprehensive Overview [22]	2017	Information Fusion	This paper presents information fusion strategies in CBIR. Deep convolutional networks for feature learning are introduced briefly but not covered thoroughly.
A Survey on Learning to Hash [23]	2018	TPAMI	This paper focuses on hash learning algorithms and introduces the similarity-preserving methods and discusses their relationships.
SIFT Meets CNN: A Decade Survey of Instance Retrieval [14]	2018	TPAMI	This paper presents a comprehensive review of instance retrieval based on SIFT and CNN methods.
Deep Learning for Instance Retrieval: A Survey	**2021**	**Ours**	**Our survey focuses on deep learning methods. We expand the review with indepth details on IIR, including methods of feature extraction, feature embedding and aggregation, and network fine-tuning.**

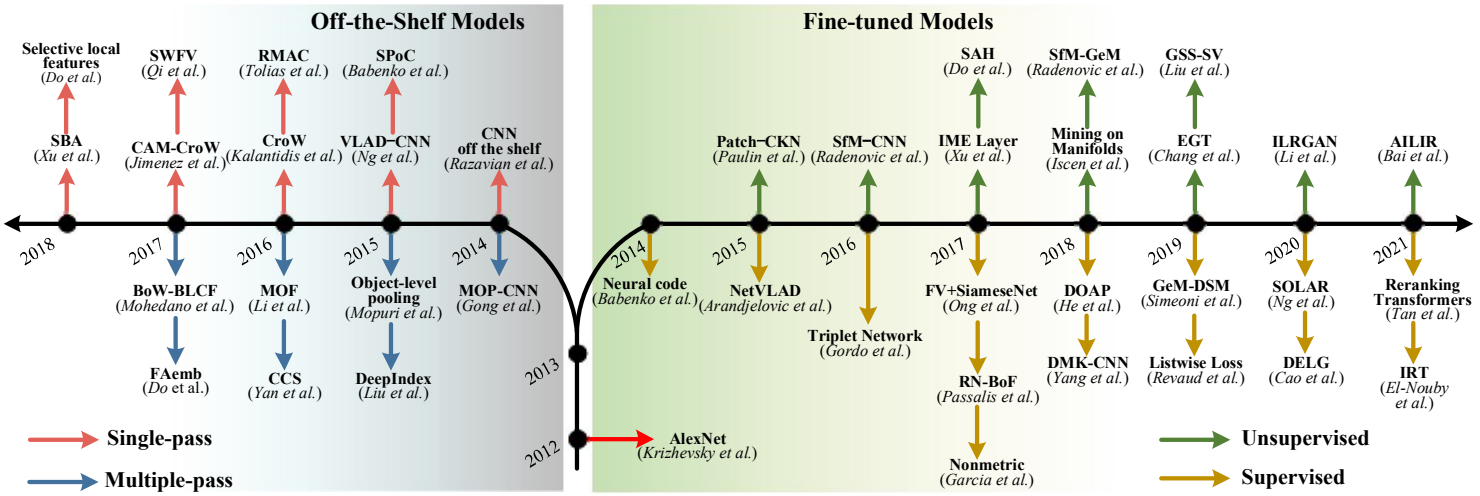


Fig. 1: Representative methods in IIR. Off-the-shelf models (left) have model parameters which are not further updated or tuned when extracting retrieval features. For single-pass schemes, the key step is the feature embedding and aggregation to promote the discriminativity of the extracted image-level activations [16], [34], [35], [36], whereas for multiple-pass schemes the goal is to extract instance features at region scales and eliminate image clutter as much as possible [31], [37], [38], [39]. In contrast, for fine-tuned models (right), the model parameters are updated for the features to be tuned towards the retrieval task and address the issue of domain shifts. For supervised fine-tuning, the key step lies in the design of objective functions and sample sampling strategies [40], [41] ,[42], [43], [44], while the success of unsupervised fine-tuning is to mine the relevance among training samples [45], [46], [47], [48], [49]. See Sections 4 and 5 for details.

four network-level and feature-level perspectives, which form the basis for this survey:

(1) *Improvements in Network Architectures* (Section 3)

What network architectures and variations have been proposed? DCNNs with more stacked layers (*i.e.*, with ever more linear convolution filters and non-linear activation functions) offer a stronger ability to extract high-level abstract and semantic-aware features [27], [50]. It is also possible to concatenate multi-scale features in parallel [51], or the use of transformers as an alternative to DCNNs [52], [53].

(2) *Deep Feature Extraction* (Section 4.1)

How are deep features actually extracted? The key step in IIR is to make the descriptors as semantic-aware as possible when using a given network architecture, usually realized by using features from FC layers [54], [55], or from convolutional layers [13], [33], or fusions of features, including layer-level [37], [56] and model-level fusions [57], [58]. Deep features can be extracted from the whole image or from image patches, which correspond to single-pass and multiple-pass schemes, respectively.

(3) *Feature Embedding and Aggregation* (Section 4.2)

How do we promote the discriminativity of the image-level or region-level activations extracted from a certain DCNN layer? Individual feature vectors can be mapped into a high-dimensional space [25], [59], [60] and then aggregated into a global feature. This mapping process can be realized by using a pre-trained codebook (*i.e.*, built separately) [54], [61] or learned as parameters during training (built simultaneously) [44], [62]. Features may be aggregated into a global feature by direct

pooling [22] or sophisticated pooling-based methods [13], [33] without feature embedding.

(4) *Network Fine-tuning for Learning Representations* (Section 5) DCNNs pretrained on source datasets for image classification are influenced by domain shifts when performing retrieval tasks on new datasets. Therefore, it is necessary to fine-tune deep networks to the specific domain [40], which can be realized by using supervised or unsupervised fine-tuning methods, of which representative methods are depicted in Figure 1.

1.2 Key Challenges

While significant progress has been witnessed in achieving *accuracy* and *efficiency*, both of these competing goals continue to present challenges:

1) Accuracy related challenges depend on the input data, the feature extractor, and the way in which the extracted features are processed:

- The instance in the query and the database images may be rotated, translated, or scaled differently, so the final features are affected by these transformations and retrieval accuracy may be degraded [31]. It is necessary to incorporate invariance into the IIR pipeline [63], [64].
- IIR systems may need to focus on only a certain object, or even only a small portion of an object. DCNNs may be affected by image clutter or background, so multiple-pass schemes have been examined where sophisticated region proposals are studied before feature extraction.
- Deep features for IIR are needed to be as discriminative as possible to distinguish instances with small differences, leading to many explorations in feature processing. These include feature fusion strategies, since features from different layers or networks have different semantic levels; feature embedding and feature aggregation, to promote feature discriminativity; and attention mechanisms, to highlight the most relevant regions within the extracted features or to enable deep networks to focus on regions of interest.
- DCNNs can be fine-tuned to be powerful feature extractors to capture fine semantic distinctions among instances. These explorations offer improved accuracy compared to the earlier work, however the area remains a major challenge.

2) Efficiency related challenges are important, especially for large-scale datasets [65]. Retrieval systems need to respond quickly when given a query image. Deep features are high-dimensional and contain semantic-aware information to support higher accuracy, yet this higher accuracy is often at the expense of efficiency. On the one hand, the retrieval efficiency is related to the format of feature vectors, *i.e.*, real valued or binary. Hash codes have advantages in storage and searching [23], [40], however for hashing methods one needs to carefully consider the loss function design [66], [67], to obtain optimal codes for high retrieval accuracy. On the other hand, efficiency is also related to the mechanism of feature matching. For example, instead of time-consuming cross-matching between local features, one can choose to use global features to perform an initial ranking and then a post-step re-ranking via the features of top-ranked images.

2 General Framework of IIR

We summarize a general framework of deep learning based IIR in Figure 2, involving four main stages.

1) Network feedforward scheme: There are two ways to feed images into DCNNs: single-pass and multiple-pass. Single-pass methods take as input the whole image, whereas multiple-pass methods depend on region extraction, such as spatial pyramid models [54] and region proposal networks [28], [43], as depicted in Figure 4. Multiple-pass scheme enables DCNNs to focus more on specific instances but are also more time-consuming [14].

2) Deep feature extraction (Section 4.1): Based on the input image or just some patches, the network activations are used as vanilla features [68]. Fully-connected layers offer a global receptive field and can generalize an entire image with a single vector. Convolutional layers demonstrate an advantage for IIR because their output tensors can preserve structural information when further processed by sophisticated pooling [13], [69], [70]. It is possible to fuse the activations from both kinds of layers, including layer-level [37], [56] and model-level fusions [57], [58].

3) Feature embedding and aggregation (Section 4.2): Based on the image-level or region-level descriptors, there are two essential steps to produce global or local features, usually followed by PCA and whitening. Feature embedding maps individual local features as a higher-dimensional vector, whereas feature aggregation summarizes the multiple mapped vectors into a single vector. Global representations may come from pooling convolutional feature maps [69], [71] or using some sophisticated weighting methods [13], [16]. For local features, the representations for all regions of interest are stored individually and used for cross-matching in the reranking stage.

4) Feature matching is a process to measure the feature similarity between images, and then return a ranked list. Global matches can be computed efficiently via Euclidean, Hamming, or some other distance metric. For local features [5], [72], the image similarity is usually evaluated by summarizing the similarities across local features, using classical RANSAC [73] or more recent variations [74], [75]. Storing local features separately and then estimating their similarity individually lead to additional memory and search costs [72], [75], therefore in most cases local features are used to re-rank the initial ranking image matched by global features [33], [66], [72], [76].

The four preceding stages for IIR (from the input data to the output ranking list) rely on using DCNNs as backbone architectures. Undoubtedly, pre-stored parameters in these backbones also can be fine-tuned (Section 5) to be better suited for instance retrieval and contribute to better performance.

3 Popular Backbone DCNN Architectures

The hierarchical structure and extensive parameterization of DCNNs has led to their success in a remarkable diversity of vision tasks. For IIR, there are four network models that serve as the basis for feature extraction: AlexNet [26], VGG [50], GoogLeNet [51], and ResNet [27].

AlexNet is the first DCNN which improved ImageNet classification accuracy by a significant margin compared to conventional methods in ILSVRC 2012. It consists of 5 convolutional layers and 3 fully-connected layers. Input images are usually resized to a fixed size during training and testing.

Inspired by AlexNet, VGGNet has two widely used versions (VGG-16, VGG-19) having 13 or 16 convolutional layers, where the convolutional filters are small (local). VGGNet is trained in a multi-scale manner where training images are cropped and re-scaled, improving feature invariance for retrieval.

Compared to AlexNet and VGGNet, GoogLeNet is deeper and wider but has fewer parameters within its 22 layers, leading to higher learning efficiency. Inception modules, each consisting

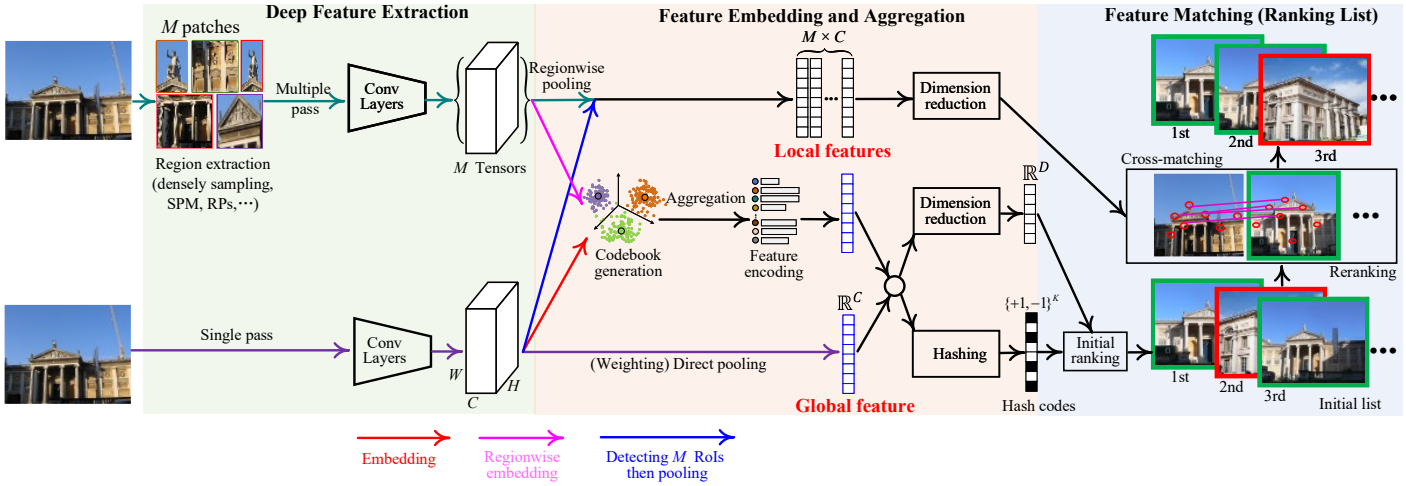


Fig. 2: General framework of IIR, which includes feature extraction on image or image patches, followed by feature embedding and aggregation methods to improve feature discriminativity. Feature matching can be performed by using global features (initial filtering) or use local features to rerank the top-ranked images matched by global features.

of branches having multiple local filters, are concatenated spatially to obtain the final features for each module. Deeper architectures are beneficial for learning higher-level abstract features to mitigate the semantic gap [20].

ResNet added further convolutional layers to extract more abstract features, and skip connections were added between convolutional layers to address the notorious vanishing gradient problem during training.

DCNN architectures have developed significantly during the past few years, for which we refer the reader to recent surveys [77], [78], [79]. Recently, transformer architectures, having a self-attention mechanism as a key ingredient, have been used as an alternative to DCNNs to perform instance retrieval [52], [53], [80].

The following sections will introduce details about deep learning for instance-level image retrieval, for which a detailed sub-categorization is shown in Figure 3.

4 Retrieval with Off-the-Shelf DCNN Models

Because of their size, deep CNNs need to be trained on exceptionally large-scale datasets, and the available datasets of such size are those for image recognition and classification. One possible scheme, then, is that deep models effectively trained for recognition and classification directly serve as off-the-shelf feature detectors for the image retrieval task, the topic of this survey. That is, one can propose to undertake image retrieval on the basis of DCNNs, trained for classification and with their pre-trained parameters frozen.

There are limitations to this approach, most fundamentally that there is a model-transfer or domain-shift challenge between tasks [14], [32], [88], meaning that models trained for classification do not necessarily extract features well suited to image retrieval. In particular, a classification decision can be successful as long as features remain within classification boundaries, however features from such models may show insufficient capacity for retrieval where feature matching itself is more important than classification. This section will survey the strategies which have been developed to improve the quality of feature representations, particularly based on feature extraction / fusion (Section 4.1) and feature embedding / aggregation (Section 4.2).

- Deep Learning for Instance-level Image Retrieval
  - Retrieval with Off-the-Shelf DCNN Models (Section 4)
    - **Deep Feature Extraction (Section 4.1)**
      - Network Feedforward Scheme (Section 4.1.1)
        - Single Feedforward Pass: MAC [69], R-MAC [33]
        - Multiple Feedforward Pass: SPM [54], RPNs [43]
      - Deep Feature Selection (Section 4.1.2)
        - Fully-connected Layer: Neural codes [40]
        - Convolutional Layer: SPoC [13], CroW [16]
      - Feature Fusion Strategy (Section 4.1.3)
        - Layer-level Fusion: MoF [37], MOP [31]
        - Model-level Fusion: ConvNet fusion [50]
    - **Feature Embedding and Aggregation (Section 4.2)**
      - Matching with Global Features (Section 4.2.1)
      - Matching with Local Features (Section 4.2.2)
      - Attention Mechanism (Section 4.2.3)
        - Non-parameteric: CroW [16], SPoC [13], SWVF [81]
        - Parameteric: CRN [82], DeepFixNet+SAM [83], [84]
      - Deep Hash Embedding (Section 4.2.4)
        - Supervised Hashing: SSDH [85]
        - Unsupervised Hashing: DeepBit [67], DSTH [86]
  - Retrieval via Learning DCNN Representations (Section 5)
    - **Supervised Fine-tuning (Section 5.1)**
      - Fine-tuning via classification loss (Section 5.1.1)
      - Fine-tuning via pairwise ranking loss (Section 5.1.2)
        - Transformation Matrix: Non-metric [41]
        - Siamese Networks [42]
        - Triplet Networks [87]
    - **Unsupervised Fine-tuning (Section 5.2)**
      - Manifold Learning Samples Mining: Diffusion Net [48]
      - Mining Samples by Clustering: SfM-GeM [45], [47]

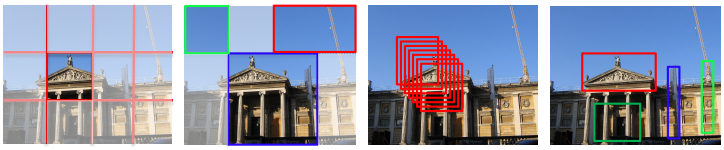
Fig. 3: This survey is organized around four key aspects in instance-level image retrieval, shown in boldface.

4.1 Deep Feature Extraction

The primary issue of feature extraction is the mechanism by which retrieval features can be extracted from off-the-shelf DCNNs, which primarily involves three aspects: network feedforward scheme, feature selection, and feature fusion.

4.1.1 Network Feedforward Scheme

Network feedforward schemes focus on how images are fed into a DCNN, which includes single-pass and multiple-pass.



(a) (b) (c) (d)

Fig. 4: Image patch generation schemes: (a) Rigid grid; (b) Spatial pyramid modeling (SPM); (c) Dense patch sampling; (d) Region proposals (RPs) from region proposal networks.

a. Single Feedforward Pass Methods.

Single feedforward pass methods take the whole image and feed it into an off-the-shelf model to extract features. The approach is relatively efficient since the input image is fed only once. For these methods, both the fully-connected layer and last convolutional layer can be used as feature extractors [68].

Early network-based IIR work focused on leveraging DCNNs as a fixed extractor to obtain global features, especially based on the fully-connected layers [30], [40], requiring close to zero engineering effort. However, extracting features in this way affects retrieval accuracy since the extracted features may include background information or activations for irrelevant objects.

The key to single-pass schemes is to embed and aggregate deep features to improve their discriminativity, such that features of two related images (*i.e.*, including the same object) are more similar than these of two unrelated images [13]. For this purpose, it is possible to first map the features from a fully-connected layer into a high-dimensional feature space and then to aggregate them into a final global feature [31]. Another direction is to treat regions in convolutional feature maps as different sub-vectors, such that a combination of sub-vectors of all feature maps are used to represent the input image [16], [33].

b. Multiple Feedforward Pass Methods.

Compared to single-pass schemes, multiple-pass methods are more time-consuming [14] because several patches are generated and then fed into the network before being aggregated as a final global feature.

The representations are usually produced from two stages: patch detection and patch description. Multi-scale image patches are obtained using sliding windows [31], [37], [38], [39] or spatial pyramid model (SPM) [54], [89], [90], as shown in Figure 4. For example, Zheng *et al*. [90] partition an image by using SPM and extract features at increasing scales, thus enabling the integration of global, regional, local contextual information.

Patch detection methods lack retrieval efficiency since irrelevant patches are also detected [33]. For example, Cao *et al*. [91] propose to merge image patches into larger regions with different hyper-parameters, where the hyper-parameter selection is viewed as an optimization problem to maximize the similarity between query and candidate features.

Instead of generating multi-scale image patches randomly or densely, region proposal methods introduce a degree of purpose. Region proposals can be generated using object detectors, such as selective search [63], edge boxes [92], [93], and BING [94]. For example, Yu *et al*. [93] propose fuzzy object matching (FOM) for instance search in which the fuzzy objects are generated from 300 object proposals and then clustered to filter out overlapping proposals. Region proposals can also be learned using such as region proposal networks (RPNs) [28], [43] and convolutional kernel networks (CKNs) [95], and then to apply these networks into end-to-end fine-tuning for learning similarity [96].

4.1.2 Deep Feature Selection

Feature selection decides the receptive field of the extracted features, *i.e.*, global-level from fully-connected layers and regional-level from convolutional layers.

a. Extracted from Fully-connected Layers

It is straightforward to select a fully-connected layer as a global feature extractor [30], [31], [40]. With PCA dimensionality reduction and normalization [30] image similarity can be measured. Since the layer is fully connected, each neuron produces image-level descriptors, but which leads to two obvious limitations for IIR: including irrelevant information, and a lack of local geometric invariance [31].

With regards to the first limitation, image-level global descriptors may include irrelevant patterns or background clutter, especially when a target instance is only a small portion of an image. It may then be more reasonable to extract region-level features at finer scales, *i.e.*, using multiple feedforward passes [30], [66]. For the second limitation, a lack of local geometric invariance affects the robustness to image transformations such as truncation and occlusion. Further, it makes the global feature incompatible with techniques such as spatial verification and re-ranking. Several methods then choose to leverage intermediate convolutional layers [13], [31], [61], [69].

b. Extracted from Convolutional Layers

Features from convolutional layers (usually the last layer) preserve more structural details which are especially beneficial for instance retrieval [69]. The neurons in a convolutional layer are connected only to a local region, and this smaller receptive field ensures that the produced features preserve more local structural information [70], [97] and are more robust to image transformations [13]. Many image retrieval methods use convolutional layers as feature extractors [34], [61], [69], [98].

Sum/average and max pooling are two simple aggregation methods to produce global features [69], [71]. For a pooled layer, the last convolutional layer usually yields superior accuracy over other shallower or later fully-connected layers [97]. There is no other operation on the feature maps before pooling, so we illustrate these methods as "direct pooling" in Figure 2.

Instead of direct pooling, many sophisticated aggregation methods have been explored, such as channel-wise or spatial-wise feature weighting on the convolutional feature maps [64], [99], [100]. These aggregation methods aim to highlight feature importance [16] or reduce the undesirable influence of bursty descriptors of some regions [35], [101]. For clarity, we illustrate the representative strategies in Figure 5, including MAC [69], R-MAC [33], GeM pooling [47], SPoC [13], CroW [16], and CAM+CroW [34]. Note that the feature aggregation of Figure 5 is usually performed before channel-wise sum/max pooling and does not embed features into a higher dimensional space.

One rationale behind using convolutional feature maps is that each such vector can act as a "dense SIFT" feature [13] since each vector corresponds to a region in the input image. Inspired by this perception, many works leverage embedding methods used for SIFT features [24], including BoW, VLAD, and FV, on the regional feature vectors and then aggregate them (*e.g.*, by sum pooling) into a global descriptor. Before aggregation, the feature embedding improves the discriminativity of individual features by mapping into a high-dimensional space [35]. The feature embedding is followed by PCA to reduce feature dimensionality and whitening to down-weight co-occurrence between features.

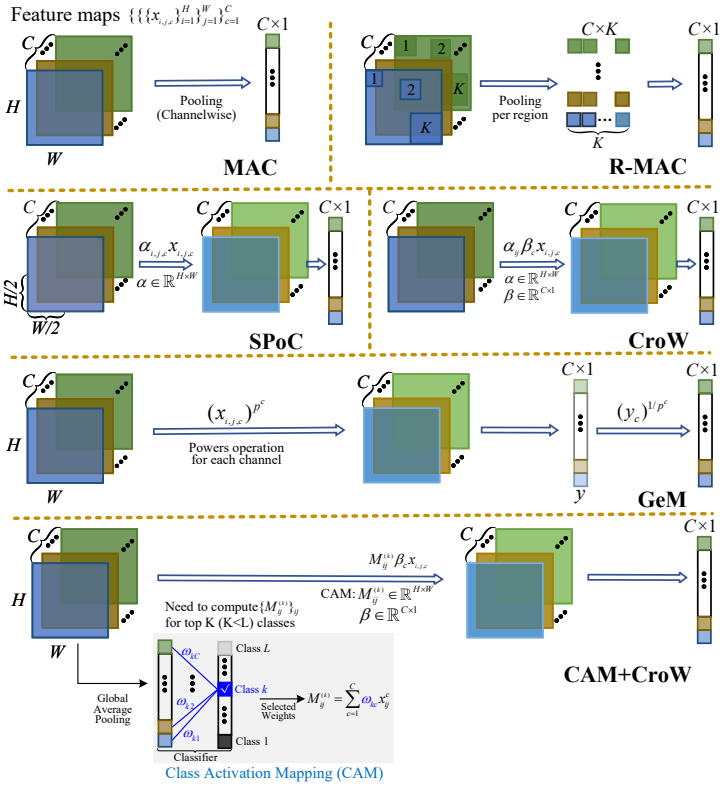


Fig. 5: Representative methods in single-pass methods, focusing on convolutional feature maps x with size $H \times W \times C$: MAC [69], R-MAC [33], GeM [47], SPoC with the Gaussian weighting scheme [13], CroW [16], and CAM+CroW [34].

4.1.3 Feature Fusion Strategies

Fusion studies the complementarity of different features which includes layer-level and model-level fusion explorations.

a. Layer-level Fusion

With layer-level fusion it is possible to fuse multiple fully-connected layers in a deep network [54], [56]. For instance, Liu *et al*. [54] introduce DeepIndex to incorporate multiple global features from different fully connected layers. The activation from the first fully-connected layer (*e.g*., VGG net) is taken as column indexing, and that from the second layer serves as row indexing. Similarly, it is also possible to fuse the activations from multiple convolutional layers. For instance, Li *et al*. [99] apply the R-MAC encoding scheme on five convolutional layers of VGG-16 and then concatenate them into a multi-scale feature vector.

Features from fully-connected layers retain global high-level semantics, whereas features from convolutional layers can present local low- and intermediate-level cues. Global and local features therefore complement each other when measuring semantic similarity and can, to some extent, guarantee retrieval performance [102], [103]. Such features can be concatenated directly [72], [102], with convolutional features normally filtered by sliding windows or region proposal nets. Direct concatenation can also be replaced by other advanced methods, such as orthogonal operations [103] or pooling-based methods, such as Multi-layer Orderless Fusion (MOF) of Li *et al*. [37], which is inspired by Multi-layer Orderless Pooling (MOP) [31]. However local features cannot play a decisive role in distinguishing subtle feature differences if global and local features are treated identically. Yu *et al*. [102] use a mapping function to assert local features in refining the return ranking lists, via an exponential mapping function for tapping the complementary strengths of convolutional and fully-connected layers. Similarly, Liu *et al*. [4] design two sub-networks on top of convolutional layers to obtain global and local features and then learn to fuse these features, thereby adaptively adjusting the fusion weights. Instead of directly fusing the layer activations, Zhang *et al*. [104] fuse the index matrices which are generated based on the two feature types extracted from the same CNN, a feature fusion which has low computational complexity.

It is worth considering which layer combinations are better for fusion given their differences and complementarity. Yu *et al*. [102] compare the performance of different combinations between fully-connected and convolutional layers on the Oxford 5k, Holiday, and UKBench datasets. The results show that the combinations including the first fully-connected layer always perform better. Li *et al*. [37] demonstrate that fusing convolutional and fully-connected layers outperforms the fusion of only convolutional layers. Fusing two convolutional layers with one fully-connected layer achieves the best performance on the Holiday and UKBench datasets.

b. Model-level Fusion

It is possible to combine features from different models; such a fusion focuses more on model complementarity, with methods categorized into *intra-model* and *inter-model*.

Intra-model fusion suggests multiple deep models having similar or highly compatible structures, while inter-model fusion involves models with differing structures. For example, the widely-used dropout strategy in AlexNet [26] can be regarded as intra-model fusion: with random connections of different neurons between two fully-connected layers, each training epoch can be viewed as the combinations of different models. As a second example, Simonyan *et al*. [50] introduce a ConvNet intra-model fusion strategy to improve the feature learning capacity of VGG where VGG-16 and VGG-19 are fused. To attend to different parts of an image object, Wang *et al*. [105] realize the multi-feature fusion by selecting all convolutional layers of VGG-16 to extract image representations, which is demonstrated to be more robust than using only single-layer features.

Inter-model fusion is a way to bridge different features given the fact that different deep networks have different receptive fields [54], [58], [84], [97]. For instance, a two-stream attention network [84] is introduced to implement image retrieval where the main network for semantic prediction is VGG-16 while an auxiliary network is used for predicting attention maps. Similarly, considering the importance and necessity of inter-model fusion to bridge the gap between mid-level and high-level features, Liu *et al*. [54] and Zheng *et al*. [97] combine VGG-19 and AlexNet to learn combined features, while Ozaki *et al*. [58] concatenate descriptors from six different models.

Inter-model and intra-model fusion are relevant to model selection. There are some strategies to determine how to combine the features from two models. It is straightforward to fuse all features from the candidate models and then learn a metric based on the concatenated features [54], [84], which is a kind of "*early fusion*" strategy. Alternatively, it is also possible to learn optimal metrics separately for the features from each model, and then to combine these metrics for final retrieval ranking [37], [106], which is a kind of "*late fusion*" strategy.

Discussion. Layer-level fusion and model-level fusion are conditioned on the fact that the associated components (layers or whole networks) have different feature description capacities. For these two fusion strategies, the key question is *what features are the best to be combined?* Some explorations have been made on

the basis of off-the-shelf deep models, such as Xuan *et al*. [107], whose work illustrates the effect of combining different numbers of features and different sizes within the ensemble. Chen *et al*. [108] analyze the performance of embedded features from off-the-shelf image classification and object detection models with respect to image retrieval.

4.2 Feature Embedding and Aggregation

The primary aim of feature embedding and aggregation is to further promote the discriminativity of extracted activations from DCNNs, to obtain final global and/or local features for retrieving specific instances.

4.2.1 Matching with Global Features

Global features can be extracted from fully-connected layers, followed by dimensionality reduction and normalization [30], [40]. They are easy to implement and there is no further aggregation process. Gong *et al*. [31] extract fully-connected activations for local image patches at three scale levels and embed patch-level activations individually using VLAD. The final concatenated features have a strong invariance to image rotations.

Convolutional feature maps can also be aggregated into compact global features. Simple aggregation methods are sum/average or max pooling [69], [71]. Sum/average pooling is less discriminative, because it takes into account all activated outputs from a convolutional layer, weakening the effect of highly activated features [35]. As a result, max pooling is particularly well suited for sparse features having a low probability of being active, however max pooling may be inferior to sum/average pooling when image features are whitened [13].

Figure 5 illustrates sophisticated feature aggregation methods using channel-wise or spatial-wise weighting [13], [64]. For example, Babenko *et al*. [13] propose sum-pooling convolutional features (SPoC) to obtain compact descriptors pre-processed with a Gaussian center prior. Similarly, it is possible to treat regions in feature maps as different sub-vectors [33], [69], [97], thus combinations of sub-vectors are used to represent the input image, such as R-MAC [33]. Since convolutional features may include repetitive patterns and each vector may correspond to identical regions, the resulting descriptors may be bursty, which makes the final aggregated global feature less distinguishable. As a solution, Pang *et al*. [101] leverage heat diffusion to weigh convolutional features at the aggregation stage, and reduce the undesirable influence of burstiness.

Convolutional features have an interpretation as descriptors of local regions, thus many works leverage embedding methods, including BoW, VLAD, and FV, to encode regional feature vectors and then aggregate them (*e.g.*, simply by a sum operation) into a global descriptor. Note that BoW and VLAD can be extended by using other metrics, such as a Hamming distance [109]. Here we briefly describe the principle of Euclidean embeddings.

BoW [25] is a widely used feature embedding which leads to a sparse vector of occurrence. Let $X = \{x_1, x_2, ..., x_T\}$ be a set of local features, each of dimensionality D. BoW requires a pre-defined codebook $C = \{c_1, c_2, ..., c_K\}$ with K centroids, usually learned offline, to cluster these local descriptors, and maps each descriptor x_t to the nearest centroid c_k. For each centroid, one can count and normalize the number of occurrences as

$$g(c_k) = \frac{1}{T}\sum_{t=1}^{T}\phi(x_t, c_k) \tag{1}$$

$$\phi(x_t, c_k) = \begin{cases} 1 & \text{if } c_k \text{ is the closest codeword for } x_t \\ 0 & \text{otherwise} \end{cases} \tag{2}$$

Thus BoW considers the number of descriptors belonging to each c_k (*i.e.* 0-order feature statistics), so the BoW representation is the concatenation of all mapped vectors:

$$G_{_{BoW}}(X) = \left[\ g(c_1), \cdots, g(c_k), \cdots, g(c_K)\ \right]^\top \tag{3}$$

BoW is simple to implement the encoding of local descriptors, such as convolutional feature maps [37], [68] or fully-connected activations [90], [104], or to encode regional descriptors [110], [111]. Mukherjee *et al*. [111] extract image patches based on information entropy and feed into a pre-trained VGG-16, then use BoW to embed and aggregate the patch-level descriptors from a fully-connected layer. Embedded BoW vectors are typically high-dimensional and sparse, so not well suited to large-scale datasets in terms of efficiency.

VLAD [59] stores the sum of residuals for each visual word. Similar to BoW, it generates K visual word centroids, then each feature x_t is assigned to its nearest visual centroid c_k:

$$g(c_k) = \frac{1}{T}\sum_{t=1}^{T}\phi(x_t, c_k)(x_t - c_k) \tag{4}$$

The VLAD representation is stacked by the residuals for all centroids, with dimension $(D \times K)$, *i.e.*,

$$G_{_{VLAD}}(X) = \left[\ \cdots, g(c_k)^\top, \cdots\ \right]^\top. \tag{5}$$

VLAD captures first-order feature statistics, *i.e.*, $(x_t - c_k)$. Similar to BoW, the performance of VLAD is affected by the number of clusters: more centroids produce larger vectors that are harder to index. For instance-level image retrieval, Gong *et al*. [31] concatenate the activations of a fully-connected layer with VLAD applied to image-level and patch-level inputs [112]. Ng *et al*. [61] replace BoW [25] with VLAD [59], and are the first to encode local features into VLAD representations. This idea inspired another milestone work [44] where, for the first time, VLAD is plugged into the last convolutional layer, which allows end-to-end training via back-propagation.

FV [60] extends BoW by encoding the first and second order statistics. FV clusters the set of local descriptors by a Gaussian Mixture Model (GMM) with K components to generate a dictionary $C = \{\mu_k; \Sigma_k; w_k\}_{k=1}^{K}$ made up of mean / covariance / weight triples [113], where the covariance may be simplified by keeping only its diagonal elements. For each local feature x_t, a GMM is given by

$$\gamma_k(x_t) = w_k \times p_k(x_t) / \Big(\sum_{j=1}^{K} w_j p_j(x_t)\Big) \quad \sum_{j=1}^{K} w_k = 1 \tag{6}$$

where $p_k(x_t) = \mathcal{N}(x_t, \mu_k, \sigma_k^2)$. All local features are assigned into each component k in the dictionary, which is computed as

$$\begin{aligned} g_{w_k} &= \frac{1}{T\sqrt{w_k}}\sum_{t=1}^{T}\Big(\gamma_k(x_t) - w_k\Big) \\ g_{u_k} &= \frac{\gamma_k(x_t)}{T\sqrt{w_k}}\sum_{t=1}^{T}\left(\frac{x_t - \mu_k}{\sigma_k}\right), \\ g_{\sigma_k^2} &= \frac{\gamma_k(x_t)}{T\sqrt{2w_k}}\sum_{t=1}^{T}\left[\left(\frac{x_t - \mu_i}{\sigma_k}\right)^2 - 1\right] \end{aligned} \tag{7}$$

The FV representation is produced by concatenating vectors from the K components:

$$G_{_{FV}}(X) = \left[\ g_{w_1}, \cdots, g_{w_K}, g_{u_1}, \cdots, g_{u_K}, g_{\sigma_1^2}, \cdots, g_{\sigma_K^2}\ \right]^\top \tag{8}$$

The FV representation defines a kernel from a generative process and captures more statistics than BoW and VLAD. FV vectors do not increase the computational cost significantly but require more memory. Applying FV without memory controls may lead to suboptimal performance [114].

Discussion. Traditionally, pooling-based aggregation methods (*e.g.*, in Figure 5) are directly plugged into deep networks and then the whole model is used end-to-end. The three embedding methods (BoW, VLAD, FV) are initially trained with large pre-defined vocabularies [54], [115]. One needs to pay attention on their properties before choosing an embedding: BoW and VLAD are computed in the rigid Euclidean space where performance is closely related to the number of centroids, whereas FV can capture higher-order statistics and improves the effectiveness of feature embedding at the expense of a higher memory cost. Further, although vocabularies are usually built separately and pre-trained before encoding deep features, it is necessary to integrate the training of networks and the learning of vocabulary parameters into a unified framework so as to guarantee training and testing efficiency. For example, VLAD is integrated into deep networks where each spatial column feature is used to construct clusters via k-means [61]. This idea led to NetVLAD [44], where deep networks are fine-tuned with the VLAD vectors. The FV method is also combined with deep networks for retrieval tasks [35], [42].

4.2.2 Matching with Local Features

Although matching with global features has high efficiency for both feature extraction and similarity computation, global features are not compatible with spatial verification and correspondence estimation, which are important procedures for instance-level retrieval tasks, motivating work on matching with local features. In terms of the matching process, global features are matched only once while local feature matching is evaluated by summarizing the similarity across all individual local features (*i.e.*, many-to-many matching).

One important aspect of local features is to detect the keypoints for an instance within an image, and then to describe the detected keypoints as a set of local descriptors. Inspired by [116], the common strategies of this whole procedure for IIR can be categorized as *detect-then-describe* and *describe-then-detect*.

In terms of *detect-then-describe*, we regard the descriptors around keypoints as local features, similar to [30], [43]. Coarse regions can be detected, for example, by using the methods depicted in Figure 4, and regions of interest in an image can be detected by using region proposal networks (RPNs) [28], [75]. The extracted coarse regions around the keypoints are fed into a DCNN, followed by feature description. Traditional detectors can also be used to detect fine regions around a keypoint. For instance, Zheng *et al*. [90] employ the popular Hessian-Affine detector [117] to get an affine-invariant local region. Paulin *et al*. [112] and Mishchuk *et al*. [110] detect regions using the Hessian-Affine detector and feed into patch-convolutional kernel networks (Patch-CKNs) [95].

Rather than performing keypoint detection early on, it is possible to postpone the detection stage on the convolutional feature maps, *i.e.*, *describe-then-detect*. One can select regions on the convolutional feature maps to obtain a set of local features [33], [38], [68]; the local maxima of the feature maps are then detected as keypoints [62]. A similar strategy is also used in network fine-tuning [5], [52], [72], [75], [118], where the keypoints on the convolutional feature maps can be selected based on attention scores predicted by an attention network [72], [5], or based on single-head and multi-head attention modules in transformers [52], [53]. This approach to keypoint selection is better for achieving computational efficiency.

After keypoint detection and description, a large number of local features are used in the matching stage to perform instance-level retrieval, and the image similarity is evaluated by matching across all local features. Local matching techniques include spatial verification and selective match kernels (SMK) [74]. Spatial verification assumes object instances are rigid so that local matches between images can be estimated as an affine transformation using RANdom SAmple Consensus (RANSAC) [73]. One limitation of RANSAC is its high computational complexity of estimating the transformation model when all local descriptors are considered; instead, it is possible to apply RANSAC to a small number of top-ranked local descriptors, such as those selected by approximate nearest neighbor [5]. SMK weighs the contributions of individual matches with a non-linear selective function, but is still memory intensive. Its extension, the Aggregated Selective Match Kernel (ASMK), focuses more on aggregating similarities between local features without explicitly modeling the geometric alignment, which can produce a more compact representation [74], [118]. Recently, Teichmann *et al*. [75] introduced Regional Aggregated Selective Match Kernel (R-ASMK) to combine information from detected regions, boosting image retrieval accuracy compared to the ASMK.

Discussion. Using local descriptors to perform instance retrieval tasks has two limitations. First, the local descriptors for an image are stored individually and independently, which is memory-intensive, and not well-suited for large-scale scenarios. Second, estimating the similarity between the query and database images depends on cross-matching all local descriptor pairs, which incurs additional searching cost. Therefore, most instance retrieval systems using local features follow a two-stage paradigm: initial filtering and re-ranking [66], [72], [76], [96], [119], as in Figure 2. The initial filtering stage is to employ a global descriptor to select a set of candidate matching images, thereby reducing the solution space; the re-ranking stage is to use local descriptors to re-rank the top-ranked images from the global descriptor.

4.2.3 Attention Mechanism

Attention mechanism can be regarded as a kind of feature aggregation, whose core idea is to highlight the most relevant feature parts, realized by computing an attention map. Approaches to obtaining attention maps can be categorized into two groups: non-parametric and parametric, as shown in Figure 6, where the main difference is whether the importance weights in the attention map are learnable.

Non-parametric weighting is a straightforward method to highlight feature importance, and the corresponding attention maps can be obtained by channel-wise or spatial-wise pooling, as in Figure 6 (a,b). For spatial-wise pooling, Kalantidis *et al*. [16] propose an effective CroW method to weight and pool feature maps, which concentrate on weighting activations at different spatial locations, without considering the relations between these activations. In contrast, Ng *et al*. [70] explore the correlations among activations at different spatial locations on the convolutional feature maps.

Channel-wise weighting methods are also popular non-parametric attention mechanisms [36], [100]. Xu *et al*. [36] rank the weighted feature maps to build "probabilistic proposals" to select regional features. Jimenez *et al*. [34] combine CroW and R-MAC to propose Classes Activation Maps (CAM) to weigh the

feature map per class. Xiang *et al*. [100] employ a Gram matrix to analyze the correlations between different channels and then obtain channel sensitivity information to tune the importance of each feature map. Channel-wise and spatial-wise weighting methods are usually integrated into a deep model to highlight feature importance [16], [105].

Parametric attention maps, shown in Figure 6 (c,d), can be learned via deep networks, where the input can be either image patches or feature maps [82], [100], [103], approaches which are commonly used in supervised metric learning [98]. Kim *et al*. [82] make the first attempt to propose a shallow network (CRN) to take as input the feature maps of convolutional layers and outputs a weighted mask indicating the importance of spatial regions in the feature maps. The resulting mask modulates feature aggregation to create a global representation of the input image. Noh *et al*. [5] design a 2-layer CNN with a softplus output layer to compute scores which indicate the importance of different image regions. Inspired by R-MAC, Kim *et al*. [120] employ a pre-trained ResNet101 to train a context-aware attention network using multi-scale feature maps.

Apart from using feature maps as inputs, a whole image can be used to learn feature importance, for which specific networks are needed [83], [84], [121]. Mohedano [83] explores different saliency models, including DeepFixNet and Saliency Attentive Model. Yang *et al*. [84] and Wei *et al*. [121] introduce a two-stream network for image retrieval in which the auxiliary stream, DeepFixNet, is used specifically for predicting attention maps, which are then fused with the feature maps produced by the main network. In a nutshell, attention mechanisms offer deep networks the capacity to attend on the most important regions within a given image. For image retrieval, attention mechanisms can be combined with supervised metric learning [70].

4.2.4 Hashing Embedding

Real-valued features extracted by deep networks are typically high-dimensional, and therefore are not well-suited to retrieval efficiency. As a result, there is significant motivation to transform deep features into more compact codes. Due to their computational and storage efficiency, hashing algorithms have been widely used for global [64], [85] and local descriptors [66], [89], [90].

Hash functions can be plugged as a layer into deep networks, so that hash codes and deep networks can be simultaneously trained and optimized, either supervised [85] or unsupervised [67]. During hash function training, the hash codes of originally similar images are embedded as closely as possible, and the hash codes of dissimilar images are as separated as possible. A hash function $h(\cdot)$ for binarizing features of an image x may be formulated as

$$b_k = h(x) = h\big(f(x;\boldsymbol{\theta})\big) \quad k = 1, \ldots, K \tag{9}$$

such that an image can be represented by the hash code $\boldsymbol{b} \in \{+1, -1\}^K$. Because hash codes are non-differentiable their optimization is difficult, so $h(\cdot)$ can be relaxed to be differentiable by using *tanh* or *sigmoid* functions [23].

When binarizing real-valued features, it is crucial to preserve image similarity and to improve hash code quality [23]. These two aspects are at the heart of hashing algorithms to maximize retrieval accuracy.

a. Hash Functions to Preserve Image Similarity

Preserving similarity seeks to minimize the inconsistencies between real-valued features and corresponding hash codes, for which a variety of strategies have been adopted.

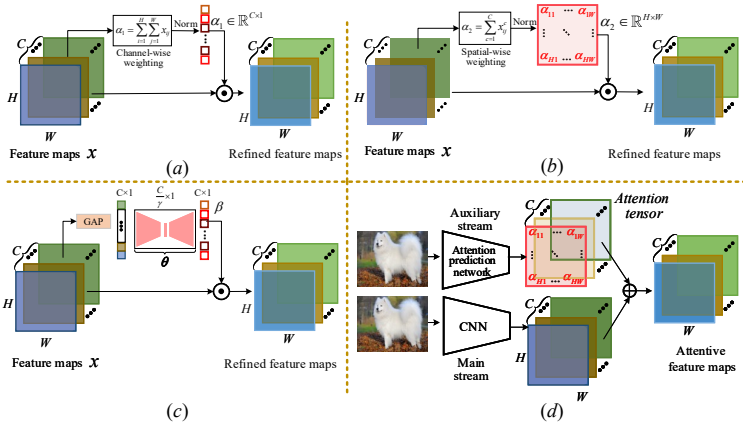


Fig. 6: Attention mechanisms are divided into two categories. (a)-(b) Non-parametric mechanisms: The attention is based on convolutional feature maps x with size $H \times W \times C$. Channel-wise attention in (a) produces a C-dimensional importance vector α_1 [16], [36]; Spatial-wise attention in (b) computes a 2-dimensional attention map α_2 [16], [34], [70]. (c)-(d) Parametric mechanisms: The attention weights are learned by a network with trainable parameters. In (c), β are provided by a sub-network with parameters θ [72], [82], [98], [120]. In (d), the attention maps, as a tensor, are predicted by some auxiliary saliency extraction models from the input image directly [83], [84], [121].

Loss functions can significantly influence similarity preservation, which includes both supervised and unsupervised methods. With class labels available, many loss functions are designed to learn hash codes in a Hamming space. As a straightforward method, one can optimize either the difference between matrices computed from the binary codes and their supervision labels [64], [86] or the difference between the hash codes and real-valued deep features [66], [67]. Song *et al*. [66] propose to learn hash codes for regional features in which each local feature is converted to a set of binary codes by multiplying a hash function and the raw RoI features, then the differences between RoI features and hash codes are characterized by an L_2 loss. Do *et al*. [122] regularize hash codes with a reconstruction loss, which ensure that codes can be reconstructed to their inputs so that similar/dissimilar inputs are mapped to similar/dissimilar hash codes.

b. Improving Hash Function Quality

A good hash function seeks to have binary codes uniformly distributed; that is, maximally filling and using the hash code space, normally on the basis of bit uncorrelation and bit balance [23], [67]. Bit uncorrelation implies that different bits are as independent as possible, so that a given set of bits can aggregate more information within a given code length [67]. Bit balance means that each bit should have a 50% chance of being +1 or -1, thereby maximizing code variance and information [23]. Morère *et al*. [64] use the uniform distribution $U(0,1)$ to build a regularization term to make the hash codes distribute evenly where the codes are learned by a Restricted Boltzmann Machine layer. Likewise, Lin *et al*. [67] optimize the mean of learned hash codes to be close to 0.5 to prevent any bit bias towards zero or one.

5 Retrieval via Learning DCNN Representations

Fine-tuning methods have been studied extensively to learn better retrieval feature. DCNNs pre-trained on source

datasets for image classification are quite robust to inter-class variability, and subsequently pair-wise supervisory information is incorporated into ranking loss to fine-tune networks by regularizing on retrieval representations. A standard dataset with clear and well-defined ground-truth labels is indispensable for fine-tuning deep models to perform accurate instance-level retrieval, otherwise it is necessary to develop unsupervised fine-tuned methods. After network fine-tuning, features can be organized as global or local to perform retrieval.

In Section 4, we presented feature processing strategies for which off-the-shelf DCNNs serve as feature extractors. However, in most cases, deep features may not be sufficient for accurate retrieval, even with the discussed strategies. In order for models to be more effective for retrieval, a common practice is network fine-tuning, *i.e.*, updating the pre-stored parameters [32]. Fine-tuning does not contradict or render irrelevant the feature methods of Section 4; indeed, those strategies are complementary and can be incorporated as part of network fine-tuning.

5.1 Supervised Fine-tuning

The way to realize supervised fine-tuning can be determined by the given class labels or pairwise supervisory information.

5.1.1 Fine-tuning via Classification Loss

When class labels of a new dataset are available, it is preferable to begin with a previously-trained DCNN, trained on a separate dataset, with the backbone DCNN typically chosen from one of AlexNet, VGG, GoogLeNet, or ResNet. The DCNN can then be fine-tuned, as shown in Figure 7 (a), by optimizing its parameters on the basis of a cross entropy loss

$$L_{CE}(\hat{p}_i, y_i) = -\sum_{i}^{c}\big(y_i \times log(\hat{p}_i)\big) \tag{10}$$

Here y_i and $\hat{p}_i$ are the ground-truth labels and the predicted logits, respectively, and c is the total number of categories. The milestone work in such fine-tuning is [40], in which AlexNet is re-trained on the Landmarks dataset. The fine-tuned network produces superior features on landmark-related datasets like Holidays [123], Oxford-5k, and Oxford-105k [124]. The newly-updated layers are used as global or local feature detectors for image retrieval.

A classification-based fine-tuning method improves the *model-level* adaptability for new datasets, which, to some extent, has mitigated the issue of model transfer for image retrieval. Recently, Boudiaf *et al*. [125] claim that cross entropy loss can minimize intra-class distances while maximizing inter-class distances. Cross entropy loss is, in essence, maximizing a common mutual information between the retrieval features and the ground-truth labels. Therefore, it can be regarded as an upper bound on a new pairwise loss, which has a structure similar to various pairwise ranking losses, of which representatives are introduced below.

5.1.2 Fine-tuning via Pairwise Ranking Loss

With affinity information indicating similar and dissimilar pairs, fine-tuning methods based on pairwise ranking loss learn an optimal metric which minimizes or maximizes the distance of pairs to maintain their similarity. Network fine-tuning via ranking loss involves two types of information [18]:

1) A pair-wise constraint, corresponding to a Siamese network as in Figure 7 (c), in which input images are paired with either a positive or negative sample;
2) A triplet constraint, associated with triplet networks as in Figure 7 (e), in which anchor images are paired with both similar and dissimilar samples [18].

These pairwise ranking loss based methods are categorized into globally supervised approaches (Figure 7 (c,d)) and locally supervised approaches (Figure 7 (g,h)), where the former ones learn a metric on global features by satisfying all constraints, whereas the latter ones focus on local areas by only satisfying the given local constraints (*e.g.* region proposals).

To be specific, consider a triplet set $X = \{(x_a, x_p, x_n)\}$ in a mini-batch, where (x_a, x_p) indicates a similar pair and (x_a, x_n) a dissimilar pair. Features $f(x;\boldsymbol{\theta})$ of one image are extracted by a network $f(\cdot)$ with parameters $\boldsymbol{\theta}$, for which we can represent the affinity information for each similar or dissimilar pair as

$$D_{ij} = D(x_i, x_j) = ||f(x_i;\boldsymbol{\theta}) - f(x_j;\boldsymbol{\theta})||_2^2 \tag{11}$$

a. Refining with Transformation Matrix.

Learning the similarity among input samples can be implemented by optimizing the weights of a linear transformation matrix [41]. It transforms the concatenated feature pairs into a common latent space using a transformation matrix $\boldsymbol{W} \in \mathbb{R}^{2d\times 1}$, where d is the feature dimension. The similarity score of these pairs are predicted via a sub-network $S_W(x_i, x_j) = f_W(f(x_i;\boldsymbol{\theta}) \cup f(x_j;\boldsymbol{\theta}); \boldsymbol{W})$ [41]. In other words, the sub-network f_W predicts how similar the feature pairs are. Given the affinity information of feature pairs $S_{ij} = S(x_i, x_j) \in \{0, 1\}$, the binary labels 0 and 1 indicate the similar (positive) or dissimilar (negative) pairs, respectively. The training of function f_W can be achieved by using a regression loss:

$$\begin{aligned} L_W(x_i, x_j) = &|S_W(x_i, x_j) - S_{ij}\big(\text{sim}(x_i, x_j) + m\big) - \\ &(1 - S_{ij})\big(\text{sim}(x_i, x_j) - m\big)| \end{aligned} \tag{12}$$

where $\text{sim}(x_i, x_j)$ can be the cosine function for guiding the training of $\boldsymbol{W}$ and m is a margin. By optimizing the regression loss and updating $\boldsymbol{W}$, deep networks maximize the similarity of similar pairs and minimize that of dissimilar pairs. It is worth noting that the pre-stored parameters in the deep models are frozen when optimizing $\boldsymbol{W}$. The pipeline of this approach is depicted in Figure 7 (b).

b. Fine-tuning with Siamese Networks.

Siamese networks represent important options in implementing metric learning for finetuning, as in Figure 7 (c) and Figure 8 (a). It is a structure composed of two branches that share the same weights across layers. Siamese networks are trained on paired data, consisting of an image pair (x_i, x_j) such that $S(x_i, x_j) \in \{0, 1\}$. A Siamese loss is formulated as

$$\begin{aligned} L_{Siam}(x_i, x_j) = &\frac{1}{2} S(x_i, x_j) D(x_i, x_j) + \\ &\frac{1}{2}\big(1 - S(x_i, x_j)\big) \max\big(0,\ m - D(x_i, x_j)\big) \end{aligned} \tag{13}$$

Siamese loss has recently been reaffirmed as a very effective metric in category-level image retrieval, outperforming many more sophisticated losses if implemented carefully [126]. Enabled by the standard Siamese network, this objective function is used to learn the similarity between semantically relevant samples under different scenarios [47], [127]. For example, Radenović *et al*. [47] employ a Siamese network on matching and non-matching global feature pairs which are aggregated by GeM-based pooling. The deep network fine tuned by the Siamese loss generalizes better and converges

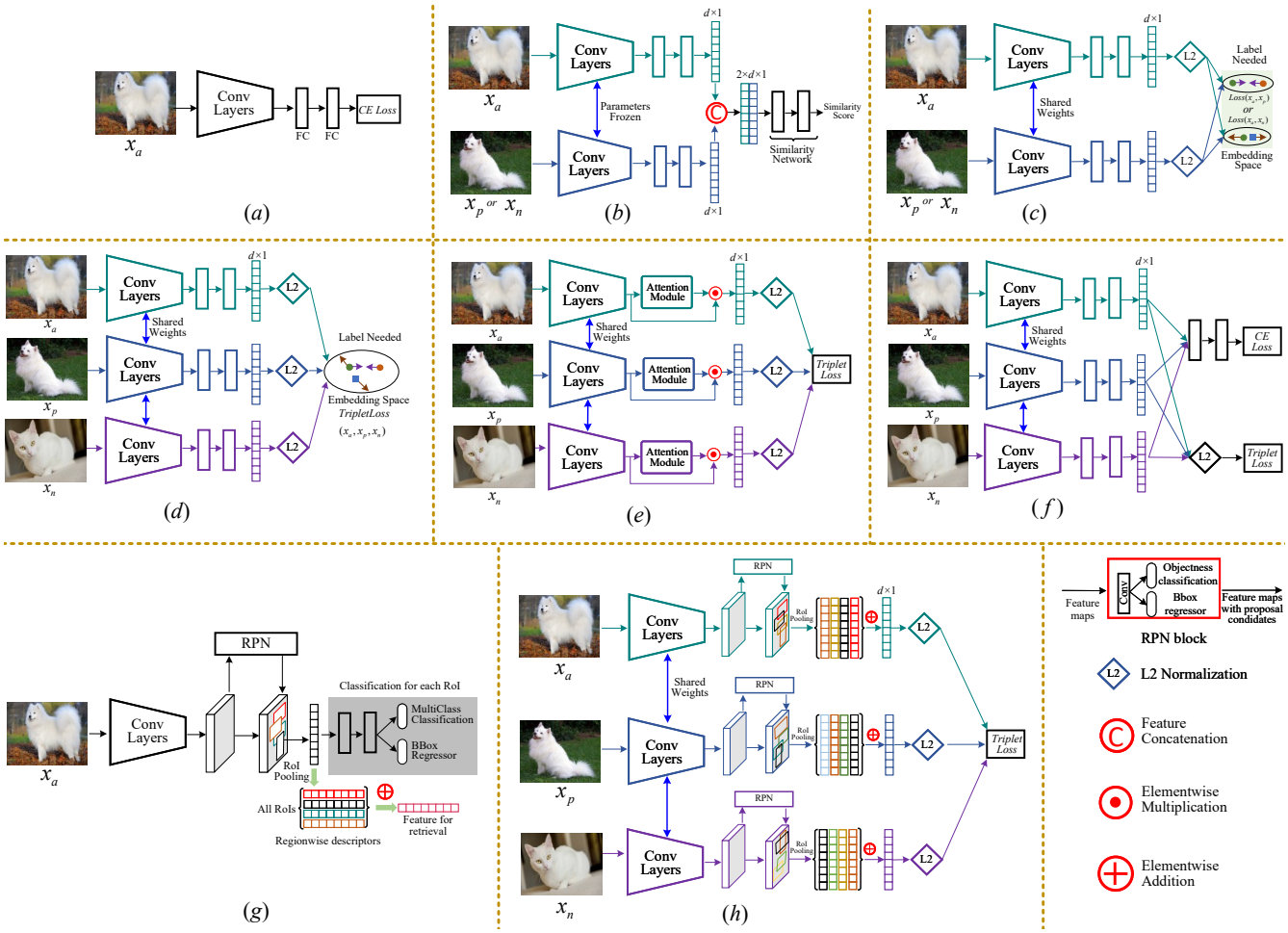


Fig. 7: Schemes of supervised fine-tuning. Anchor, positive, and negative images are indicated by x_a, x_p, x_n, respectively. (a) classification loss; (b) similarity learning by using a transformation matrix; (c) Siamese loss; (d) triplet loss; (e) an attention block into DCNNs to highlight regions; (f) combining classification loss and pairwise ranking loss; (g) region proposal networks (RPNs) to locate the RoI and highlight specific regions or instances; (h) inserting the RPNs of (g) into DCNNs, such that the RPNs extract regions or instances at the convolutional layer.

at higher retrieval performance. Ong *et al*. [42] leverage the Siamese network to learn image features which are then fed into the Fisher Vector model for further encoding. Siamese networks can also be applied to hashing learning in which the Euclidean distance $D(\cdot)$ in Eq. 13 is computed for binary codes [128].

An implicit drawback of the Siamese loss is that it may penalize similar image pairs even if the margin between these pairs is small or zero [128], if the constraint is too strong and unbalanced. At the same time, it is hard to map the features of similar pairs to the same point when images contain complex contents or scenes. To tackle this limitation, Cao *et al*. [129] adopt a double-margin Siamese loss [128] to relax the penalty for similar pairs by setting a margin m_1 instead of zero, in which case the original single-margin Siamese loss is re-formulated as

$$L_{D\_Siam}(x_i, x_j) = \frac{1}{2} S(x_i, x_j) \max\left(0, D(x_i, x_j) - m_1\right) + \frac{1}{2}\left(1 - S(x_i, x_j)\right) \max\left(0, m_2 - D(x_i, x_j)\right) \quad (14)$$

where m_1>0 and m_2>0 are the margins affecting the similar and dissimilar pairs, respectively, as in Figure 8 (b), meaning that the double margin Siamese loss only applies a contrastive force when the distance of a similar pair is larger than m_1. The mAP metric of retrieval is improved when using the double margin Siamese loss [128].

More recently, transformers have been trained under the regularization of cross entropy [52] and Siamese loss [80] for instance-level retrieval and achieved competitive performance, positioning it as an alternative to convolutional architectures. As observed by [80], the transformer-based architecture is less impacted than convolutional networks by feature collapse since each input feature is projected to different sub-spaces before the multi-headed attention. Moreover, the transformer backbone operates as a learned aggregation operator, thereby avoiding the design of sophisticated feature aggregation methods.

c. Fine-tuning with Triplet Networks.

Triplet networks optimize similar and dissimilar pairs simultaneously. As shown in Figure 7 (d) and Figure 8 (c), the plain triplet networks adopt a ranking loss for training:

$$L_{Triplet}(x_a, x_p, x_n) = \max\left(0, m + D(x_a, x_p) - D(x_a, x_n)\right) \quad (15)$$

which indicates that the distance of an anchor-negative pair $D(x_a, x_n)$ should be larger than that of an anchor-positive pair $D(x_a, x_p)$ by a certain margin m.

To focus on specific regions or objects, local supervised metric learning has been explored [48], [66], [87], [96]. In these methods, some regions or objects are extracted using region proposal networks (RPNs) [28] which subsequently can be plugged into deep networks and trained in an end-to-end manner, as shown

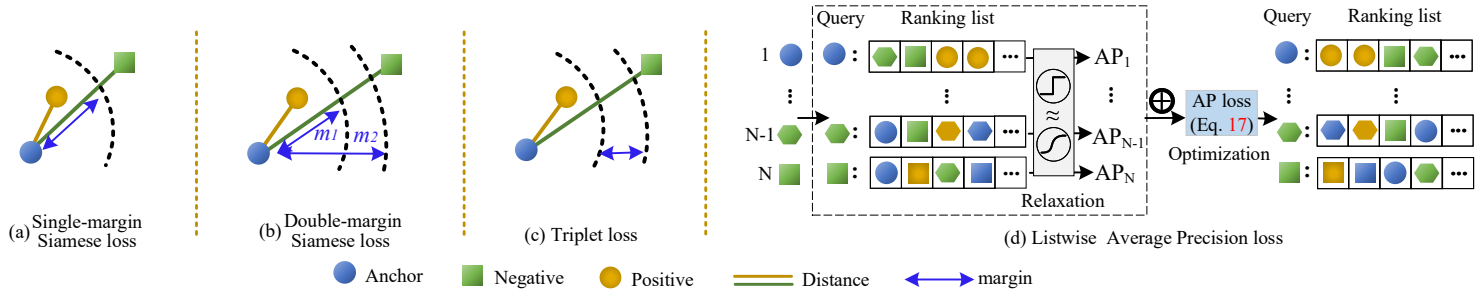


Fig. 8: Illustrations of different losses for network fine-tuning. We use shape to indicate different instances. The same shape with different colors denotes images that include the same instance. (a)-(c) have been introduced in the text. (d) Listwise AP loss considers a mini-batch of N features simultaneously and directly optimizes the Average-Precision computed from these features.

in Figure 7 (g), in which Faster R-CNN [28] is fine-tuned for instance search [96]. RPNs yield the regressed bounding box coordinates of objects and are trained by the multi-class classification loss. The final networks extract better regional features by RoI pooling and perform spatial ranking for retrieval.

RPNs [28] enable deep models to learn regional features for particular instances objects [43], [87]. RPNs used in the triplet formulation are shown in Figure 7 (h). For training, besides the triplet loss, regression loss (RPNs loss) is used to minimize the regressed bounding box relative to ground-truth. In some cases, jointly training an RPN loss and triplet loss leads to unstable results, a problem addressed in [43] by first training a CNN to produce R-MAC using a rigid grid, after which the parameters in convolutional layers are fixed and RPNs are trained to replace the rigid grid.

Attention mechanisms can also be combined with metric learning for fine-tuning [66], as in Figure 7 (e), where the attention module is typically end-to-end trainable and takes as input the convolutional feature maps. Song *et al*. [66] introduce a convolutional attention layer to explore spatial-semantic information, highlighting regions in images to significantly improve the discrimination for image retrieval.

Recent studies [100], [130] have jointly optimized the triplet loss and classification loss to further improve network capacity, as shown in Figure 7 (f). The overall joint function is

$$L_{Joint} = \alpha \cdot L_{Triplet}(x_{i,a}, x_{i,p}, x_{i,n}) + \beta \cdot L_{CE}(\hat{p}_i, y_i) \quad (16)$$

where the cross entropy loss (CE loss) L_{CE} is defined in Eq. (10) and the triplet loss $L_{Triplet}$ in Eq. (15). α and β are hyper-parameters tuning the tradeoff between the two loss functions.

5.1.3 Discussion

In some cases, pairwise ranking loss cannot effectively learn the variations between samples and still suffers from a weaker generalization capability if the training set is not ordered correctly. Therefore, pairwise ranking loss requires careful sample mining and weighting strategies to obtain the most informative training pairs, especially when considering mini-batches. The hard negative mining strategy is commonly used [44], [47], [118], however further sophisticated mining strategies have recently been developed. Mishchuk *et al*. [110] calculate a pair-wise distance matrix on all mini-batch samples to select two closest negative and one anchor-positive pair to form a triplet to fine-tune. Instead of traversing all possible two-tuple or three-tuple combinations, it is possible to consider all positive samples in one cluster and negative samples another. Liu *et al*. [4] introduce the group-group loss to decrease the intra-group distance and increase the inter-group distance. Considering all samples may be beneficial for stabilizing optimization and promoting generalization performance due to a larger data diversity, however the extra computational cost remains an issue to be addressed.

Substantial research has been devoted to pair-wise ranking loss, while cross entropy loss, mainly used for classification, has been largely overlooked. Recently, Boudiaf *et al*. [125] claim that cross entropy loss can match and even surpass the pair-wise ranking loss when carefully tuned on fine-grained category-level retrieval tasks. In fact, the greatest improvements have come from enhanced training schemes (*e.g*., data augmentation, learning rate polices, batch normalization freeze) rather than intrinsic properties of pairwise ranking loss. Further, although several sophisticated ranking losses have been explored and validated for category-level retrieval, Musgrave *et al*. [126] revisited these losses and found that most of them perform on par to vanilla Siamese loss and triplet loss, so there is merit to consider these losses also for instance-level image retrieval tasks.

Both cross entropy loss and pair-wise ranking loss regularize on the embedded features and the corresponding labels so as to maximize their mutual information [125]. Their effectiveness is not guaranteed to give retrieval results that also optimize mAP [131]. To tackle this limitation one can directly optimize the average precision (AP) metric using the listwise AP loss,

$$L_{mAP} = 1 - \frac{1}{N}\sum_{i=1}^{N} \mathrm{AP}(x_i^\top X_N, Y_i) \quad (17)$$

which optimizes the global ranking of thousands of images simultaneously, instead of only a few images at a time. Here Y_i is the binary label to evaluate the relevance between batch images. $X_N = \{x_1, x_2, ...x_j, ..., x_N\}$ denotes the features of all images, where each x_i is used as a potential query to rank the remaining batch images. Each similarity score $x_i^\top x_j$ can be measured by a cosine function.

It is demonstrated that training with AP-based loss improves retrieval performance [131], [132]. However average precision, as a metric, is normally non-differentiable. To directly optimize the AP loss during back-propagation, the key is that the indicator function for AP computing needs to be relaxed using methods such as triangular kernel-based soft assignment [131] or sigmoid function [132], as shown in Figure 8 (d).

5.2 Unsupervised Fine-tuning

Supervised network fine-tuning becomes infeasible when there is insufficient supervisory information, normally because of cost or unavailability. Therefore unsupervised fine-tuning methods for image retrieval are quite necessary, but less studied [133]. For unsupervised fine-tuning, two directions are to mine relevance among features via manifold learning, and via clustering techniques, each discussed below.

5.2.1 Mining Samples with Manifold Learning

Manifold learning focuses on capturing intrinsic correlations on a manifold structure to mine or deduce revelance, as illustrated in Figure 9. Initial similarities between the extracted global features [134] or local features [15], [135] are used to construct an affinity matrix, which is then re-evaluated and updated using manifold learning [136]. According to the manifold similarity in the updated affinity matrix, positive and hard negative samples are selected for metric learning using pairwise ranking loss based functions such as pair loss [48], [135] or triplet loss [133], [137]. Note that this is different from the aforementioned methods for pairwise ranking loss based fine-tuning methods, where the hard positive and negative samples are explicitly selected from an ordered dataset according to the given affinity information.

It is important to capture the geometry of the manifold of deep features, generally involving two steps [136], known as diffusion. First, the affinity matrix (Figure 9) is interpreted as a weighted kNN graph, where each vector is represented by a node, and edges are defined by the pairwise affinities of two connected nodes. Then, the pairwise affinities are re-evaluated in the context of all other elements by diffusing the similarity values through the graph [49], [133], [135], [137], with recent strategies proposed such as regularized diffusion (RDP) [138] and regional diffusion [135]. For more details on diffusion methods refer to survey [136].

Most algorithms follow the two steps of [136]; the differences among methods lie primarily in three aspects:

1) **Similarity initialization**, which affects the subsequent kNN graph construction in an affinity matrix. Usually, an inner product [49] or Euclidean distance [46] is directly computed for the affinities. A Gaussian kernel function can be used [136], [137], or consider regional similarity from image patches [135].
2) **Transition matrix definition**, a row-stochastic matrix [136], determines the probabilities of transiting from one node to another in the graph. These probabilities are proportional to the affinities between nodes, which can be measured by Geodesic distance (*e.g.* the summation of weights of relevant edges).
3) **Iteration scheme**, to re-valuate and update the values in the affinity matrix by the manifold similarity until some convergence is achieved. Most algorithms are iteration-based [133], [136], as illustrated in Figure 9.

Diffusion process algorithms are indispensable for unsupervised fine-tuning. Better image similarity is guaranteed when it is improved based on initialization (*e.g.* regional similarity [135] or higher order information [46]). Diffusion is normally iterative and is computationally demanding [137], a limitation which cannot meet the efficiency requirements of image retrieval. To reduce the computational complexity, Bai *et al*. [138] propose a regularized diffusion process, facilitated by an efficient iteration-based solver. Zhao *et al*. [137] regard the diffusion process as a non-linear kernel mapping function, which is then modelled by a deep neural network. Other studies replace the diffusion process on a kNN graph with a diffusion network [48], which is derived from graph convolution networks [139], an end-to-end trainable framework which allows efficient computation during training and testing.

Once the manifold space is learned, samples are mined by computing geodesic distances based on the Floyd-Warshall algorithm or by comparing the set difference [133]. The selected samples are fed into deep networks to perform fine-tuning.

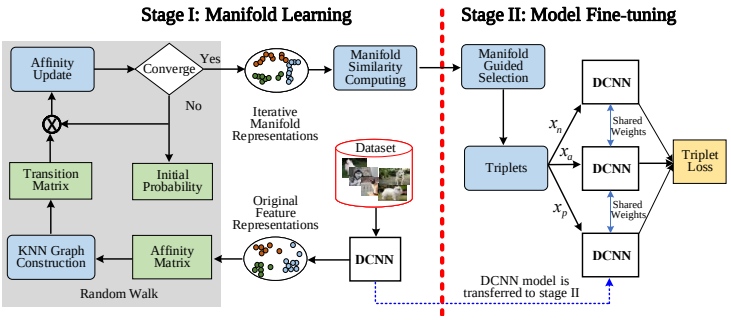


Fig. 9: Paradigm of manifold learning for unsupervised metric learning, based on triplet loss.

5.2.2 Mining Samples by Clustering

Clustering is used to explore proximity information that has been studied in instance-level image retrieval [45], [140], [141], [142], [143]. The rationale behind these methods is that samples in a cluster are likely to satisfy a degree of similarity.

One class of methods for clustering deep features is via k-means. Given k cluster centroids, during each training epoch a deep network alternates between two steps: first, a soft assignment between the feature representations and the cluster centroids; second, the cluster centroids are refined and, at the same time, the deep network is updated by learning from current high confidence assignments using a certain regularization. These two steps are repeated until a convergence criterion is met, at which point the cluster assignments are used as pseudo-labels [141], [142]. Alternatively, the pseudo-labels can be calculated from the samples in a cluster, *e.g.*, the mean values. For example, Tzelepi *et al*. [140] compute k nearest feature representations with respect to a query feature and then compute their mean vectors, which is used as a target for the query feature. In this case, fine-tuning is performed by minimizing the squared distance between each query feature and the mean of its k nearest features. Liu *et al*. [86] propose a self-taught hashing algorithm using a kNN graph construction to generate pseudo labels that are used to analyze and guide network training. Shen *et al*. [144] and Radenović *et al*. [45], [47] use Structure-from-Motion (SfM) for each image cluster to explore sample reconstructions to select images for triplet loss. Clustering methods depend on the Euclidean distance, making it difficult to reveal the intrinsic relationship between objects.

There are further techniques for instance retrieval, such as by using AutoEncoder [122], [145], generative adversarial networks (GANs) [146], convolutional kernel networks [112], [147], and graph convolutional networks [48]. For these methods, they focus on devising novel unsupervised frameworks to realize unsupervised learning, instead of iterative similarity diffusion or cluster refinement on feature space. For example, instead of performing iterative traversal on a set of nearest neighbors defined by kNN graph, Liu *et al*. [48] employ graph convolutional networks [139] to directly encode the neighbor information into image descriptors and then train the deep models to learn a new feature space. This method is demonstrated to significantly improve retrieval accuracy while maintaining efficiency. GANs are also explored, for the first time, for instance-level retrieval in an unsupervised fashion [146]. The generator retrieves images that contain similar instances as a given image, while the discriminator judges whether the retrieved images have the specified instance which appeared in the query image. During training, the discriminator and the

generator play a min-max game via an adversarial reward which is computed based on the cosine distance between the query image and the images retrieved by the generator.

6 State of the Art Performance

6.1 Datasets

To demonstrate the effectiveness of methods, we choose the following commonly-used datasets for performance comparison:

UKBench (UKB) [148] consists of 10,200 images of objects. This dataset has 2,550 groups of images, each group having four images of the same object from different viewpoints or illumination conditions. All images can be used as a query.

Holidays [123] consists of 1,491 images collected from personal holiday albums. Most images are scene-related. The dataset comprises 500 groups of similar images with a single query image for each group.

Oxford-5k [124] consists of 5,062 images for 11 Oxford buildings. Each building is associated with five hand-drawn bounding box queries. An additional disjoint set of 100,000 distractor images is added to obtain Oxford-105k.

Paris-6k [149] includes 6,412 images and is categorized into 12 groups by architecture. Images are annotated with the same types of labels as used in Oxford-5k. Recently, additional queries and distractor images have been added into Oxford-5k and Paris-6k, producing the Revisited Oxford ($\mathcal{R}$Oxford) and Revisited Paris ($\mathcal{R}$Paris) datasets [150]. We also undertake partial comparisons under the *hard* evaluation protocol on these revisited datasets.

INSTRE [151] consists of 28,543 images from 250 different object classes, including three disjoint subsets[2]: INSTRE-S1, INSTRE-S2, INSTRE-M. The evaluation of INSTRE follows the protocol in [135].

Google Landmarks Dataset (GLD) [5], [65] consists of GLD-v1 and GLD-v2. The number of images in GLD-v1 shrinks over time as these images might be deleted. GLD-v2 has the advantage of stability where all images have permissive licenses [72]. GLD-v2 consists of over 5M images and over 200k distinct instance labels, making it the largest instance recognition dataset to date. It is divided into three subsets: (i) 118k query images with ground-truth annotations, (ii) 4.1M training images of 203k landmarks with labels, and (iii) 762k index images of 101k landmarks. The training set is further cleaned by removing clutter, consisting of a subset "GLD-v2-clean" containing 1.6M images of 81k landmarks.

6.2 Evaluation Metrics

Average precision (AP) refers to the coverage area under the precision-recall (PR) curve. A larger AP implies a higher PR curve and better retrieval accuracy. AP can be calculated as

$$AP = \frac{\sum_{k=1}^{N} P(k) \cdot rel(k)}{R} \tag{18}$$

where R denotes the number of relevant results for the query image from the total number N of images. $P(k)$ is the precision of the top k retrieved images, and $rel(k)$ is an indicator function equal to 1 if the item within rank k is a relevant image and 0 otherwise. Mean average precision (mAP) is adopted for the evaluation over all query images,

$$\frac{1}{Q}\sum_{q=1}^{Q} AP(q) \tag{19}$$

2. https://github.com/imatge-upc/salbow

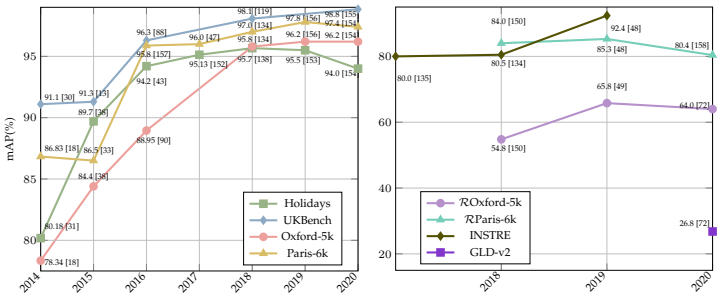


Fig. 10: Performance improved from 2014 to 2020.

where Q is the number of query images.

The **N-S score** is a metric used for UKBench [148]; the N-S score is the average for the top-4 precision over the dataset.

6.3 Performance Comparison and Analysis

Overview. Figure 10 summarizes the performance over 6 datasets from 2014 to 2020. Early on, the powerful feature extraction of DCNNs led to rapid improvements. Subsequent key ideas have been to extract instance features at the region level to reduce image clutter [31], and to improve feature discriminativity by using methods including feature fusion [90], [152], [155], feature aggregation [33], [69], and feature embedding [90]. Fine-tuning is an important strategy to improve performance by tuning deep networks specific for learning instance features [138], [154]. For instance, the accuracy increases steadily from 78.34% [18] to 96.2% [156] on the Oxford-5k dataset when manifold learning is used to fine-tune deep networks. The mAP on $\mathcal{R}$Paris-6k and $\mathcal{R}$Oxford-5k is smaller than Paris-6k and Oxford-5k, leaving room for improvement.

We report results using off-the-shelf models (Table 2) and fine-tuning networks (Table 3). In Table 2, single-pass and multiple-pass are analyzed, while supervised and unsupervised fine-tuning are compared in Table 3.

Evaluation for single feedforward pass. The common practice using this scheme is to enhance feature discrimination. In Table 2, we observe that fully-connected layers used as feature extractors may have a lower accuracy (*e.g.*, 74.7% on Holidays in [40]), compared to using convolutional layers because the fully-connected layers lack structural information. Layer-level feature fusion improves retrieval accuracy; for example, Yu *et al*. [102] combined three layers (mAP of 91.4% on Holidays), outperforming the performance of non-fusion method [13] (mAP of 80.2%). Moreover, convolutional features embedded by a BoW model are competitive on Oxford-5k and Paris-6k (73.9% and 82.0%, respectively), while its codebook size is 25k, which may affect retrieval efficiency. For single-pass schemes, the methods shown in Figure 5 improve the discrimination of convolutional feature maps and perform differently in Table 2, 66.9% of R-MAC [149] and 58.9% of SPoC [13] on Oxford-5k, differences which we see as critical factors for further analysis.

Evaluation for multiple feedforward pass. Results for the methods of Figure 4 are reported. Among them, extracting image patches densely using Overfeat [160] has the highest performance on the 4 datasets [30], and rigid grid [115] is competitive (mAP of 87.2% on Paris-6k). These two methods consider more patches, even background information, when used for feature extraction. Instead of generating patches densely, region proposals and spatial pyramid modeling introduce a degree of purpose and efficiency in processing image objects. Spatial information is better maintained using multiple-pass schemes than with single-pass. For example, a

TABLE 2: Performance evaluation of off-the-shelf DCNN models. "•" indicates that the models or layers are combined to learn features; "PCA_w" indicates PCA with whitening on the extracted features to improve robustness; "MP" means Max Pooling; "SP" means Sum Pooling. The CNN-M network with "$*$" has an architecture similar to that of AlexNet. "-" means that the results were not reported.

Type	Method	Backbone DCNN	Output Layer	Embedding Aggregation	Feature Dimension	Holidays	UKB	Oxford5k (+100k)	Paris6k (+100k)	Brief Conclusions and Highlights
Single-pass	Neural codes [40]	AlexNet	FC6	PCA	128	74.7	3.42 (N-S)	43.3 (38.6)	–	Compressed neural codes of different layers are explored. AlexNet is also fine-tuned for retrieval.
	R-MAC [33]	VGG16	Conv5	R-MAC + PCA_w	512	–	–	66.9 (61.6)	– (75.7)	Sliding windows with different scales on convolutional feature maps to encode multiple image regions.
	CroW [16]	VGG16	Conv5	CroW + PCA_w	256	85.1	–	68.4 (63.7)	76.5 (69.1)	The spatialwise and channelwise weighting mechanisms are utilized to highlight crucial convolutional features.
	BLCF [68]	VGG16	Conv5	BoW + PCA_w	25k	–	–	73.9 (59.3)	82.0 (64.8)	Both global features and local features are explored, demonstrating that local features have higher accuracy.
	SPoC [13]	VGG16	Conv5	SPoC + PCA_w	256	80.2	3.65 (N-S)	58.9 (57.8)	–	Exploring Gassian weighting scheme *i.e.*, the centering prior, to improve the discrimination of features.
	Multi-layer CNN [102]	VGG16	FC6 • Conv4∼5	SP	4096	91.4	3.68 (N-S)	61.5 (–)	–	Layer-level feature fusion and the complementary properties of different layers are explored.
Multiple-pass	Deepindex [54]	AlexNet • VGG19	FC6-7 • FC17-18	BoW + PCA	512	81.7	3.32 (N-S)	–	75.4 (–)	Exploring layer-level and model-level fusion methods. Image patches are extracted using spatial pyramid modeling.
	MOF [37]	CNN-M* [159]	FC7 • Conv	SP or MP + BoW	20k	76.8	3.00 (N-S)	–	–	Exploring layer-level fusion scheme. Image patches are extracted using spatial pyramid modeling.
	Multi-scale CNN [69]	VGG16	Conv5	SP or MP + PCA_w	32k	89.6	95.1 (mAP)	84.3 (–)	87.9 (–)	Image patches are extracted in a dense manner. Geometric invariance is considered when aggregating patch features.
	CNNaug-ss [30]	Overfeat [160]	FC	PCA_w	15k	84.3	91.1 (mAP)	68.0 (–)	79.5 (–)	Image patches are extracted densely. Image regions at different locations with different sizes are included.
	MOP-CNN [31]	AlexNet	FC7	VLAD + PCA_w	2048	80.2	–	–	–	Image patches are extracted densely. Multi-scale patch features are further embedded into VLAD descriptors.
	CCS [55]	GoogLeNet	Conv	VLAD + PCA_w	128	84.1	3.81 (N-S)	64.8 (–)	76.8 (–)	Object proposals are extracted by RPNs. Object level and point level feature concatenation schemes are explored.
	OLDFP [63]	AlexNet	FC6	MP + PCA_w	512	88.5	3.81 (N-S)	60.7 (–)	66.2 (–)	Exploring the impact of proposal number. Patches are extracted by RPNs and the features are encoded in an orderless way.
	LDD [115]	VGG19	Conv5	BoW + PCA_w	500k	84.6	–	83.3 (–)	87.2 (–)	Image patches are obtained using a uniform square mesh. Patch features are encoded into BoW descriptors.

TABLE 3: Performance evaluation of methods in which DCNN models are fine-tuned, in a supervised or an unsupervised manner. "CE Loss" means the models are fine-tuned using the classification-based loss function in the form of Eq. 10. "Siamese Loss" is in the form of Eq. 13. "Regression Loss" is in the form of Eq. 12. "Triplet Loss" is in the form of Eq. 15.

Type	Method	Backbone DCNN	Output Layer	Embedding Aggregation	Feature Dimension	Loss Function	Holidays	UKB	Oxford5k (+100k)	Paris6k (+100k)	Brief Conclusions and Highlights
Supervised Fine-tuning	DELF [5]	ResNet-101	Conv4 Block	Attention + PCA_w	CE Loss	2048	–	–	83.8 (82.6)	85.0 (81.7)	Exploring the FCN to extract region-level features and construct feature pyramids of different sizes.
	Neural codes [40]	AlexNet	FC6	PCA	CE Loss	128	78.9	3.29 (N-S)	55.7 (52.3)	–	The first work which fine-tunes deep networks for image retrieval. Compressed neural codes and different layers are explored.
	Nonmetric [41]	VGG16	Conv5	PCA_w	Regression Loss	512	–	–	88.2 (82.1)	88.2 (82.9)	Visual similarity learning of similar and dissimilar pairs is performed by a neural network, optimized using regression loss.
	Faster R-CNN [96]	VGG16	Conv5	MP / SP	Regression Loss	512	–	–	75.1 (–)	80.7 (–)	RPN is fine-tuned, based on bounding box coordinates and class scores for specific region query which is region-targeted.
	SIAM-FV [42]	VGG16	Conv5	FV + PCA_w	Siamese Loss	512	–	–	81.5 (76.6)	82.4 (–)	Fisher Vector is integrated on top of VGG and is trained with VGG simultaneously.
	SIFT-CNN [127]	VGG16	Conv5	SP	Siamese Loss	512	88.4	3.91 (N-S)	–	–	SIFT features are used as supervisory information for mining positive and negative samples.
	Quartet-Net [129]	VGG16	FC6	PCA	Siamese Loss	128	71.2	87.5 (mAP)	48.5 (–)	48.8 (–)	Quartet-net learning is explored to improve feature discrimination where double-margin contrastive loss is used.
	NetVLAD [44]	VGG16	VLAD Layer	PCA_w	Triplet Loss	256	79.9	–	62.5 (–)	72.0 (–)	VLAD is integrated at the last convolutional layer of VGG16 network as a plugged layer.
	Deep Retrieval [87]	ResNet-101	Conv5 Block	MP + PCA_w	Triplet Loss	2048	90.3	–	86.1 (82.8)	94.5 (90.6)	Dataset is cleaned automatically. Features are encoded by R-MAC. RPN is used to extract the most relevant regions.
Unsupervised Fine-tuning	MoM [133]	VGG16	Conv5	MP + PCA_w	Siamese Loss	64	87.5	–	78.2 (72.6)	85.1 (78.0)	Exploring manifold learning for mining dis/similar samples. Features are tested globally and regionally.
	GeM [47]	VGG16	Conv5	GeM Pooling	Siamese Loss	512	83.1	–	82.0 (76.9)	79.7 (72.6)	Fine-tuning CNNs on an unordered dataset. Samples are selected from an automated 3D reconstruction system.
	SfM-CNN [45]	VGG16	Conv5	PCA_w	Siamese Loss	512	82.5	–	77.0 (69.2)	83.8 (76.4)	Employing Structure-from-Motion to select positive and negative samples from unordered images.
	IME-CNN [46]	ResNet-101	IME Layer	MP	Regression Loss	2048	–	–	92.0 (87.2)	96.6 (93.3)	Graph-based manifold learning is explored within an IME layer to mine the matching and non-matching pairs in unordered datasets.
	MDP-CNN [137]	ResNet-101	Conv5 Block	SP	Triplet Loss	2048	–	–	85.4 (85.1)	96.3 (94.7)	Exploring global feature structure by modeling the manifold learning to select positive and negative pairs.

shallower network (AlexNet) and region proposal networks in [63] have a UKBench N-Score of 3.81, higher than using deeper networks [13], [40], [102]. Besides feeding image patches into the same network, model-level fusion also exploits complementary spatial information to improve retrieval accuracy. For instance, as reported in [54], which combines AlexNet and VGG, the results on Holidays (81.74% of mAP) and UKBench (3.32 of N-Score) are better than these in [37] (76.75% and 3.00, respectively).

Evaluation for supervised fine-tuning. Compared to off-the-shelf models, fine-tuning deep networks usually improves accuracy, see Table 3. For instance, the result on Oxford-5k [33] by using a pre-trained VGG is improved from 66.9% to 81.5% in [42] when a single-margin Siamese loss is used. Similar trends can be also observed on the Paris-6k dataset. For classification-based fine-tuning, its performance may be improved by using powerful DCNNs and feature enhancement methods such as the attention mechanism in [5], with an mAP increased from 55.7% in [40] to 83.8% in [5] on Oxford-5k. As for pairwise ranking loss fine-tuning, in some cases the loss used for fine-tuning is essential for performance improvement. For example, RPN is re-trained using regression loss on Oxford-5k and Paris-6k (75.1% and 80.7%, respectively) [96]. Its results are lower than the results from [41] (88.2% and 88.2%, respectively) where a transformation matrix is used to learn visual similarity. However, when RPN is trained by using triplet loss such as [87], the effectiveness of retrieval is improved significantly where the results are 86.1% (on Oxford-5k) and 94.5% (on Paris-6k). Feature embedding methods are important for retrieval accuracy; Ong *et al*. [42] embedded *Conv5* feature maps by Fisher Vector and achieved an mAP of 81.5% on Oxford-5k, while embedding feature maps by using VLAD achieves an mAP of 62.5% on this dataset [44], [45].

Evaluation for unsupervised fine-tuning. Compared to supervised fine-tuning, unsupervised fine-tuning methods are relatively less explored. The difficulty for unsupervised fine-tuning is to mine sample relevance without ground-truth labels. In general, unsupervised fine-tuning methods should be expected to have lower performance than supervised.

For instance, supervised fine-tuning using Siamese loss [127] achieves an mAP 88.4% on Holidays, while unsupervised fine-tuning using the same loss function in [45], [47], [133] achieves 82.5%, 83.1%, and 87.5%, respectively. However, unsupervised fine-tuning methods can achieve a similar accuracy, even outperform the supervised fine-tuning, if a suitable feature embedding method is used. For instance, Zhao *et al*. [137] explore global feature structure modeling the manifold learning, producing an mAP of 85.4% (on Oxford-5k) and 96.3% (on Paris-6k), which is similar to supervised results [87] of 86.1% (on Oxford-5k) and 94.5% (on Paris-6k). As another example, the precision of ResNet-101 fine-tuned by cross entropy loss achieves 83.8% on Oxford-5k [5], while the precision is further improved to 92.0% when an IME layer is used to embed features and fine-tuned in an unsupervised way [46]. Note that fine-tuning strategies are related to the type of the target retrieval datasets. As demonstrated in Table 4 and [72], fine-tuning on different datasets may produce a different final retrieval performance.

Network depth. We compare the efficacy of DCNNs by depth, following the fine-tuning protocols[3] in [47]. For fair comparisons, all convolutional features from these backbone DCNNs are aggregated by MAC [69], and fine-tuned by using the same loss function with the same learning rate, thus the adopted methods are the same except for the DCNN depth. We use the default feature dimension (*i.e.* AlexNet (256), VGG (512), GoogLeNet (1024), ResNet-50/101 (2048)). The results are reported in Figure 11 (a). We observe that the deeper networks consistently lead to better accuracy due to extracting more discriminative features.

Feature aggregation methods. The methods of embedding convolutional feature maps were illustrated in Figure 5. We use the off-the-shelf VGG (without updating parameters) on the Oxford and Paris datasets. The results are reported in Figure 11 (b). We observe that the different ways to aggregate the same off-the-shelf DCNN leads to differences in retrieval performance. These reported results provide a reference for feature aggregation when one uses convolutional layers for performing retrieval tasks.

Global feature dimension and regional search. We add fully-connected layers on the top of pooled convolutional features of ResNet-50 to obtain global descriptors with their dimensions varying from 32 to 8192. The results of 5 datasets are shown in Figure 11 (c). It is expected that higher-dimension features usually capture more semantics and are helpful for retrieval. The performance tends to be stable when the dimension is very large.

We compare of regional search in Figure 11 (d). Note that this is different from the re-ranking scenario. Instead, we obtain a different number of regions on one image with a 40% overlap of neighboring regions. Convolutional features of each region are pooled as 2048 features by MAC and then stored independently (*i.e.*, without the feature aggregation operation), which largely increases the memory requirements and the search time as more regions are extracted for each image. As in Figure 11 (c), the holistic image is regarded as the case where only one region is used. As more regions are extracted from the same image, the retrieval mAP declines slightly. A possible reason is that more background or irrelevant regions have also been extracted and used for cross-matching (*i.e.*, many-to-many matching), and negatively affect the performance.

Finetuning datasets and retrieval reranking. We compare performance on $\mathcal{R}$Oxford-5k, $\mathcal{R}$Paris-6k, and GLD-v2, aiming at comparing the role of different fine-tuning training sets and the effectiveness of retrieval reranking. Table 4 lists 8 experimental scenarios using two network backbones, as in [72]. When using the global features only, the cleaned version of the training set is helpful to improve the performance, as observed in Cases 1/5 and Cases 2/6 for ResNet-50/ResNet-101. As an important postprocessing strategy, reranking further boosts the retrieval accuracy after the initial filtering step by using global features.

TABLE 4: Evaluations of training sets and retrieval reranking. Numerical results are cited from [72].

Conditions		Global	Local reranking	Training set		$\mathcal{R}$Oxf	$\mathcal{R}$Par	GLD-v2 testing
				GLD-v1	GLD-v2 -clean			
ResNet -50	Case 1	✓	✗	✓	✗	45.1	63.4	20.4
	Case 2	✓	✗	✗	✓	51.0	71.5	24.1
	Case 3	✓	✓	✓	✗	54.2	64.9	22.3
	Case 4	✓	✓	✗	✓	57.9	71.0	24.3
ResNet -101	Case 5	✓	✗	✓	✗	51.2	64.7	21.7
	Case 6	✓	✗	✗	✓	55.6	72.4	26.0
	Case 7	✓	✓	✓	✗	59.3	65.5	24.3
	Case 8	✓	✓	✗	✓	64.0	72.8	26.8

7 Conclusions and Future Directions

In this survey, we have reviewed deep learning methods for instance retrieval with a new taxonomy, identified milestone approaches, revealed connections among various methods, made performance comparison of representative methods, and discussed merits and limitations. It is clear that deep learning has shown significant progress in IIR, nevertheless there remain many unsolved problems, and the following lists some potential future research directions.

(1) Large Scale Instance Retrieval. While the field of IIR is progressing rapidly, most SOTA methods are tested on very small datasets with limited instance categories. One possible reason that small datasets have been the dominant benchmarks is that it is not easy to gather instance labels at scale. However, this should not be the main obstacle of pushing instance retrieval towards large scale and fine grained which is relevant for practical applications. To achieve such a goal[4], at least three important issues need to be addressed. Firstly, large scale datasets applicable to generic search purpose need to be introduced. Secondly, efficient and effective approaches for generating such large scale datasets in either a supervised or unsupervised fashion are needed. Finally, compact, scalable, yet discriminative feature representations should be developed.

(2) Specialized vs Generic Instance Retrieval. Certainly, there have been increasing interests in specialized instance retrieval, such as landmark retrieval, pedestrian retrieval, vehicle retrieval, *etc*. In addition, specialized instance retrieval is heading towards large scale. By contrast, generic instance retrieval also has great practical value. As a start, combining existing specialized datasets into one or conducting cross-dataset research are also interesting.

(3) Invariant Feature Representations. One of the main challenges in instance retrieval is the large intraclass variability including variations in view point, scale, illumination, weather condition and background clutter *etc*. Despite the significant progress brought by deep learning recently, these challenges remain open as DCNNs have little builtin invariance. However, achieving invariance is critical for instance retrieval. Fortunately,

3. https://github.com/filipradenovic/cnnimageretrieval-pytorch

4. The GLDv2 collected for special applications has been developed towards such a goal, posing novel challenges like long tailed class distribution and noisy labels.

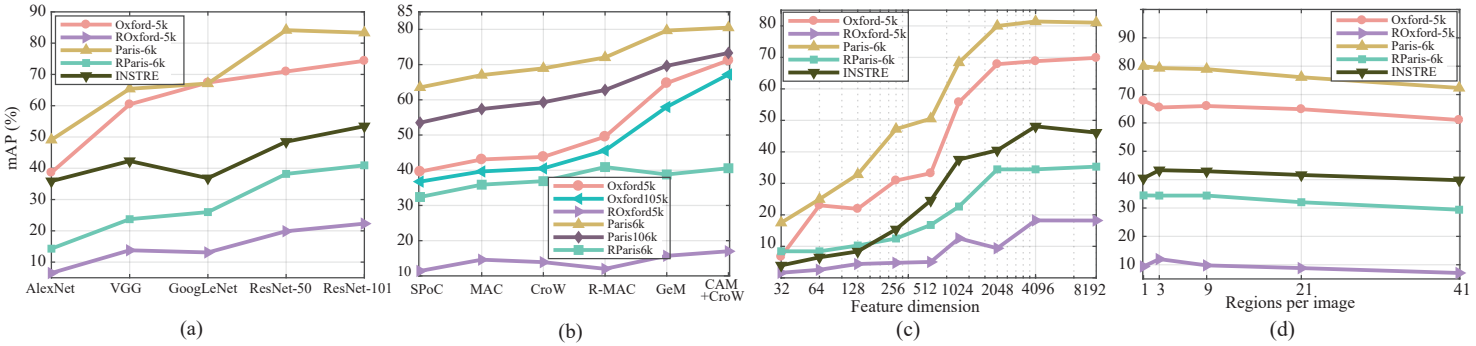


Fig. 11: (a) The effectiveness of different DCNNs on 5 datasets; (b) Comparison of the feature aggregation methods in Figure 5; (c) The impact of global feature dimension by using ResNet-50; (d) Regional search evaluation.

before deep learning, there were numerous experience in local interesting point detection and local invariant feature description. Therefore, it would be beneficial to combine the best of both worlds.

(4) Incremental Image Retrieval. Current image retrieval methods focus on static datasets and are not suited for incremental scenarios [161]. Most methods assume that all instances are available during training. This assumption may be restrictive in real world applications since new instances can constantly emerge. It is time consuming to train deep models on both old and new images repeatedly, while finetuning deep models only on the new images may lead to catastrophic forgetting, thereby degrades their retrieval performance for the already trained instances. Therefore, one direction would be to build an up-to-date retrieval model by incremental learning methods to handle continuous streams of new instances.

(5) Adversarial Robustness. Currently, deep learning has become dominant in image retrieval systems and also been developed in real world applications. However, to build an advanced image retrieval system, increasing the retrieval accuracy alone is not sufficient. It should also be able to resist potential attacks. Recently, deep learning networks have been proven to be fooled by adversarial examples [162], [163], *i.e.*, images added with intentionally designed yet nearly imperceptible perturbations. This raises serious safety concerns. However, in the field of image retrieval, adversarial robustness of image retrieval [162], [164] has received very limited attention and deserves more attention in the future.

Acknowledgment

The authors would like to thank the pioneer researchers in instance retrieval and other related fields. This work was also partially supported by the Academy of Finland under grant 331883 and the National Natural Science Foundation of China under Grant 61872379, 61806218, 62022091 and 71701205, and China Scholarship Council (No. 201703170183).

References

[1] A. W. Smeulders, M. Worring, S. Santini, A. Gupta, and R. Jain, "Content-based image retrieval at the end of the early years," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 22, no. 12, pp. 1349–1380, 2000.

[2] M. S. Lew, N. Sebe, C. Djeraba, and R. Jain, "Content-based multimedia information retrieval: State of the art and challenges," *ACM Trans. Multimedia Comput. Commun. Appl.*, vol. 2, no. 1, pp. 1–19, 2006.

[3] L. Zheng, L. Shen, L. Tian, S. Wang, J. Wang, and Q. Tian, "Scalable person re-identification: A benchmark," in *ICCV*, 2015, pp. 1116–1124.

[4] X. Liu, S. Zhang, X. Wang, R. Hong, and Q. Tian, "Group-group loss-based global-regional feature learning for vehicle re-identification," *IEEE Trans. Image Process.*, vol. 29, pp. 2638–2652, 2019.

[5] H. Noh, A. Araujo, J. Sim, T. Weyand, and B. Han, "Largescale image retrieval with attentive deep local features," in *ICCV*, 2017, pp. 3456–3465.

[6] U. Chaudhuri, B. Banerjee, and A. Bhattacharya, "Siamese graph convolutional network for content based remote sensing image retrieval," *Comput. Vis. Image Underst.*, vol. 184, pp. 22–30, 2019.

[7] L. R. Nair, K. Subramaniam, and G. Prasannavenkatesan, "A review on multiple approaches to medical image retrieval system," in *Intelligent Computing in Engineering*, 2020, vol. 1125, pp. 501–509.

[8] Z. Liu, P. Luo, S. Qiu, X. Wang, and X. Tang, "Deepfashion: Powering robust clothes recognition and retrieval with rich annotations," in *CVPR*, 2016, pp. 1096–1104.

[9] A. Gordo and D. Larlus, "Beyond instance-level image retrieval: Leveraging captions to learn a global visual representation for semantic retrieval," in *CVPR*, 2017, pp. 6589–6598.

[10] B. Barz and J. Denzler, "Content-based image retrieval and the semantic gap in the deep learning era," in *ICPR*, 2021, pp. 245–260.

[11] X. Wang, X. Han, W. Huang, D. Dong, and M. R. Scott, "Multi-similarity loss with general pair weighting for deep metric learning," in *CVPR*, 2019, pp. 5022–5030.

[12] J. Wang, Y. Song, T. Leung, C. Rosenberg, J. Wang, J. Philbin, B. Chen, and Y. Wu, "Learning fine-grained image similarity with deep ranking," in *CVPR*, 2014, pp. 1386–1393.

[13] A. Babenko and V. Lempitsky, "Aggregating local deep features for image retrieval," in *ICCV*, 2015, pp. 1269–1277.

[14] L. Zheng, Y. Yang, and Q. Tian, "SIFT meets CNN: A decade survey of instance retrieval," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 40, no. 5, pp. 1224–1244, 2018.

[15] W. Zhang, C.-W. Ngo, and X. Cao, "Hyperlink-aware object retrieval," *IEEE Trans. Image Process.*, vol. 25, no. 9, pp. 4186–4198, 2016.

[16] Y. Kalantidis, C. Mellina, and S. Osindero, "Cross-dimensional weighting for aggregated deep convolutional features," in *ECCV*, 2016, pp. 685–701.

[17] L. Zhang and Y. Rui, "Image search from thousands to billions in 20 years," *ACM Trans. Multimedia Comput. Commun. Appl.*, vol. 9, no. 1s, p. 36, 2013.

[18] J. Wan, D. Wang, S. C. H. Hoi, P. Wu, J. Zhu, Y. Zhang, and J. Li, "Deep learning for content-based image retrieval: A comprehensive study," in *ACM MM*, 2014, pp. 157–166.

[19] A. Alzu'bi, A. Amira, and N. Ramzan, "Semantic content-based image retrieval: A comprehensive study," *J. Vis. Commun. Image Represent.*, vol. 32, pp. 20–54, 2015.

[20] X. Li, T. Uricchio, L. Ballan, M. Bertini, C. G. Snoek, and A. D. Bimbo, "Socializing the semantic gap: A comparative survey on image tag assignment, refinement, and retrieval," *ACM Comput. Surv. (CSUR)*, vol. 49, no. 1, pp. 1–39, 2016.

[21] W. Zhou, H. Li, and Q. Tian, "Recent advance in content-based image retrieval: A literature survey," *arXiv preprint arXiv:1706.06064*, 2017.

[22] L. Piras and G. Giacinto, "Information fusion in content based image retrieval: A comprehensive overview," *Inf. Fusion*, vol. 37, pp. 50–60, 2017.

[23] J. Wang, T. Zhang, N. Sebe, H. T. Shen *et al.*, "A survey on learning to hash," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 40, no. 4, pp. 769–790, 2018.

[24] D. G. Lowe, "Distinctive image features from scale-invariant keypoints," *Int. J. Comput. Vis.*, vol. 60, no. 2, pp. 91–110, 2004.

[25] J. Sivic and A. Zisserman, "Video google: A text retrieval approach to object matching in videos," in *CVPR*, 2003, pp. 1470–1477.

[26] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *NeurIPS*, 2012, pp. 1097–1105.

[27] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *CVPR*, 2016, pp. 770–778.

[28] S. Ren, K. He, R. Girshick, and J. Sun, "Faster R-CNN: Towards real-

time object detection with region proposal networks," in *NeurIPS*, 2015, pp. 91–99.
[29] S. Minaee, Y. Y. Boykov, F. Porikli, A. J. Plaza, N. Kehtarnavaz, and D. Terzopoulos, "Image segmentation using deep learning: A survey," *IEEE Trans. Pattern Anal. Mach. Intell.*, 2021.
[30] A. Sharif Razavian, H. Azizpour, J. Sullivan, and S. Carlsson, "CNN features off-the-shelf: an astounding baseline for recognition," in *CVPR workshops*, 2014, pp. 806–813.
[31] Y. Gong, L. Wang, R. Guo, and S. Lazebnik, "Multi-scale orderless pooling of deep convolutional activation features," in *ECCV*, 2014, pp. 392–407.
[32] J. Yosinski, J. Clune, Y. Bengio, and H. Lipson, "How transferable are features in deep neural networks?" in *NeurIPS*, 2014, pp. 3320–3328.
[33] G. Tolias, R. Sicre, and H. Jégou, "Particular object retrieval with integral max-pooling of CNN activations," in *ICLR*, 2015, pp. 1–12.
[34] A. Jiménez, J. M. Alvarez, and X. Giró Nieto, "Class-weighted convolutional features for visual instance search," in *BMVC*, 2017, pp. 1–12.
[35] T.-T. Do, T. Hoang, D.-K. L. Tan, H. Le, T. V. Nguyen, and N.-M. Cheung, "From selective deep convolutional features to compact binary representations for image retrieval," *ACM Trans. Multimedia Comput. Commun. Appl.*, vol. 15, no. 2, pp. 1–22, 2018.
[36] J. Xu, C. Wang, C. Qi, C. Shi, and B. Xiao, "Unsupervised part-based weighting aggregation of deep convolutional features for image retrieval," in *AAAI*, 2018, pp. 7436–7443.
[37] Y. Li, X. Kong, L. Zheng, and Q. Tian, "Exploiting hierarchical activations of neural network for image retrieval," in *ACM MM*, 2016, pp. 132–136.
[38] A. Sharif Razavian, J. Sullivan, A. Maki, and S. Carlsson, "A baseline for visual instance retrieval with deep convolutional networks," in *ICLR*, 2015.
[39] T.-T. Do and N.-M. Cheung, "Embedding based on function approximation for large scale image search," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 40, no. 3, pp. 626–638, 2017.
[40] A. Babenko, A. Slesarev, A. Chigorin, and V. Lempitsky, "Neural codes for image retrieval," in *ECCV*, 2014, pp. 584–599.
[41] N. Garcia and G. Vogiatzis, "Learning non-metric visual similarity for image retrieval," *Image Vis. Comput.*, vol. 82, pp. 18–25, 2019.
[42] E.-J. Ong, S. Husain, and M. Bober, "Siamese network of deep fisher-vector descriptors for image retrieval," *arXiv preprint arXiv:1702.00338*, 2017.
[43] A. Gordo, J. Almazán, J. Revaud, and D. Larlus, "Deep image retrieval: Learning global representations for image search," in *ECCV*, 2016, pp. 241–257.
[44] R. Arandjelovic, P. Gronat, A. Torii, T. Pajdla, and J. Sivic, "NetVLAD: CNN architecture for weakly supervised place recognition," in *CVPR*, 2015, pp. 5297–5307.
[45] F. Radenović, G. Tolias, and O. Chum, "CNN image retrieval learns from BoW: Unsupervised fine-tuning with hard examples," in *ECCV*, 2016, pp. 3–20.
[46] J. Xu, C. Wang, C. Qi, C. Shi, and B. Xiao, "Iterative manifold embedding layer learned by incomplete data for large-scale image retrieval," *IEEE Trans. Multimedia*, vol. 21, no. 6, pp. 1551–1562, 2017.
[47] F. Radenović, G. Tolias, and O. Chum, "Fine-tuning CNN image retrieval with no human annotation," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 41, no. 7, pp. 1655–1668, 2018.
[48] C. Liu, G. Yu, M. Volkovs, C. Chang, H. Rai, J. Ma, and S. K. Gorti, "Guided similarity separation for image retrieval," in *NeurIPS*, 2019, pp. 1554–1564.
[49] C. Chang, G. Yu, C. Liu, and M. Volkovs, "Explore-exploit graph traversal for image retrieval," in *CVPR*, 2019, pp. 9423–9431.
[50] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *arXiv preprint arXiv:1409.1556*, 2014.
[51] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich, "Going deeper with convolutions," in *CVPR*, 2015, pp. 1–9.
[52] F. Tan, J. Yuan, and V. Ordonez, "Instance-level image retrieval using reranking transformers," *ICCV*, 2021.
[53] S. Gkelios, Y. Boutalis, and S. A. Chatzichristofis, "Investigating the vision transformer model for image retrieval tasks," *arXiv preprint arXiv:2101.03771*, 2021.
[54] Y. Liu, Y. Guo, S. Wu, and M. S. Lew, "Deepindex for accurate and efficient image retrieval," in *ICMR*, 2015, pp. 43–50.
[55] K. Yan, Y. Wang, D. Liang, T. Huang, and Y. Tian, "CNN vs. SIFT for image retrieval: Alternative or complementary?" in *ACM MM*, 2016, pp. 407–411.
[56] W. Zhou, H. Li, J. Sun, and Q. Tian, "Collaborative index embedding for image retrieval," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 40, no. 5, pp. 1154–1166, 2017.
[57] H. Liu, Y. Tian, Y. Yang, L. Pang, and T. Huang, "Deep relative distance learning: Tell the difference between similar vehicles," in *CVPR*, 2016, pp. 2167–2175.
[58] K. Ozaki and S. Yokoo, "Large-scale landmark retrieval/recognition under a noisy and diverse dataset," in *CVPR Workshop*, 2019.
[59] H. Jégou, M. Douze, C. Schmid, and P. Pérez, "Aggregating local descriptors into a compact image representation," in *CVPR*, 2010, pp. 3304–3311.
[60] F. Perronnin and C. Dance, "Fisher kernels on visual vocabularies for image categorization," in *CVPR*, 2007, pp. 1–8.
[61] J. Yue-Hei Ng, F. Yang, and L. S. Davis, "Exploiting local features from deep networks for image retrieval," in *CVPR workshops*, 2015, pp. 53–61.
[62] O. Siméoni, Y. Avrithis, and O. Chum, "Local features and visual words emerge in activations," in *CVPR*, 2019, pp. 11 651–11 660.
[63] K. Reddy Mopuri and R. Venkatesh Babu, "Object level deep feature pooling for compact image representation," in *CVPR Workshops*, 2015, pp. 62–70.
[64] O. Morere, J. Lin, A. Veillard, L.-Y. Duan, V. Chandrasekhar, and T. Poggio, "Nested invariance pooling and RBM hashing for image instance retrieval," in *ICMR*, 2017, pp. 260–268.
[65] T. Weyand, A. Araujo, B. Cao, and J. Sim, "Google landmarks dataset v2-a large-scale benchmark for instance-level recognition and retrieval," in *CVPR*, 2020, pp. 2575–2584.
[66] J. Song, T. He, L. Gao, X. Xu, and H. T. Shen, "Deep region hashing for efficient large-scale instance search from images," in *AAAI*, vol. 32, 2018.
[67] K. Lin, J. Lu, C.-S. Chen, J. Zhou, and M.-T. Sun, "Unsupervised deep learning of compact binary descriptors," *IEEE Trans. Pattern Anal. Mach. Intell.*, 2018.
[68] E. Mohedano, K. McGuinness, N. E. O'Connor, A. Salvador, F. Marqués, and X. Giro-i Nieto, "Bags of local convolutional features for scalable instance search," in *ICMR*, 2016, pp. 327–331.
[69] A. S. Razavian, J. Sullivan, S. Carlsson, and A. Maki, "Visual instance retrieval with deep convolutional networks," *ITE Trans. Media Technol. Appl.*, vol. 4, no. 3, pp. 251–258, 2016.
[70] T. Ng, V. Balntas, Y. Tian, and K. Mikolajczyk, "SOLAR: Second-order loss and attention for image retrieval," in *ECCV*, 2020, pp. 253–270.
[71] S. Pang, J. Xue, J. Zhu, L. Zhu, and Q. Tian, "Unifying sum and weighted aggregations for efficient yet effective image representation computation," *IEEE Trans. Image Process.*, vol. 28, no. 2, pp. 841–852, 2018.
[72] B. Cao, A. Araujo, and J. Sim, "Unifying deep local and global features for efficient image search," in *ECCV*, 2020, pp. 726–743.
[73] M. A. Fischler and R. C. Bolles, "Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography," *Communications of the ACM*, vol. 24, no. 6, pp. 381–395, 1981.
[74] G. Tolias, Y. Avrithis, and H. Jégou, "Image search with selective match kernels: aggregation across single and multiple images," *Int. J. Comput. Vis.*, vol. 116, no. 3, pp. 247–261, 2016.
[75] M. Teichmann, A. Araujo, M. Zhu, and J. Sim, "Detect-to-retrieve: Efficient regional aggregation for image search," in *CVPR*, 2019, pp. 5109–5118.
[76] S. Pang, J. Ma, J. Zhu, J. Xue, and Q. Tian, "Improving object retrieval quality by integration of similarity propagation and query expansion," *IEEE Trans. Multimedia*, vol. 21, no. 3, pp. 760–770, 2018.
[77] L. Liu, W. Ouyang, X. Wang, P. Fieguth, J. Chen, X. Liu, and M. Pietikäinen, "Deep learning for generic object detection: A survey," *Int. J. Comput. Vis.*, vol. 128, no. 2, pp. 261–318, 2020.
[78] A. Khan, A. Sohail, U. Zahoora, and A. S. Qureshi, "A survey of the recent architectures of deep convolutional neural networks," *Artif. Intell. Rev.*, vol. 53, no. 8, pp. 5455–5516, 2020.
[79] W. Liu, Z. Wang, X. Liu, N. Zeng, Y. Liu, and F. E. Alsaadi, "A survey of deep neural network architectures and their applications," *Neurocomputing*, vol. 234, pp. 11–26, 2017.
[80] A. El-Nouby, N. Neverova, I. Laptev, and H. Jégou, "Training vision transformers for image retrieval," *arXiv preprint arXiv:2102.05644*, 2021.
[81] C. Qi, C. Shi, J. Xu, C. Wang, and B. Xiao, "Spatial weighted fisher vector for image retrieval," in *ICME*, 2017, pp. 463–468.
[82] H. J. Kim, E. Dunn, and J.-M. Frahm, "Learned contextual feature reweighting for image geo-localization," in *CVPR*, 2017, pp. 3251–3260.
[83] E. Mohedano, K. McGuinness, X. Giró-i Nieto, and N. E. O'Connor, "Saliency weighted convolutional features for instance search," in *CBMI*, 2018, pp. 1–6.
[84] F. Yang, J. Li, S. Wei, Q. Zheng, T. Liu, and Y. Zhao, "Two-stream attentive CNNs for image retrieval," in *ACM MM*, 2017, pp. 1513–1521.
[85] H.-F. Yang, K. Lin, and C.-S. Chen, "Supervised learning of semantics-preserving hash via deep convolutional neural networks," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 40, no. 2, pp. 437–451, 2018.
[86] Y. Liu, J. Song, K. Zhou, L. Yan, L. Liu, F. Zou, and L. Shao, "Deep self-taught hashing for image retrieval," *IEEE Trans Cybern.*, vol. 49, no. 6, pp. 2229–2241, 2018.

[87] A. Gordo, J. Almazan, J. Revaud, and D. Larlus, "End-to-end learning of deep visual representations for image retrieval," *Int. J. Comput. Vis.*, vol. 124, no. 2, pp. 237–254, 2017.

[88] H. Azizpour, A. S. Razavian, J. Sullivan, A. Maki, and S. Carlsson, "Factors of transferability for a generic convnet representation," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 38, no. 9, pp. 1790–1802, 2016.

[89] W. Zhao, H. Luo, J. Peng, and J. Fan, "Spatial pyramid deep hashing for large-scale image retrieval," *Neurocomputing*, vol. 243, pp. 166–173, 2017.

[90] L. Zheng, S. Wang, J. Wang, and Q. Tian, "Accurate image search with multi-scale contextual evidences," *Int. J. Comput. Vis.*, vol. 120, no. 1, pp. 1–13, 2016.

[91] J. Cao, L. Liu, P. Wang, Z. Huang, C. Shen, and H. T. Shen, "Where to focus: Query adaptive matching for instance retrieval using convolutional feature maps," *arXiv preprint arXiv:1606.06811*, 2016.

[92] C. L. Zitnick and P. Dollár, "Edge boxes: Locating object proposals from edges," in *ECCV*, 2014, pp. 391–405.

[93] T. Yu, Y. Wu, S. D. Bhattacharjee, and J. Yuan, "Efficient object instance search using fuzzy objects matching." in *AAAI*, 2017, pp. 4320–4326.

[94] S. Sun, W. Zhou, Q. Tian, and H. Li, "Scalable object retrieval with compact image representation from generic object regions," *ACM Trans. Multimedia Comput. Commun. Appl.*, vol. 12, no. 2, pp. 1–21, 2015.

[95] J. Mairal, P. Koniusz, Z. Harchaoui, and C. Schmid, "Convolutional kernel networks," in *NeurIPS*, 2014, pp. 2627–2635.

[96] A. Salvador, X. Giró-i Nieto, F. Marqués, and S. Satoh, "Faster R-CNN features for instance search," in *CVPR Workshops*, 2016, pp. 9–16.

[97] L. Zheng, Y. Zhao, S. Wang, J. Wang, and Q. Tian, "Good practice in CNN feature transfer," *CoRR*, vol. abs/1604.00133, 2016.

[98] Y. Lou, Y. Bai, S. Wang, and L.-Y. Duan, "Multi-scale context attention network for image retrieval," in *ACM MM*, 2018, pp. 1128–1136.

[99] Y. Li, Y. Xu, J. Wang, Z. Miao, and Y. Zhang, "MS-RMAC: Multiscale regional maximum activation of convolutions for image retrieval," *IEEE Signal Process Lett.*, vol. 24, no. 5, pp. 609–613, 2017.

[100] X. Xiang, Z. Wang, Z. Zhao, and F. Su, "Multiple saliency and channel sensitivity network for aggregated convolutional feature," in *AAAI*, vol. 33, no. 01, 2019, pp. 9013–9020.

[101] S. Pang, J. Ma, J. Xue, J. Zhu, and V. Ordonez, "Deep feature aggregation and image re-ranking with heat diffusion for image retrieval," *IEEE Trans. Multimedia*, vol. 21, no. 6, pp. 1513–1523, 2018.

[102] W. Yu, K. Yang, H. Yao, X. Sun, and P. Xu, "Exploiting the complementary strengths of multi-layer CNN features for image retrieval," *Neurocomputing*, vol. 237, pp. 235–241, 2017.

[103] M. Yang, D. He, M. Fan, B. Shi, X. Xue, F. Li, E. Ding, and J. Huang, "Dolg: Single-stage image retrieval with deep orthogonal fusion of local and global features," in *ICCV*, 2021, pp. 11 772–11 781.

[104] Z. Zhang, Y. Xie, W. Zhang, and Q. Tian, "Effective image retrieval via multilinear multi-index fusion," *IEEE Trans. Multimedia*, vol. 21, no. 11, pp. 2878–2890, 2019.

[105] Q. Wang, J. Lai, Z. Yang, K. Xu, P. Kan, W. Liu, and L. Lei, "Improving cross-dimensional weighting pooling with multi-scale feature fusion for image retrieval," *Neurocomputing*, vol. 363, pp. 17–26, 2019.

[106] L. Zheng, S. Wang, L. Tian, F. He, Z. Liu, and Q. Tian, "Query-adaptive late fusion for image search and person re-identification," in *CVPR*, 2015, pp. 1741–1750.

[107] H. Xuan, R. Souvenir, and R. Pless, "Deep randomized ensembles for metric learning," in *ECCV*, 2018, pp. 723–734.

[108] B.-C. Chen, L. S. Davis, and S.-N. Lim, "An analysis of object embeddings for image retrieval," *arXiv preprint arXiv:1905.11903*.

[109] F. Wang, W.-L. Zhao, C.-W. Ngo, and B. Merialdo, "A hamming embedding kernel with informative bag-of-visual words for video semantic indexing," *ACM Trans. Multimedia Comput. Commun. Appl.*, vol. 10, no. 3, pp. 1–20, 2014.

[110] A. Mishchuk, D. Mishkin, F. Radenović, and J. Matas, "Working hard to know your neighbor's margins: Local descriptor learning loss," in *NeurIPS*, 2017, pp. 4827–4838.

[111] A. Mukherjee, J. Sil, A. Sahu, and A. S. Chowdhury, "A bag of constrained informative deep visual words for image retrieval," *Pattern Recognition Letters*, vol. 129, pp. 158–165, 2020.

[112] M. Paulin, M. Douze, Z. Harchaoui, J. Mairal, F. Perronin, and C. Schmid, "Local convolutional features with unsupervised training for image retrieval," in *ICCV*, 2015, pp. 91–99.

[113] H. Jegou, F. Perronnin, M. Douze, J. Sánchez, P. Perez, and C. Schmid, "Aggregating local image descriptors into compact codes," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 34, no. 9, pp. 1704–1716, 2012.

[114] J. Sánchez, F. Perronnin, T. Mensink, and J. Verbeek, "Image classification with the fisher vector: Theory and practice," *Int. J. Comput. Vis.*, vol. 105, no. 3, pp. 222–245, 2013.

[115] J. Cao, Z. Huang, and H. T. Shen, "Local deep descriptors in bag-of-words for image retrieval," in *ACM MM*, 2017, pp. 52–58.

[116] M. Dusmanu, I. Rocco, T. Pajdla, M. Pollefeys, J. Sivic, A. Torii, and T. Sattler, "D2-net: A trainable cnn for joint description and detection of local features," in *CVPR*, 2019, pp. 8092–8101.

[117] K. Mikolajczyk and C. Schmid, "Scale & affine invariant interest point detectors," *Int. J. Comput. Vis.*, vol. 60, no. 1, pp. 63–86, 2004.

[118] G. Tolias, T. Jenicek, and O. Chum, "Learning and aggregating deep local descriptors for instance-level recognition," in *ECCV*, 2020, pp. 460–477.

[119] J. Yang, J. Liang, H. Shen, K. Wang, P. L. Rosin, and M.-H. Yang, "Dynamic match kernel with deep convolutional features for image retrieval," *IEEE Trans. Image Process.*, vol. 27, no. 11, pp. 5288–5302, 2018.

[120] J. Kim and S.-E. Yoon, "Regional attention based deep feature for image retrieval," in *BMVC*, 2018, pp. 209–223.

[121] S. Wei, L. Liao, J. Li, Q. Zheng, F. Yang, and Y. Zhao, "Saliency inside: Learning attentive CNNs for content-based image retrieval," *IEEE Trans. Image Process.*, vol. 28, no. 9, pp. 4580–4593, 2019.

[122] T.-T. Do, D.-K. Le Tan, T. T. Pham, and N.-M. Cheung, "Simultaneous feature aggregating and hashing for large-scale image search," in *CVPR*, 2017, pp. 6618–6627.

[123] H. Jegou, M. Douze, and C. Schmid, "Hamming embedding and weak geometric consistency for large scale image search," in *ECCV*, 2008, pp. 304–317.

[124] J. Philbin, O. Chum, M. Isard, J. Sivic, and A. Zisserman, "Object retrieval with large vocabularies and fast spatial matching," in *CVPR*, 2007, pp. 1–8.

[125] M. Boudiaf, J. Rony, I. M. Ziko, E. Granger, M. Pedersoli, P. Piantanida, and I. B. Ayed, "A unifying mutual information view of metric learning: cross-entropy vs. pairwise losses," in *ECCV*, 2020, pp. 548–564.

[126] K. Musgrave, S. Belongie, and S.-N. Lim, "A metric learning reality check," in *ECCV*, 2020, pp. 681–699.

[127] Y. Lv, W. Zhou, Q. Tian, S. Sun, and H. Li, "Retrieval oriented deep feature learning with complementary supervision mining," *IEEE Trans. Image Process.*, vol. 27, no. 10, pp. 4945–4957, 2018.

[128] J. Lin, O. Morere, A. Veillard, L.-Y. Duan, H. Goh, and V. Chandrasekhar, "Deephash for image instance retrieval: Getting regularization, depth and fine-tuning right," in *ICMR*, 2017, pp. 133–141.

[129] J. Cao, Z. Huang, P. Wang, C. Li, X. Sun, and H. T. Shen, "Quartet-net learning for visual instance retrieval," in *ACM MM*, 2016, pp. 456–460.

[130] W. Min, S. Mei, Z. Li, and S. Jiang, "A two-stage triplet network training framework for image retrieval," *IEEE Trans. Multimedia*, vol. 22, no. 12, pp. 3128–3138, 2020.

[131] J. Revaud, J. Almazán, R. S. Rezende, and C. R. d. Souza, "Learning with average precision: Training image retrieval with a listwise loss," in *ICCV*, 2019, pp. 5107–5116.

[132] A. Brown, W. Xie, V. Kalogeiton, and A. Zisserman, "Smooth-AP: Smoothing the path towards large-scale image retrieval," in *ECCV*, 2020, pp. 677–694.

[133] A. Iscen, G. Tolias, Y. Avrithis, and O. Chum, "Mining on manifolds: Metric learning without labels," in *CVPR*, 2018, pp. 7642–7651.

[134] A. Iscen, Y. Avrithis, G. Tolias, T. Furon, and O. Chum, "Fast spectral ranking for similarity search," in *CVPR*, 2018, pp. 7632–7641.

[135] A. Iscen, G. Tolias, Y. Avrithis, T. Furon, and O. Chum, "Efficient diffusion on region manifolds: Recovering small objects with compact CNN representations," in *CVPR*, 2017, pp. 2077–2086.

[136] M. Donoser and H. Bischof, "Diffusion processes for retrieval revisited," in *CVPR*, 2013, pp. 1320–1327.

[137] Y. Zhao, L. Wang, L. Zhou, Y. Shi, and Y. Gao, "Modelling diffusion process by deep neural networks for image retrieval," in *BMVC*, 2018, pp. 161–174.

[138] B. Song, X. Bai, Q. Tian, and L. J. Latecki, "Regularized diffusion process on bidirectional context for object retrieval," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 41, no. 5, pp. 1213–1226, 2018.

[139] T. N. Kipf and M. Welling, "Semi-supervised classification with graph convolutional networks," in *ICLR*, 2017.

[140] T. Maria and T. Anastasios, "Deep convolutional image retrieval: A general framework," *Signal Process. Image Commun.*, vol. 63, pp. 30–43, 2018.

[141] M. Caron, P. Bojanowski, A. Joulin, and M. Douze, "Deep clustering for unsupervised learning of visual features," in *ECCV*, 2018, pp. 132–149.

[142] W. Jiang, Y. Wu, C. Jing, T. Yu, and Y. Jia, "Unsupervised deep quantization for object instance search," *Neurocomputing*, vol. 362, pp. 60–71, 2019.

[143] B. Ke, J. Shao, Z. Huang, and H. T. Shen, "Feature reconstruction by laplacian eigenmaps for efficient instance search," in *ICMR*, 2018, pp. 231–239.

[144] T. Shen, Z. Luo, L. Zhou, R. Zhang, S. Zhu, T. Fang, and L. Quan, "Matchable image retrieval by learning from surface reconstruction," in *ACCV*, 2018, pp. 415–431.

[145] J. Hu, R. Ji, H. Liu, S. Zhang, C. Deng, and Q. Tian, "Towards visual feature translation," in *CVPR*, 2019, pp. 3004–3013.

[146] C. Bai, H. Li, J. Zhang, L. Huang, and L. Zhang, "Unsupervised adversarial instance-level image retrieval," *IEEE Trans. Multimedia*, 2021.
[147] M. Paulin, J. Mairal, M. Douze, Z. Harchaoui, F. Perronnin, and C. Schmid, "Convolutional patch representations for image retrieval: an unsupervised approach," *Int. J. Comput. Vis.*, vol. 121, no. 1, pp. 149–168, 2017.
[148] D. Nister and H. Stewenius, "Scalable recognition with a vocabulary tree," in *CVPR*, 2006, pp. 2161–2168.
[149] J. Philbin, O. Chum, M. Isard, J. Sivic, and A. Zisserman, "Lost in quantization: Improving particular object retrieval in large scale image databases," in *CVPR*, 2008, pp. 1–8.
[150] F. Radenovic, A. Iscen, G. Tolias, Y. Avrithis, and O. Chum, "Revisiting oxford and paris: Large-scale image retrieval benchmarking," in *CVPR*, 2018.
[151] S. Wang and S. Jiang, "INSTRE: a new benchmark for instance-level object retrieval and recognition," *ACM Trans. Multimedia Comput. Commun. Appl.*, vol. 11, no. 3, pp. 37:1–37:21, 2015.
[152] A. Alzu'bi, A. Amira, and N. Ramzan, "Content-based image retrieval with compact deep convolutional features," *Neurocomputing*, vol. 249, pp. 95–105, 2017.
[153] S. S. Husain and M. Bober, "REMAP: Multi-layer entropy-guided pooling of dense CNN features for image retrieval," *IEEE Trans. Image Process.*, vol. 28, no. 10, pp. 5201–5213, 2019.
[154] L. T. Alemu and M. Pelillo, "Multi-feature fusion for image retrieval using constrained dominant sets," *Image Vis Comput*, vol. 94, p. 103862, 2020.
[155] L. P. Valem and D. C. G. Pedronette, "Graph-based selective rank fusion for unsupervised image retrieval," *Pattern Recognit Lett*, 2020.
[156] F. Yang, R. Hinami, Y. Matsui, S. Ly, and S. Satoh, "Efficient image retrieval via decoupling diffusion into online and offline processing," in *AAAI*, vol. 33, 2019, pp. 9087–9094.
[157] H.-F. Yang, K. Lin, and C.-S. Chen, "Cross-batch reference learning for deep classification and retrieval," in *ACM MM*, 2016, pp. 1237–1246.
[158] J. Ouyang, W. Zhou, M. Wang, Q. Tian, and H. Li, "Collaborative image relevance learning for visual re-ranking," *IEEE Trans. Multimedia*, 2020.
[159] K. Chatfield, K. Simonyan, A. Vedaldi, and A. Zisserman, "Return of the devil in the details: Delving deep into convolutional nets," in *BMVC*, 2014.
[160] P. Sermanet, D. Eigen, X. Zhang, M. Mathieu, R. Fergus, and Y. LeCun, "Overfeat: Integrated recognition, localization and detection using convolutional networks," in *ICLR*, 2014.
[161] W. Chen, Y. Liu, W. Wang, T. Tuytelaars, E. M. Bakker, and M. Lew, "On the exploration of incremental learning for fine-grained image retrieval," in *BMVC*, 2020.
[162] J. Li, R. Ji, H. Liu, X. Hong, Y. Gao, and Q. Tian, "Universal perturbation attack against image retrieval," in *ICCV*, 2019, pp. 4899–4908.
[163] F. Yang, Z. Zhong, H. Liu, Z. Wang, Z. Luo, S. Li, N. Sebe, and S. Satoh, "Learning to attack real-world models for person re-identification via virtual-guided meta-learning," in *AAAI*, vol. 35, no. 4, 2021, pp. 3128–3135.
[164] G. Tolias, F. Radenovic, and O. Chum, "Targeted mismatch adversarial attack: Query with a flower to retrieve the tower," in *ICCV*, 2019, pp. 5037–5046.