Deep Image Retrieval: A Survey

Wei Chen, Yu Liu, Weiping Wang, Erwin Bakker, Theodoros Georgiou,
Paul Fieguth, Li Liu, *Senior Member, IEEE*, and Michael S. Lew

Abstract—In recent years a vast amount of visual content has been generated and shared from various fields, such as social media platforms, medical images, and robotics. This abundance of content creation and sharing has introduced new challenges. In particular, searching databases for similar content, *i.e.*, content based image retrieval (CBIR), is a long-established research area, and more efficient and accurate methods are needed for real time retrieval. Artificial intelligence has made progress in CBIR and has significantly facilitated the process of intelligent search. In this survey we organize and review recent CBIR works that are developed based on deep learning algorithms and techniques, including insights and techniques from recent papers. We identify and present the commonly-used databases, benchmarks, and evaluation methods used in the field. We collect common challenges and propose promising future directions. More specifically, we focus on image retrieval with deep learning and organize the state of the art methods according to the types of deep network structure, deep features, feature enhancement methods, and network fine-tuning strategies. Our survey considers a wide variety of recent methods, aiming to promote a global view of the field of category-based CBIR.

Index Terms—Content based image retrieval, deep learning, convolutional neural networks, literature survey

1 Introduction

CONTENT based image retrieval (CBIR) is the problem of searching for semantically matched or similar images in a large image gallery by analyzing their visual content, given a query image that describes the user's needs, as illustrated in Figure 1(a). CBIR has been a longstanding research topic in the computer vision and multimedia community [1, 2]. With the present exponentially increasing amount of image and video data, a development of appropriate information systems to efficiently manage such large image collections is badly needed, with image searching being one of the most indispensable techniques used for interacting with visual collections. Therefore there is nearly endless potential for applications of CBIR, such as person re-identification [3], remote sensing [4], medical image search [5], and shopping recommendation in online marketplaces [6], among many others.

CBIR can broadly be grouped into instance level retrieval and category level retrieval, as depicted in Figure 1(b). In instance level image retrieval, a query image of a particular object or scene (*e.g.*, the Eiffel Tower) is given and the goal is to find images containing the same object or scene that may be captured under different viewpoints, illumination conditions, or subject to occlusions [7, 8]. In contrast, for category level image retrieval the goal is to find images of the same class as the query (*e.g.*, dogs, cars, *etc.*). Instance level retrieval is more challenging and promising as it satisfies specific objectives for many applications. Notice that we limit the focus of this survey to instance-level image retrieval and in the following, if not further specified, "image retrieval" and "instance retrieval" are considered equivalent and will be used interchangeably.

Finding a desired image can require 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 [7, 8, 9, 10, 11]. To enable accurate and efficient retrieval of massive image collections, *compact yet rich feature representations* are at the core of CBIR.

Fig. 1: Illustration of (a) the CBIR problem, and (b) instance-level retrieval versus category-level retrieval.

In the past two decades, remarkable progress has been made in image feature representations, which mainly consist of two important stages: feature engineering and feature learning

Wei Chen, Erwin Bakker, Theodoros Georgiou, and Michael S. Lew are with Leiden Institute of Advanced Computer Science, Leiden University, The Netherlands.
Yu Liu is with ESAT-PSI, KU Leuven, Belgium.
Weiping Wang is with College of Systems Engineering, NUDT, China.
Paul Fieguth is with the Systems Design Engineering Department, University of Waterloo, Canada.
Li Liu is with College of Systems Engineering, NUDT, China, and with Center for Machine Vision and Signal Analysis, University of Oulu, Finland.
Corresponding author: Professor Li Liu, li.liu@oulu.fi

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

Title	Year	Published in	Main Contents
Image Search from Thousands to Billions in 20 Years [12]	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 [13]	2014	ACM MM	This paper introduces supervised metric learning methods for fine-tuning AlexNet. It focuses on deep learning methods but lacks many details in deep instance-based image retrieval.
Semantic Content-based Image Retrieval: A Comprehensive Study [14]	2015	JVCI	This paper presents a comprehensive study on the entire process and the achievements of CBIR. Mainly conventional methods are presented; deep learning is introduced as a sub-section with limited details.
Socializing the Semantic Gap: A Comparative Survey on Image Tag Assignment, Refinement, and Retrieval [15]	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 [16]	2017	arXiv	This survey presents image retrieval from 2003 to 2016, including query categories, feature representation, feature organization. Neural networks are introduced in a section and mainly discussed as a future direction.
Information Fusion in Content-based Image Retrieval: A Comprehensive Overview [17]	2017	Information Fusion	This paper presents information fusion strategies in content-based image retrieval. Deep convolutional networks for feature learning are introduced briefly but not covered thoroughly.
A Survey on Learning to Hash [18]	2018	T-PAMI	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 [8]	2018	T-PAMI	This paper presents a comprehensive review of instance retrieval using SIFT and CNN-based methods. Feature learning methods using CNNs are lacking details.
Deep Image Retrieval: A Survey	2020	Ours	Our survey focuses on deep learning methods. We expand the review with in-depth details on CBIR, including structures of deep networks, types of deep features, feature enhancement strategies, and network fine-tuning.

(particularly deep learning). In the feature engineering era (*i.e.*, pre-deep learning), the field was dominated by milestone hand-engineered feature descriptors, such as the Scale-Invariant Feature Transform (SIFT) [19]. The feature learning stage, the deep learning era since 2012, begins with artificial neural networks, particularly the breakthrough ImageNet and the Deep Convolutional Neural Network (DCNN) AlexNet [20]. Since then, deep learning techniques have impacted a broad range of research areas, since DCNNs can learn powerful feature representations with multiple levels of abstraction directly from data, bypassing multiple steps in traditional feature engineering. Deep learning techniques have attracted enormous attention and have brought about considerable breakthroughs in many computer vision tasks, including image classification [20, 21, 22], object detection [23], semantic segmentation [24], and image retrieval [10, 13, 14].

Excellent surveys for traditional image retrieval methods can be found in [1, 2, 8]. This paper, in contrast, focuses on deep learning based methods, and a comparison of our work with other published surveys [8, 14, 15, 16] is shown in Table 1. Deep learning for image retrieval is comprised of the essential stages shown in Figure 2 and various methods, focusing on one or more stages, have been proposed to improve retrieval accuracy and efficiency. In this survey, we include comprehensive details about these methods, including the structures of deep networks, feature fusion, feature enhancement methods, and network fine-tuning strategies, motivated by the following questions that have been driving research in this domain:

1) *By using off-the-shelf models only, how do deep features outperform hand-crafted features?*
2) *In case of domain shifts across training datasets, how can we adapt off-the-shelf models to maintain or even improve retrieval performance?*
3) *Since deep features are generally high-dimensional, how can we effectively utilize them to perform efficient image retrieval, especially for large-scale datasets?*

1.1 Summary of Progress since 2012

After a highly successful image retrieval implementation based on AlexNet [20], significant exploration of DCNNs for retrieval tasks has been undertaken, broadly along the lines of the preceding three questions just identified, above. That is, the DCNN methods are divided into (1) off-the-shelf and (2) fine-tuned models, as shown in Figure 3, with parallel work on (3) effective features. Whether a DCNN is considered off-the-shelf or fine-tuned depends on whether the DCNN parameters are updated [25] or are based on DCNNs with fixed parameters [25, 26, 27]. For feature maps researchers have proposed encoding and aggregation methods, such as R-MAC [28], CroW [10], and SPoC [7].

Recent progress for improving image retrieval can be categorized into network-level and feature-level perspectives, for which a detailed categorization is shown in Figure 4. Broadly this survey will examine the four areas outlined as follows:

(1) *Improvements in Network Architectures* (Section 2)

Using stacked linear filters (*e.g.* convolution) and non-linear activation functions (ReLU, *etc.*), deep networks with different depths obtain features at different levels. Deeper networks with more layers provide a more powerful learning capacity so as to extract high-level abstract and semantic-aware features [21, 46]. It is possible to concatenate multi-scale features in parallel, such as the Inception module in GoogLeNet [47], which we refer to as widening.

(2) *Deep Feature Extraction* (Section 3.1)

Neurons of FC layers and convolutional layers have different receptive fields, which provides three ways to extract features: local features from convolutional layers [7, 59], global features from FC layers [32, 60] and fusions of two kinds of features [61, 62], where the fusion scheme includes layer-level and model-level methods. Deep features can be extracted from the whole image or from image patches, which corresponds to single pass and multiple pass feedforward schemes, respectively.

(3) *Deep Feature Enhancement* (Section 3.2)

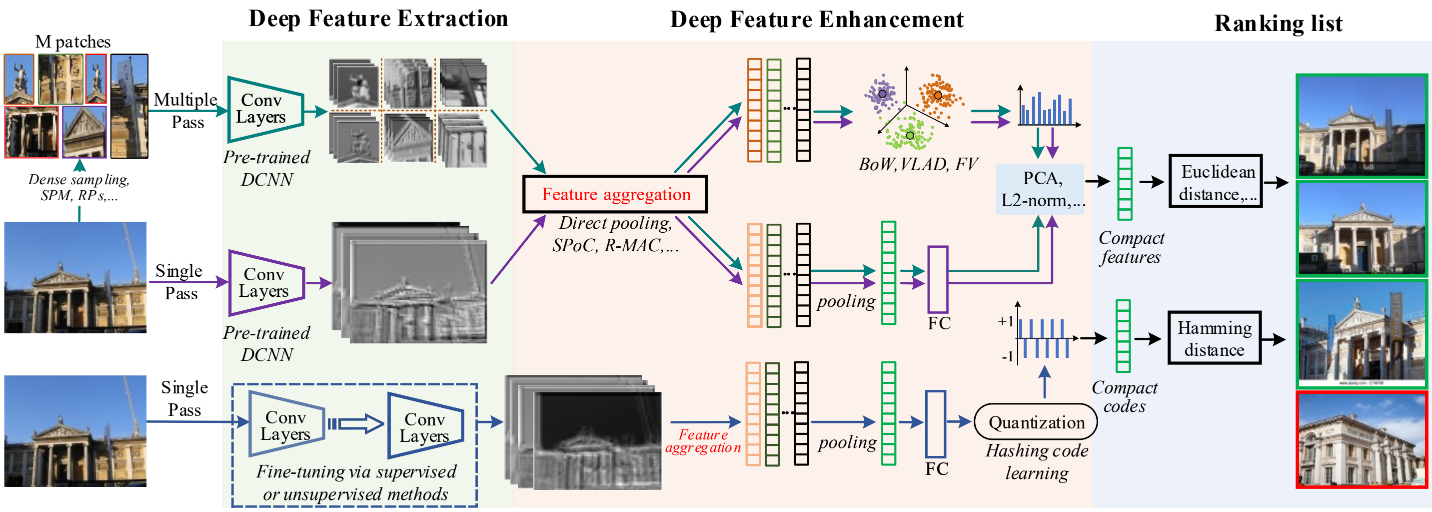

Fig. 2: In deep image retrieval, feature embedding and aggregation methods are used to enhance the discrimination of deep features. Similarity is measured on these enhanced features using Euclidean or Hamming distances.

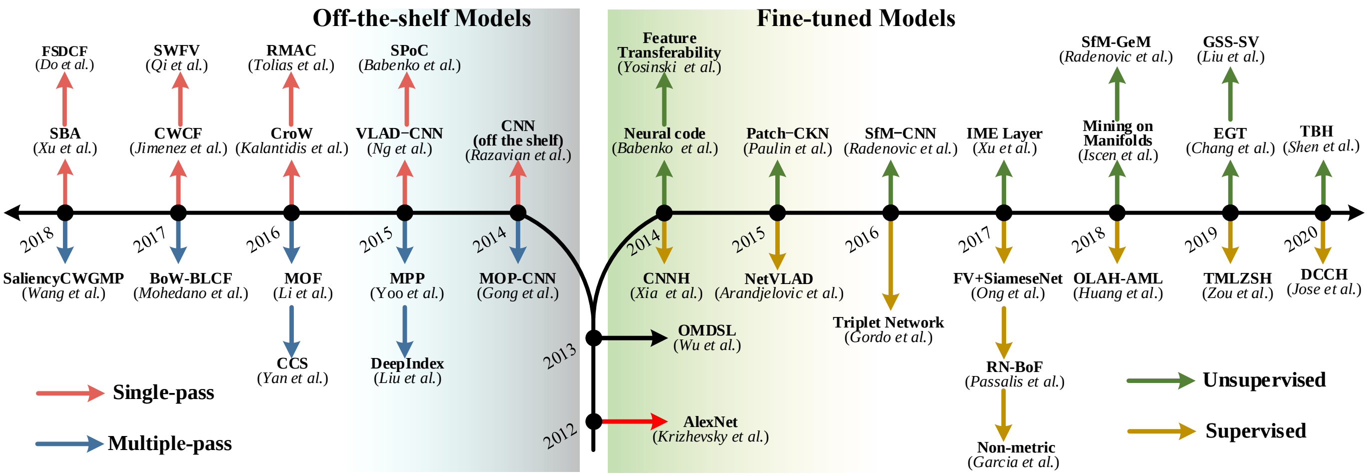

Fig. 3: Representative methods in deep image retrieval, which are most fundamentally categorized according to whether the DCNN parameters are updated [25]. Off-the-shelf models (left) have model parameters which are not further updated or tuned when extracting features for image retrieval. The relevant methods focus on improving representations quality either by feature enhancement [10, 29, 30, 31] when using single pass schemes or by extracting representations for image patches [32] when using multiple pass schemes. In contrast, in fine-tuned models (right) the model parameters are updated for the features to be fine-tuned towards the retrieval task and addresses the issue of domain shifts. The fine-tuning may be supervised [33, 34, 35, 36, 37, 38, 39] or unsupervised [40, 41, 42, 43, 44, 45]. See Sections 3 and 4 for details.

Feature enhancement is used to improve the discriminative ability of deep features. Directly, aggregate features can be trained simultaneously with deep networks [17]; alternatively, feature embedding methods including BoW [63], VLAD [64], and FV [65] embed local features into global ones. These methods are trained with deep networks separately (codebook-based) or jointly (codebook-free). Further, hashing methods [18] encode the real-valued features into binary codes to improve retrieval efficiency. The feature enhancement strategy can significantly influence the efficiency of image retrieval.

(4) *Network Fine-tuning for Learning Representations* (Section 4)

Deep networks pre-trained on source datasets for image classification are transferred to new datasets for retrieval tasks. However, the retrieval performance is influenced by the domain shifts between the datasets. Therefore, it is necessary to fine-tune the deep networks to the specific domain [34, 56, 66], which can be realized by using supervised fine-tuning methods. However in most cases image labeling or annotation is time-consuming and difficult, so it is necessary to develop unsupervised methods for network fine-tuning.

1.2 Key Challenges

Deep learning has been successful in learning very powerful features. Nevertheless, several significant challenges remain with regards to

1) *reducing the semantic gap*,
2) *improving retrieval scalability*, and
3) *balancing retrieval accuracy and efficiency*.

We finish the introduction to this survey with a brief overview of each of these challenges:

1. Reducing the semantic gap: The semantic gap characterizes the difference, in any application, between the high-level concepts of humans and the low-level features typically derived

Deep Learning for Image Retrieval

- **Improvement in Deep Network Architecture (Section 2)**
  - Deepen Networks: Alexnet [20], VGG [46], ResNet [21]
  - Widen Networks: GoogLeNet [47], DenseNet [22], *etc.*
- Retrieval with Off-the-Shelf DCNN Models (Section 3)
  - **Deep Feature Extraction (Section 3.1)**
    - Network Feedforward Scheme (Section 3.1.1)
      - Single Feedforward Pass: MAC [48], R-MAC [28]
      - Multiple Feedforward Pass: SPM [32], RPNs [38]
    - Deep Feature Selection (Section 3.1.2)
      - Fully-connected Layer: Layer Concatenation [49]
      - Convolutional Layer: SPoC [7], CroW [10]
    - Feature Fusion Strategy (Section 3.1.3)
      - Layer-level Fusion: MoF [50], MOP [26]
      - Model-level Fusion: ConvNet fusion [46]
  - **Deep Feature Enhancement (Section 3.2)**
    - Feature Aggregation (Section 3.2.1)
    - Feature Embedding (Section 3.2.2)
    - Attention Mechanism (Section 3.2.3)
      - Non-parametric: SPoC [7], TSWVF [51]
      - Parametric: DeepFixNet+SAM [52, 53]
    - Deep Hash Embedding (Section 3.2.4)
      - Supervised Hashing: Metric Learning [35, 54]
      - Unsupervised Hashing: KNN [55], k-means [56]
- **Retrieval via Learning DCNN Representations (Section 4)**
  - Supervised Fine-tuning (Section 4.1)
    - Classification-based Fine-tuning (Section 4.1.1)
    - Verification-based Fine-tuning (Section 4.1.2)
      - Transformation Matrix: Non-metric [36]
      - Siamese Networks: [37, 57]
      - Triplet Networks: [37, 57]
  - Unsupervised Fine-tuning (Section 4.2)
    - Manifold Learning Sample Mining: Diffusion Net [43]
    - AutoEncoder-based Fine-tuning: KNN [58], GANs [45]

Fig. 4: This survey is organized around four key aspects in deep image retrieval, shown in boldface.

from images [15]. There is significant interest in learning deep features which are higher-level and semantic-aware, to better preserve the similarities of images [15]. In the past few years, various feature learning strategies, including feature fusion [26, 50] and feature enhancement methods [7, 28, 51] have been introduced into image retrieval. However, this area remains a major challenge and continues to require significant effort.

2. Improving retrieval scalability: The tremendous numbers and diversity of datasets lead to domain shifts for which existing retrieval systems may not be suited [8]. Currently available deep networks are initially trained for image classification tasks, which leads to a challenge in extracting features, since such features are less scalable and perform comparatively poorly on the target retrieval datasets, so network fine-tuning on retrieval datasets is crucial for mitigating this challenge. The current dilemma is that the increase in retrieval datasets raises the difficulty of annotation, making the development of unsupervised fine-tuning methods a priority.

3. Balancing retrieval accuracy and efficiency: Deep features are usually high dimensional, which therefore contain more semantic information to support higher accuracy, yet this higher accuracy is often at the expense of efficiency. Feature enhancement methods, like hash learning, are one approach to tackling this issue [18, 34], however hashing learning needs to carefully consider the loss function design, such as quantization loss [9, 11], to obtain optimal codes for high retrieval accuracy.

2 Popular Backbone DCNN Architectures

The hierarchical structure and extensive parameterization of DCNNs has led to their success in a remarkable diversity of computer vision tasks. For image retrieval, there are four models which predominantly serve as the networks for feature extraction, including AlexNet [20], VGG [46], GoogLeNet [47], and ResNet [21].

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 stages.

Inspired by AlexNet, VGGNet has two widely used versions: VGG-16 and VGG-19, including 13 convolutional layers and 16 convolutional layers, respectively, but where all of the convolutional filters are small (local), 3×3 in size. VGGNet is trained in a multi-scale manner where training images are cropped and re-scaled, which improves the feature invariance for the retrieval task.

Compared to AlexNet and VGGNet, GoogLeNet is deeper and wider but has fewer parameters within its 22 layers, leading to higher learning efficiency. GoogLeNet has repeatedly-used inception modules, each of which consists of four branches where 5×5, 3×3, and 1×1 filter sizes are used. Four branches are concatenated spatially to obtain the final features for each module. It has been demonstrated that deeper architectures are beneficial for learning higher-level abstract features to mitigate the semantic gap [15].

Finally, ResNet is developed by adding more convolutional layers to extract more abstract features. Skip connections are added between convolutional layers to address the notorious vanishing gradient problem when training this network.

DCNN architectures have developed significantly during the past few years, for which we refer readers to recent surveys [67, 68]. This paper focuses on introducing relevant techniques including feature fusion, feature enhancement, and network fine-tuning, based on popular DCNN backbones for performing image retrieval.

3 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 the off-the-shelf feature detectors for the image retrieval task of interest in 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 with this approach, such that the deep features may not outperform classical hand-crafted features. Most fundamentally, there is a model-transfer or domain-shift issue between tasks [8, 27, 69], meaning that models trained for classification do not necessarily possess features well suited to image retrieval. In particular, a classification decision can be made as long as the features remain within the classification boundaries, therefore the layers from such models may show insufficient capacity for retrieval tasks where feature matching

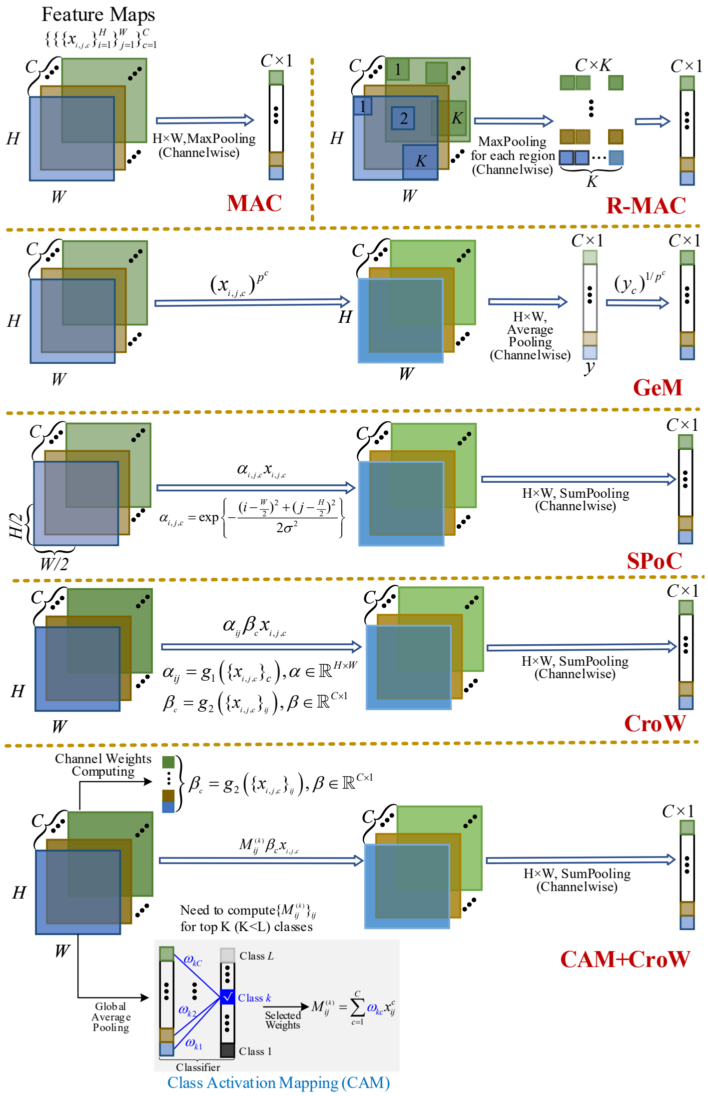

Fig. 5: Representative methods in single feedforward frameworks, focusing on convolutional feature maps: MAC [48], R-MAC [28], GeM pooling [42], SPoC with the Gaussian weighting scheme [7], CroW [10], and CAM+CroW [29]. Note that $g_1()$ and $g_2()$ represent spatial-wise and channel-wise weighting functions, respectively.

is more important than final classification probabilities. This section will survey the strategies which have been developed to improve the quality of feature representations, particularly based on feature extraction / fusion (Section 3.1) and feature enhancement (Section 3.2).

3.1 Deep Feature Extraction

3.1.1 Network Feedforward Scheme

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 [70].

The fully-connected layer has a global receptive field so that it is able to produce more semantic-aware features [13]. After normalization and dimensionality reduction, these features are used for direct similarity measurement without further feature processing and admitting efficient search strategies [25, 26, 34].

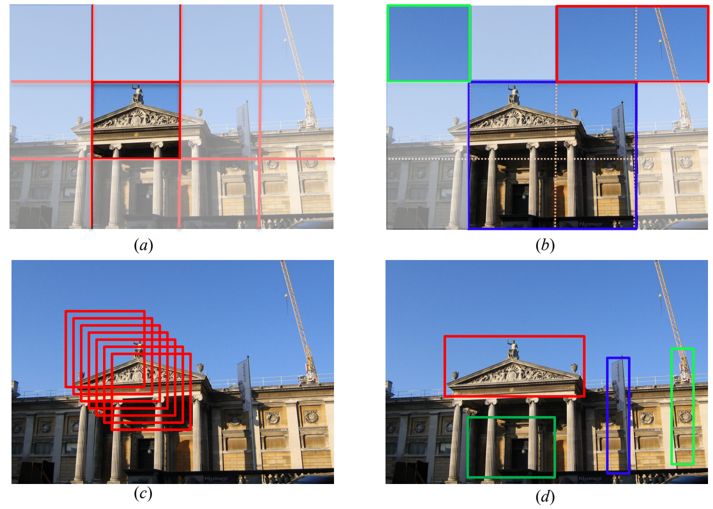

Fig. 6: Image patch generation schemes: (a) Rigid grid; (b) Spatial pyramid modeling (SPM) splits an image into different scales and positions (blue, green and red boxes); (c) Dense patch sampling, where a fixed-size sliding window samples the image; (d) Region proposals (RP), in which the specific object or instance is extracted as region proposals.

Using the fully-connected layer may result in insufficient performance since it lacks geometric invariance and spatial information, so the last convolutional layer can be examined instead. The research foci associated with the use of convolutional features is to improve their discrimination, where representative strategies are shown in Figure 5. One direction is to treat regions in feature maps as different sub-vectors, thus combinations of different sub-vectors of all feature maps are used to represent the input image. For instance, Gordo *et al.* [38] apply regional maximum activation of convolutions (R-MAC) [28] to obtain relevant regions on each feature map, which filters out some irrelevant (background) information and is beneficial for extracting instance-relevant features. Inspired by R-MAC, Li *et al.* [59] propose a non-linear feature embedding method for visual object retrieval and achieve remarkable performance improvements compared to the state of the art.

b. Multiple Feedforward Pass Methods.

Compared to single-pass schemes, multiple pass methods are more time-consuming [8] because several patches are generated from an input image and are both fed into the network before being encoded as a final global feature.

Multiple-pass strategies can lead to higher retrieval accuracy since representations are produced from two stages: patch detection and patch description. Multi-scale image patches are obtained using sliding windows [26, 71], random cropping [25, 57], and spatial pyramid model (SPM) [32], as illustrated in Figure 6. For example, Xu *et al.* [72] randomly sample windows within an image at different scales and positions, then "edgeness" scores are calculated to represent the edge density within the windows.

These patch detection methods lack retrieval efficiency for large-scale datasets since irrelevant patches are also fed into deep networks, therefore it is necessary to analyze image patches [28]. As an example, Cao *et al.* [73] propose to merge image patches into larger regions with different hyper-parameters, then the hyper-parameter selection is viewed as an optimization problem under the target of maximizing the similarity between features of the query and the candidates.

Instead of generating multi-scale image patches randomly or densely, region proposal methods introduce a degree of purpose in processing image objects, which is more efficient and less memory demanding. Region proposals can be generated using unsupervised object detectors, such as selective search [74] and edge boxes [75]. Aside from using object detectors, Xie *et al*. [76] introduce a manual object detection method in which the proposal layers are defined according to the number of objects. Region proposals can also be learned using deep networks, such as region proposal networks (RPNs) [23, 38] and convolutional kernel networks (CKNs) [77], and then to apply these deep networks into end-to-end fine-tuning scenarios for learning similarity [78, 79].

3.1.2 Deep Feature Selection

a. Extracted from Fully-connected Layers

It is straightforward to select a fully-connected layer as a feature extractor [25, 26, 34, 49]. With PCA dimensionality reduction and normalization [25], the similarity of images is measured using Euclidean or cosine distances. Only the last fully-connected layer may limit the overall retrieval performance, Jun *et al*. [49] propose to concatenate features from multiple fully-connected layers, and Song *et al*. [78] indicate that making a direct connection between the first fully-connected layer and the last layer achieves coarse-to-fine improvements.

As noted, a fully-connected layer has a global receptive field in which each neuron has connections to all neurons of the previous layer. This property leads to two obvious limitations for image retrieval: a lack of spatial information and a lack of local geometric invariance [49].

For the first limitation, researchers focus on the inputs of networks, *i.e.*, using multiple feedforward passes [25]. Compared to taking as input the whole image, discriminative features from the image patches better retain spatial information.

For the second limitation, a lack of local geometric invariance affects the robustness to image transformations such as truncation and occlusion. To address these issues, several works introduce methods which leverage intermediate convolutional layers [7, 26, 48, 80].

b. Extracted from Convolutional Layers

Features from convolutional layers (usually the last layer) preserve more structural details which are especially beneficial for instance-level retrieval [48]. Usually, the robustness of convolutional features is improved after pooling because convolutional layers preserve more local structural information, such as edges and corners.

The neurons in a convolutional layer are connected only to a local region of the input feature maps and share parameters in a convolutional volume. The smaller receptive field ensures that the produced features are more robust to image transformations like truncation and occlusion [7].

A convolutional layer arranges the spatial information well and produces location-adaptive features [81, 82]. Various image retrieval methods use convolutional layers as local detectors [7, 28, 29, 48, 80, 82]. For instance, Razavian *et al*. [48] make the first attempt to perform spatial max pooling on the feature maps of an off-the-shelf DCNN model; Babenko *et al*. [7] propose sum-pooling convolutional features (SPoC) to obtain compact descriptor feature maps pre-processed with a Gaussian center prior (see Figure 5). Ng *et al*. [82] explore the correlations between activations at different locations on the feature maps, thus improving the final feature descriptor. Kulkarni *et al*. [83] use the BoW model to embed convolutional features separately. Yue *et al*. [80] replace BoW [63] with VLAD [64], and are the first to encode local features into VLAD features. This idea inspired another milestone work [39] where, for the first time, VLAD is used as a layer plugged into the last convolutional layer. The plugged-in layer is end-to-end trainable via back-propagation.

3.1.3 Feature Fusion Strategy

a. Layer-level Fusion

Fusing features from different layers aims at combining different feature properties within a feature extractor. It is possible to fuse multiple fully-connected layers in a deep network [49]: For instance, Yu *et al*. [84] explore different methods to fuse the activations from different fully-connected layers and introduce the best-performed P_i-fusion strategy to aggregate the features with different balancing weights, and Jun *et al*. [49] construct multiple fully-connected layers in parallel on the top of ResNet backbone, then concatenate the global features from these layers to obtain the combined global features.

Features from fully-connected layers contain high-level semantic information, but lack detailed structural information, while features from convolutional layers contain more structural information but suffer from background noise and semantic ambiguity [61]. Thus, global features and local features can complement each other when measuring semantic similarity and can, to some extent, guarantee retrieval performance.

Global features and local features can be concatenated directly [61, 85, 86]. Before concatenation, convolutional feature maps are filtered by sliding windows or region proposal nets. Pooling-based methods can be applied for feature fusion as well. For example, Li *et al*. [50] propose a Multi-layer Orderless Fusion (MOF) approach, which is inspired by Multi-layer Orderless Pooling (MOP) [26] for image retrieval. However local features can not play a decisive role in distinguishing subtle feature differences because global and local features are treated identically. For this limitation, Yu *et al*. [61] propose using a mapping function to take more advantage of local features in which they are used to refine the return ranking lists. In their work, the exponential mapping function is the key for tapping the complementary strengths of the convolutional layers and fully-connected layers. Similarly, Cao *et al*. [86] unify the global and local descriptors for two-stage image retrieval in which attentively selected local features are employed to refine the results obtained using global features.

It is worth introducing a fusion scheme to explore *which* layer combination is better for fusion given their differences of extracting features. For instance, Chatfield *et al*. [87] demonstrate that fusing convolutional layers and fully-connected layers outperforms the methods that fuse only convolutional layers. In the end, fusing two convolutional layers with one fully-connected layer achieves the best performance.

b. Model-level Fusion

It is possible to explore feature fusion on different models; such fusion focuses on model complementarity to achieve improved performance, categorized into two groups: *intra-model* and *inter-model*.

Generally, intra-model fusion suggests multiple deep models having similar or highly compatible structures, while inter-model fusion involves models with more differing structures. For instance, the widely-used dropout strategy in AlexNet [20] 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*. [46] introduce a ConvNet fusion strategy to improve the feature learning capacity of VGG where VGG-16 and VGG-19 are fused. This intra-model fusion strategy reduces the top-5 error by 2.7% in image classification compared to a single counterpart network. Similarly, Liu *et al*. [88] propose to mix different VGG variants to strengthen the feature learning for fine-grained vehicle retrieval. Ding *et al*. [89] propose a selective deep ensemble (SDE) framework to combine ResNet-26 and ResNet-50 improve the accuracy of fine-grained instance retrieval. To attend to different parts of the object in an image, Kim *et al*. [90] train an ensemble of three attention modules to learn features with different diversities. Each module is based on different Inception blocks in GoogLeNet.

Inter-model fusion is a way to bridge different features given the fact that different deep networks have different receptive fields [32, 53, 81, 91, 92, 93]. For instance, a two-stream attention network [53] is proposed to implement image retrieval where the mainstream network for semantic prediction is VGG-16 while the auxiliary stream network for predicting attention maps is DeepFixNet [94]. Similarly, considering the importance and necessity of inter-model fusion to bridge the gap between mid-level and high-level features, Liu *et al*. [32] and Zheng *et al*. [81] propose to use VGG-19 and AlexNet to learn combined features, while Ozaki *et al*. [92] make an ensemble to concatenate descriptors from six different models to boost retrieval performance. To illustrate the effect of different parameter choices within the model ensemble, Xuan *et al*. [93] combine ResNet and Inception V1 [47] for retrieval, concentrating on the embedding size and number of embedded features.

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 concatenate all types of features from the candidate models and then learning a metric based on the concatenated features [53], 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 uniformly combine these metrics for final retrieval ranking [33], which is a kind of "*late fusion*" strategy.

Discussion. Layer-level fusion and model-level fusion are conditioned on the fact that the involved 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 for answering this question based on off-the-shelf deep models. For example, Xuan *et al*. [93] illustrate the effect of combining different numbers of features and different sizes within the ensemble. Chen *et al*. [95] analyze the performance of embedded features from image classification and object detection models with respect to image retrieval. They study the discrimination of feature embeddings of different off-the-shelf models which, to some extent, implicitly guides the model selection when conducting the inter-model level fusion for feature learning.

3.2 Deep Feature Enhancement

3.2.1 Feature Aggregation

Feature enhancement methods aggregate or embed features to improve the discrimination of deep features. In terms of feature aggregation, sum/average pooling and max pooling are two commonly used methods applied on convolutional feature maps. In particular, sum/average pooling is less discriminative, because it takes into account all activated outputs from a convolutional layer, as a result it weakens the effect of highly activated features [30]. On the contrary, max pooling is particularly well suited for sparse features that have a low probability of being active. Max pooling may be inferior to sum/average pooling if the output feature maps are no longer sparse [96].

Convolutional feature maps can be directly aggregated to produce global features by spatial pooling. For example, Razavian *et al*. [48, 71] apply max pooling on the convolutional features for retrieval. Babenko *et al*. [7] leverage sum pooling with a Gaussian weighting scheme to aggregate convolutional features (*i.e.* SPoC). Note that this operation usually is followed by L2 normalization and PCA dimensionality reduction.

As an alternative to the holistic approach, it is also possible to pool some regions in a feature map [7, 48, 81], such as done by R-MAC [28], where max pooling is used to aggregate some regions on feature maps. Also, it is shown that the pooling strategy used in the last convolutional layer usually yields superior accuracy over other shallower convolutional layers and even fully-connected layers [81].

3.2.2 Feature Embedding

Apart from direct pooling or regional pooling, it is possible to embed the convolutional feature maps into a high dimensional space to obtain compact features. The commonly used embedding methods include BoW, VLAD, and FV. The dimensionality of these embedded features can be reduced using PCA. Note that BoW and VLAD can be extended by using other metrics, such as Hamming distance [97]. Here we briefly describe the principle of the embedding methods for the case of Euclidean distance metric.

BoW [63] is a widely adopted encoding method in which semantic similarity is measured using a standard distance metric. BoW encoding takes advantage of sparse representations, which is beneficial for fast retrieval on large-scale datasets. Let $\vec{X} = \{\vec{x}_1, \vec{x}_2, ..., \vec{x}_T\}$ be a set of local features, each of which has dimensionality D. BoW requires a pre-defined codebook $\vec{C} = \{\vec{c}_1, \vec{c}_2, ..., \vec{c}_K\}$ with K centroids to cluster these local descriptors, and maps each descriptor $\vec{x}_t$ to the nearest word $\vec{c}_k$. For each centroid $\vec{c}_k$, we count and normalize the number of occurrences by

$$g(\vec{c}_k) = \frac{1}{T}\sum_{t=1}^{T}\phi(\vec{x}_t, \vec{c}_k) \tag{1}$$

$$\phi(\vec{x}_t, \vec{c}_k) = \begin{cases} 1 & \text{if } \vec{c}_k \text{ is the closest codeword for } \vec{x}_t \\ 0 & \text{otherwise} \end{cases} \tag{2}$$

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

$$G_{_{BoW}}(\vec{X}) = \left[\ g(\vec{c}_1), \cdots, g(\vec{c}_k), \cdots, g(\vec{c}_K)\ \right]^{\top} \tag{3}$$

BoW representation is the histogram of the number of local descriptors assigned to each visual word, so that its dimension is equal to the number of centroids. This method is simple to implement to encode local descriptors, such as convolutional feature maps [50, 70, 83]. However, the embedded vectors are high dimensional and sparse, which are not well suited to large-scale datasets.

VLAD [64] stores the sum of residuals for each visual word. Specifically, similar to BoW, it generates K visual word centroids, then each feature $\vec{x}_t$ is assigned to its nearest visual centroid $\vec{c}_k$ and computes the difference $(\vec{x}_t - \vec{c}_k)$:

$$g(\vec{c}_k) = \frac{1}{T}\sum_{t=1}^{T}\phi(\vec{x}_t, \vec{c}_k)(\vec{x}_t - \vec{c}_k) \tag{4}$$

where $\phi(\vec{x}_t, \vec{c}_k)$ as defined in (2). Finally, the VLAD representation is stacked by the residuals for all centroids, with dimension $(D \times K)$, *i.e.*,

$$G_{VLAD}(\vec{X}) = [\ \cdots, g(\vec{c}_k)^\top, \cdots\ ]^\top. \tag{5}$$

VLAD captures first order feature statistics, *i.e.*, $(\vec{x}_t - \vec{c}_k)$. Similar to BoW, the performance of VLAD is affected by the number of clusters, thereby larger centroids produce larger vectors that are harder to index. For image retrieval, for the first time, Ng *et al.* [80] embed the feature maps from the last convolutional layer into VLAD representations, however VLAD has higher effectiveness than BoW.

The FV method [65] extends BoW by encoding the first and second order statistics continuously. FV clusters the set of local descriptors by a Gaussian Mixture Model (GMM), with K components, to generate a dictionary $C = \{\mu_k; \Sigma_k; w_k\}_{k=1}^{K}$, where w_k, μ_k, Σ_k denote the weight, mean vector, and covariance matrix of the k-th Gaussian component, respectively [98]. The covariance can be simplified by keeping only its diagonal elements, *i.e.*, $\sigma_k = \sqrt{diag(\Sigma_k)}$. For each local feature x_t, a GMM is given by

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

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

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

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

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

The FV representation defines a kernel from a generative process and captures more statistics than BoW and VLAD. FV vectors do not increase computational costs significantly but require more memory due to the larger feature vector sizes. Applying FV without memory controls may lead to suboptimal performance [99].

Discussion. Traditionally, sum pooling and max pooling are directly plugged into deep networks and used end-to-end, whereas the embedding methods, including BoW, VLAD, and FV, are initially trained separately with pre-defined vocabularies [32, 104]. For these three methods, their properties should be considered when embedding deep features. For instance, BoW and VLAD are computed in the rigid Euclidean space where the performance is closely related to the number of centroids. The FV embedding method can capture higher order statistics than BoW or VLAD, thus the FV embedding improves the effectiveness of feature enhancement at the expense of a higher memory cost. When any one of these methods is used, it is necessary to integrate them in an end-to-end manner so as to guarantee training and testing efficiency. For example, the VLAD method is integrated into deep networks where each spatial column feature is used to construct clusters via k-means [80]. This idea led to a follow-up approach, NetVLAD [39], where deep networks are fine-tuned with the VLAD vector. The FV embedding method is also explored and combined with deep networks for retrieval tasks [37, 105].

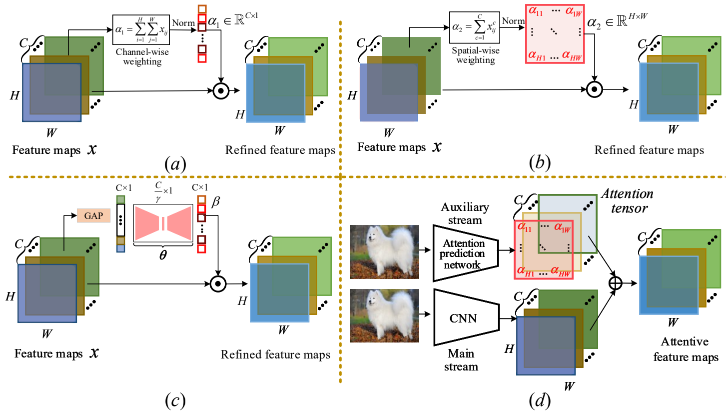

Fig. 7: Four attention mechanisms are shown, divided into two categories. (a)-(b) Non-parametric mechanisms: The attention is based on convolutional feature maps. Channel-wise attention in (a) produces a C-dimensional importance vector α_1 [10, 31, 100]. Similarly, spatial-wise attention in (b) computes a 2-dimensional attention map α_2 [10, 29, 62, 82]. (c)-(d) Parametric mechanisms: The attention weights β are provided by a sub-network with trainable parameters (*e.g.* θ in (c)) [101, 102]. Likewise, some off-the-shelf models [94, 103] can be used to predict the attention maps from the input image directly.

3.2.3 Attention Mechanisms

The core idea of attention mechanisms is to highlight the most relevant features and to avoid the influence of irrelevant activations, realized by computing an attention map. Approaches to obtain attention maps can be categorized into two groups: non-parametric and parametric-based, as shown in Figure 7, 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. The corresponding attention maps can be obtained by channel-wise or spatial sum-pooling, as in Figure 7(a,b). For the spatial-wise pooling of Figure 7(b), Babenko *et al.* [7] apply a Gaussian center prior scheme to spatially weight the activations of a convolutional layer prior to aggregation. Kalantidis *et al.* [10] propose a more effective CroW method to weight and pool feature maps. These spatial-wise methods only concentrate on weighting activations at different spatial locations, without considering the relations between these activations. Instead, Ng *et al.* [82] explore the correlations among activations at different spatial locations on the convolutional feature maps. In addition to spatial-wise attention mechanisms, channel-wise weighting methods of Figure 7(a) are also popular non-parametric attention mechanisms. Xu *et al.* [31] rank the weighted feature maps to build the "probabilistic proposals" to further select regional features. Similarly, Jimenez *et al*. [29] combine CroW and R-MAC to propose Classes Activation Maps (CAM) to weight feature maps for each class. Qi *et al*. [51] introduce Truncated Spatial Weighted FV (TSWVF) to enhance the representation of Fisher Vector.

Attention maps can be learned from deep networks, as shown in Figure 7(c,d), where the input can be either image patches or feature maps from the previous convolutional layer. The parametric attention methods are more adaptive and are commonly used in supervised metric learning. For example, Li *et al*. [101] propose stacked fully-connected layers to learn an attention model for multi-scale image patches. Similarly, Noh [102] designs 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*. [106] employ a pre-trained ResNet101 to train a context-aware attention network using multi-scale feature maps.

Instead of using feature maps as inputs, a whole image can be used to learn feature importance, for which specific networks are needed. For example, Mohedano [52] explore different saliency models, including DeepFixNet [94] and Saliency Attentive Model (SAM)[103], to learn salient regions for input images. Similarly, Yang *et al*. [53] introduce a two-stream network for image retrieval in which the auxiliary stream, DeepFixNet, is used specifically for predicting attention maps.

In a nutshell, attention mechanisms offer deep networks the capacity to highlight the most important regions of a given image, widely used in computer vision. For image retrieval specifically, attention mechanisms can be combined with supervised metric learning [82, 90, 107].

3.2.4 Deep Hash 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. Hashing algorithms have been widely used for large-scale image search due to their computational and storage efficiency [18, 108].

Hash functions can be plugged as a layer into deep networks, so that hash codes can be trained and optimized with deep networks simultaneously. During hash function training, the hash codes of originally similar objects are embedded as close as possible, and the hash codes of dissimilar objects are as separated as possible. A hash function may be formulated as

$$b_{nk} = g(x_n) = g(f(x_n; \boldsymbol{\theta})) \quad k = 1, \ldots, K \tag{9}$$

where binary hash codes $b_n \in \{+1, -1\}^K$ are generated by a hash function $g(\cdot)$. Because hash codes are non-differentiable their optimization is difficult, so $g(\cdot)$ can be relaxed to be differentiable by using *tanh* or *sigmoid* functions [18].

When binarizing real-valued features, it is crucial (1) to preserve image similarity and (2) to improve hash code quality [18]. 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 the real-valued features and corresponding hash codes, for which a variety of strategies have been adopted.

The design of the objective loss function can significantly influence similarity preservation, which includes both supervised and unsupervised approaches. With the class label available, many loss functions are designed to learn hash codes in a Hamming space. As a straightforward method, one can optimize the difference between matrices computed from the binary codes and their supervision labels [109]. Other studies regularize hash codes with a center vector, for instance a class-specific center loss is devised to encourage hash codes of images to be close to the corresponding centers, reducing the intra-class variations [108]. Similarly, Kang *et al*. [110] introduce a max-margin t-distribution loss which concentrates more similar data into a Hamming ball centered at the query term, such that a reduced penalization is applied to data points within the ball, a method which improves the robustness of hash codes when the supervision labels may be inaccurate. Moreover metric learning, including Siamese loss [54], triplet loss [35, 111, 112], and adversarial learning [111, 113], is used to retain semantic similarity where only dissimilar pairs keep their distance within a margin. In terms of unsupervised hashing learning, it is essential to capture some relevance among samples, which has been accomplished by using Bayes classifiers [114], KNN graphs [55, 58, 115], k-means algorithms [56], and network structures such as AutoEncoders [116, 117, 118] and generative adversarial networks [45, 55, 119, 120].

Separate from the loss function, it is also important to design deep network frameworks for learning. For instance, Long *et al*. [112] apply unshared-weight CNNs on two datasets where a triplet loss and an adversarial loss are utilized to address the domain shifts. Considering the lack of label information, Cao *et al*. [113] present coined Pair Conditional WGAN, a new extension of Wasserstein generative adversarial networks (WGAN), to generate more samples conditioned on the similarity information.

b. Improving Hash Function Quality

Improving hash code quality aims at making the binary codes uniformly distributed, that is, maximally filling and using the hash code space, normally on the basis of bit uncorrelation and bit balance [18]. Bit uncorrelation implies that different bits have little redundancy of information, so that a given set of bits can aggregate more information within a given code length. Bit uncorrelation can be encouraged via regularization terms such as orthogonality [121] and mutual information [122]. Bit balance means that each bit should have a 50% chance of being +1 or -1, thereby maximizing code variance and information [18].

4 Retrieval via Learning DCNN Representations

In Section 3, we presented feature fusion and enhancement strategies for which off-the-shelf DCNNs only serve as extractors to obtain features. However, in most cases, deep features may not be sufficient for high accuracy retrieval, even with the strategies which were discussed. In order for models to have higher scalability and to be more effective in retrieval tasks, one important solution is in fine-tuning, to update the pre-stored parameters [13, 27, 66], a common practice in deep image retrieval. However fine-tuning does not contradict or render irrelevant the feature processing methods of Section 3; indeed, those strategies are complementary and can be incorporated as part of network fine tuning.

This section focuses on supervised and unsupervised fine-tuning methods for the updating of network parameters.

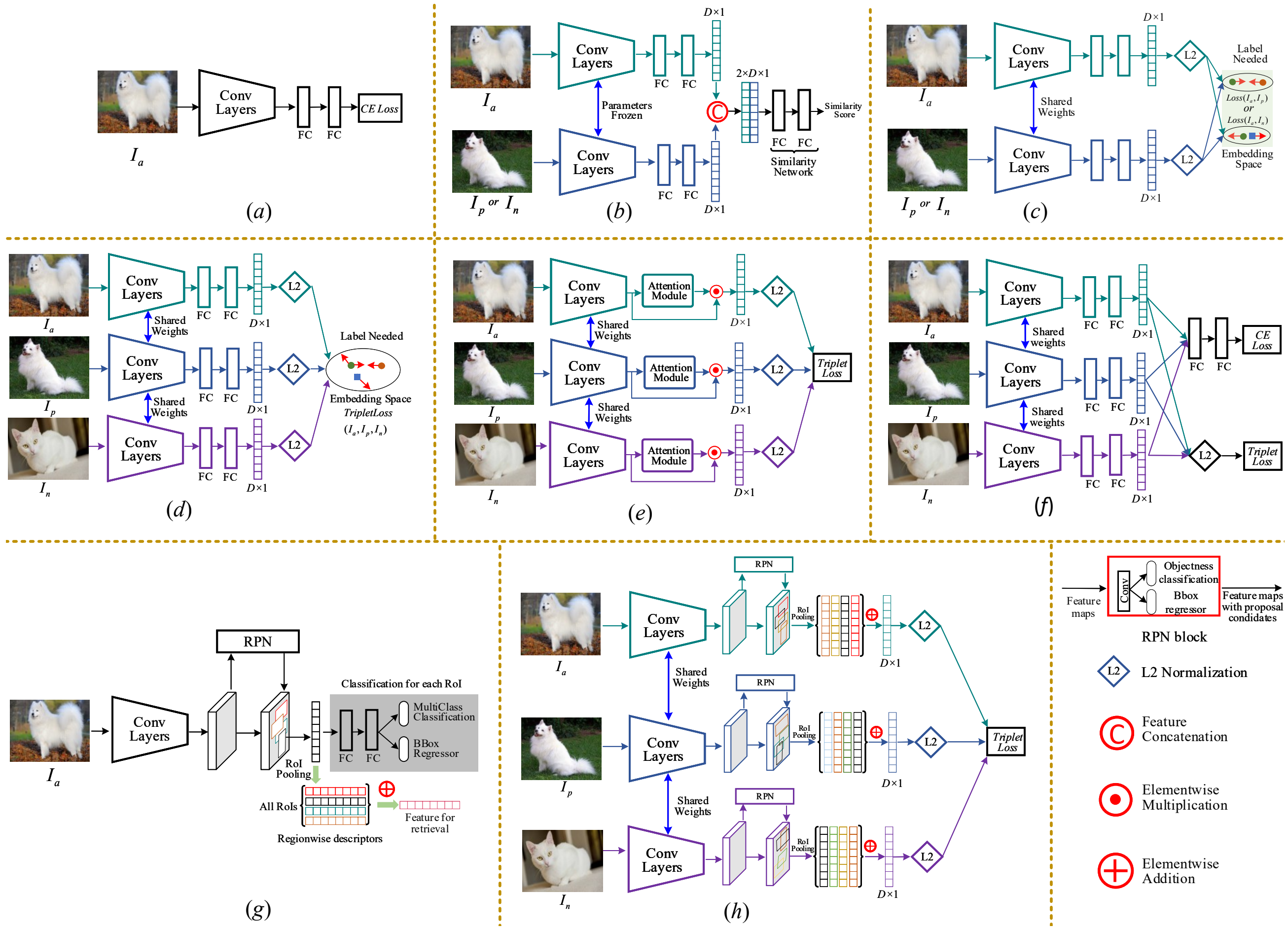

Fig. 8: Schemes of supervised metric learning. Anchor, positive, and negative images are indicated by x_a, x_p, x_n, respectively. (a) classification-based; (b) using a linear or non-linear transformation matrix for learning the similarity of image pairs; (c) Siamese networks; (d) region proposal networks (RPNs) to locate the RoI and highlight specific regions or instances; (e) triplet loss for fine-tuning; (f) inserting the RPNs of (d) into DCNNs, such that the RPNs extract regions or instances at the convolutional layer; (g) an attention block into DCNNs to highlight regions; (h) combining classification-based and verification-based loss for fine-tuning.

4.1 Supervised Fine-tuning

4.1.1 *Classification-based Fine-tuning*

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 subsequently fine-tuned, as depicted in Figure 8(a), by optimizing its parameters on the basis of a cross entropy loss L_{CE}:

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

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 [34], in which AlexNet is re-trained on the Landmarks dataset with 672 pre-defined categories. 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 in deep image retrieval. However, there still exists room to improve in terms of classification-based supervised learning. On the one hand, the fine-tuned networks are quite robust to inter-class variability, but may have some difficulties in learning discriminative intra-class variability to distinguish particular objects. On the other hand, class label annotation is time-consuming and labor-intensive for some practical applications. To this end, verification-based fine-tuning methods are combined with classification methods to further improve network capacity.

4.1.2 *Verification-based Fine-tuning*

With affinity information indicating similar and dissimilar image pairs, verification-based fine-tuning methods learn an optimal metric which minimizes or maximizes the distance of pairs to validate and maintain their similarity. Compared to classification-based learning, verification-based learning focuses on both inter-class and intra-class samples.

Verification-based learning involves two types of information [13]:

1) A pair-wise constraint, corresponding to a Siamese network as in Figure 8(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 8(e), in which anchor images are paired with both similar and dissimilar samples [13].

These verification-based learning methods are categorized into global supervised approaches (Figure 8(e)) and local supervised approaches (Figure 8(d)), where the former learn a metric on global features by satisfying all constraints, whereas the latter 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 deep 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 the input samples can be implemented by optimizing the weights of a linear transformation matrix [36]. It transforms the concatenated feature pairs into a common latent space using a transformation matrix $\boldsymbol{W}\in\mathbb{R}^{d\times d}$, where d is the feature dimension. Then, the similarity score of these pairs are predicted via a sub-network $S_W(x_i, x_j) = f_W(f(x_i;\boldsymbol{\theta}) \cup \ f(x_j;\boldsymbol{\theta}); \boldsymbol{W})$ [36, 134]. 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:

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

where $sim(x_i, x_j)$ can be the cosine function for guiding training $\boldsymbol{W}$. m is defined as a margin and remains equal in following loss functions. By optimizing the regression loss and updating the transformation matrix $\boldsymbol{W}$, 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 8(b) where the weights of the two DCNNs are not necessarily shared.

b. Fine-tuning with Siamese Networks.

Siamese networks are important options to implement metric learning for fine-tuning, as shown in Figure 8(c). It is a structure composed of two branches that share the same weights across the layers. Siamese networks are trained on paired data consisting of an image pair (x_i, x_j) according to $S(x_i, x_j) \in \{0, 1\}$. A Siamese loss function is formulated as:

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

A standard Siamese network and a Siamese loss are employed to learn the similarity between the semantically relevant samples under different scenarios. For example, Simo *et al.* [135] introduce a Siamese network to learn the similarity between paired image patches, which focuses more on the specific regions within an image. Ong *et al.* [37] leverage the Siamese network to learn image features which are then fed into the Fisher Vector model for further encoding. In addition, Siamese network can also be applied for hashing learning in which the Euclidean distance formulation $D(\cdot)$ in Eq. 13 is replaced by Hamming distance [54].

c. Fine-tuning with Triplet Networks.

Triplet networks [134] optimize the similar and dissimilar pairs simultaneously. As shown in Figure 8(e), the plain triplet networks adopt a ranking loss for tr-training:

$$L_{Triplet}(x_a, x_p, x_n) = \max(0, m + D(x_a, x_p) - D(x_a, x_n))) \tag{14}$$

This loss 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. The triplet loss is used to learn fine-grained image features [57, 91, 136], and used for constraining hash codes learning [35, 111, 112].

To focus on specific regions or objects, local supervised metric learning has been explored [43, 79, 137, 138]. In these methods, some regions or objects are extracted using region proposal networks (RPNs) [23] which subsequently can be plugged in deep networks and trained in an end-to-end manner. As shown in Figure 8(d), Faster R-CNN [23] is Fine-tuned for instance search [79]. 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 the spatial ranking for instance retrieval.

RPNs [23] enable deep models to learn regional features for particular instances or objects [38, 138]. RPNs used in the triplet formulation are shown in Figure 8(f). For training, besides the triplet loss, regression loss (PRNs loss) is used to minimize the regressed bounding box according to ground-truth region of interest. In some cases, jointly training an RPNs loss and triplet loss leads to unstable results. This is addressed by the authors [38] first training a CNN to produce R-MAC using a rigid grid. Afterwards, the parameters in convolutional layers are fixed and RPNs are trained to replace the rigid grid.

Furthermore, attention mechanism is also combined with metric learning [107, 137], see Figure 8(g). The attention module usually is end-to-end trainable and takes as input the convolutional feature maps. For instance, Song *et al.* [137] introduce a convolutional layer as attention layers to explore spatial-semantic information, the highlighted regions in images significantly improve the discrimination for inter-class and intra-class features for image retrieval.

Recently, jointly optimizing the triplet loss and classification loss function is being studied [49, 85]. This is depicted in Figure 8(h). Fine-tuned models that only use triplet constraint may be inferior classification accuracy for similar instances [85]. The classification loss does not predict the intra-class similarity, but locate the relevant images at different levels. Given these considerations, it is natural to combine and optimize triplet constraint and classification loss jointly [49]. The overall joint function is formulated as:

$$L_{Joint} = \alpha \cdot L_{Triplet}(x_{i,a}, x_{i,p}, x_{i,n}) + \beta \cdot L_{CE}(\hat{p}_i, y_i) \tag{15}$$

where cross-entropy loss (CE loss) L_{CE} is defined as in Eq. (10) and triplet loss $L_{Triplet}$ is equal to Eq. (14). α and β are the trade-off hyper-parameters to tune the two loss functions.

An implicit drawback of Siamese loss in Eq. 13 is that it may penalize similar image pairs even the margin between these pairs is at a small or zero distance, which may degrade the performance [139]. The constraint is too strong and unbalanced. Meanwhile, it is hard to map the features of similar pairs

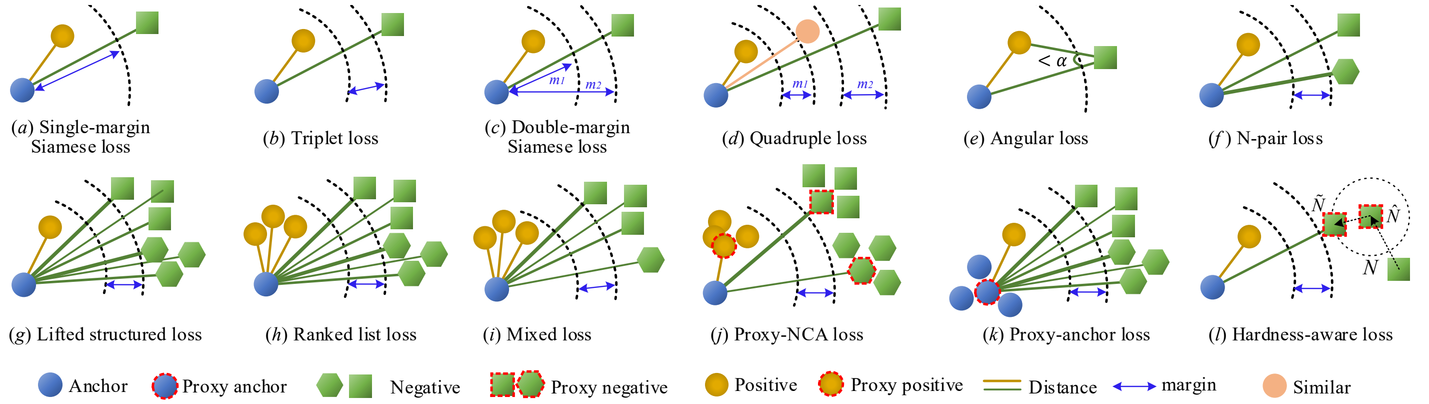

Fig. 9: Illustrations of sample mining strategies in metric learning. Here, we take 3 classes for illustration, where different shapes indicate different classes. Loss term that includes m_1 and m_2 denotes this loss involves two margins. Multiple pairs are considered in some loss terms and assigned with distinct weights during training, indicated by different line width. (a)-(c) have been introduced in the text. (d) Quadruple loss [125]: a sample similar to the anchor is used to construct a double margin. (e) Angular loss [126]: the angle at the negative of triple triangles is computed to obtain higher order geometric constraints. (f) N-pair loss [127]: it identifies a positive sample from $N-1$ negative samples of N-1 classes. (g) Lift structured loss [128]: it considers the structure relationships of three positive and three negative samples. (h) Ranked list loss [129]: it considers all samples to explore intrinsic structured information. (i) Mixed loss [130]: it captures three positive and three negative samples which are initially closely distributed, another anchor-negative pair initially lies very close to the anchor. (j) Proxy-NCA loss [131]: it computes proxy positive and negative samples for each class, and are trained with a true anchor sample. (k) Proxy-anchor loss [132]: the anchor sample is represented by a proxy. (l) Hardness-aware loss [133]: the synthetic negative is mapped from an existing hard negative, their hard levels are manipulated adaptively within a certain range.

to the same point when images contain complex contents or scenes. To tackle this limitation, Cao *et al*. [140] adopt a double-margin Siamese loss [139] to relax the penalty for similar pairs. To be specific, the threshold between the similar pairs is set to a margin m_1 instead of being zero. In this case, the original single-margin Siamese loss is re-formulated as following:

$$L(x_i, x_j) = \frac{1}{2}S(x_i, x_j)\max(0, D(x_i, x_j) - m_1) + \frac{1}{2}(1 - S(x_i, x_j))\max(0, m_2 - D(x_i, x_j)) \tag{16}$$

where $m_1>0$ and $m_2>0$ are the margins affecting the similar and dissimilar pairs, respectively. Therefore, the double margin Siamese loss only applies a contrastive force where 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 [139].

Discussion. Most verification-based supervised learning methods rely on the basic Siamese or triplet networks. The follow-up studies are focusing on exploring methods to further improve their capacities for robust feature similarity estimation. Generally, the network structure, loss function, and sample selection are important factors for the success of verification-based methods [141].

For a loss function, various loss functions have been proposed recently [125, 127, 128, 129, 131]. Some of the loss functions use more samples or additional constraints. For example, Chen *et al*. [125] incorporate Quadruple samples for constraining relationships between anchor, positive, negative, and similar images. The N-pair loss [127] and lifted structure loss [128] even define constraints on all images and employ the structural information of samples in a mini-batch.

The sampling strategy greatly affects the feature learning and training convergence rate. To date, various sampling strategies such as clustering have been introduced. For comparison, we illustrate 12 sampling strategies in Figure 9. Aside from sampling within a mini-batch, other work explore to mine samples outside a mini-batch even from the whole dataset. This may be beneficial for stabilizing optimization due to a larger data diversity and richer training information. For example, Wang *et al*. [142] propose a cross-batch memory (XBM) mechanism that memorizes the embedding of past iterations, allowing the model to collect sufficient hard negative pairs across multiple mini-batches. Harwood *et al*. [143] provide a framework named smart mining to collect hard samples from the entire training set. It is reasonable to achieve better performance when more samples are used to fine-tune network. However, the possible additional computational cost during training is a core issue to be addressed.

Recently, directly optimizing the average precision (AP) metric using the listwise AP loss [144] is a way to consider a large number of image simultaneously. Training with this loss has been demonstrated to improve retrieval performance [144, 145, 146]. However, Average precision, as a metric, is normally non-differentiable and non-smooth. To directly optimize the AP loss, the AP metric needs to be relaxed by using methods such as soft-binning approximation [144, 145], sigmoid function [146].

4.2 Unsupervised Fine-tuning

Supervised network fine-tuning becomes infeasible when there is not enough supervisory information because these information is costly to assemble or sometimes unavailable. Given these limitations, unsupervised fine-tuning methods for image retrieval are quite necessary but less studied [147].

For unsupervised learning, one direction is to mine some relevances among features to obtain ranking information. Another one is to devise novel unsupervised frameworks (*e.g.* AutoEncoders). To this end, we categorize the unsupervised fine-tuning methods into manifold learning-based and AutoEncoder-based methods.

4.2.1 *Mining Samples with Manifold Learning*

Manifold learning focuses on capturing intrinsic correlations on the manifold structure to mine some revelances. We show this process in Figure 10. Initial similarities between the original extracted features are used for constructing an affinity matrix. Then, the values in this matrix are re-evaluated and updated using manifold learning [148]. According to the manifold similarity in the updated affinity matrix, positive and hard negative samples are selected for metric learning using pair loss [43, 149], triplet loss [150, 151], or N-pair loss [147], *etc*. Note that this is different from the aforementioned methods for verification-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 relations from the manifold of deep features. Generally, there are two steps included [148]. First, the affinity matrix (see Figure 10) 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 [44, 149, 150, 151]. These two steps are known as the diffusion process. Recently, some new similarity diffusion methods are proposed like the regularized diffusion process (RDP) [152] and the regional diffusion mechanism [149]. For more details, we refer to the survey [148] which lists 72 variants of diffusion methods.

Most existing algorithms follow a similar principle (*i.e.* the random walk algorithm [148]). The differences lie in three aspects: similarity initialization, transition matrix definition, and iteration scheme. For the first aspect, the similarity initialization in an affinity matrix affects the subsequent KNN graph construction. Usually, an inner product [44, 147] or Euclidean distance [41] is directly computed for the affinities. Also a Guassian kernel function can be used for initialization of an affinity matrix [148, 151]. Iscen *et al.* [149] consider regional similarity from image patches to build the affinity matrix. For the second aspect, the formation of transition matrix (*e.g.* row-stochastic matrix [148]) determines 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). Finally, an iteration scheme guarantees the values in affinity matrix are re-valuated and updated by the manifold similarity until some kinds of convergence are achieved. Most existing algorithms are iteration-based [148, 150], as illustrated in Figure 10.

Diffusion process algorithms are indispensable for unsupervised metric learning. Better image similarity is guaranteed when it is improved on the mentioned initialization methods (*e.g.* regional similarity [149] or high order information [41] for similarity initialization). However, the diffusion process requires more computations and complex search due to the iteration scheme [151]. This limitation cannot meet the efficiency requirements of image retrieval. For mitigating this, Nicolas *et al.* [147] apply the closed-form convergence solution of a random walk in each mini-batch to estimate the manifold similarities instead of running many iterations. Some studies replace the diffusion process on a KNN graph with a diffusion network [43], which is derived from graph convolution networks [153]. Their end-to-end framework allows efficient computation during the training and testing stage.

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 [150]. The selected samples are fed into deep networks to perform fine-tuning.

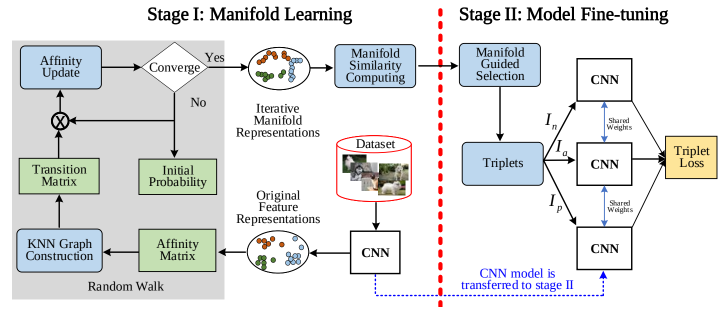

Fig. 10: Paradigm of manifold learning for unsupervised metric learning.

It is possible to explore the proximity information to make clusterings in the Euclidean space, subsequently the training set is split into different groups. For example, Tzelepi *et al.* [154] explore a fully unsupervised fine-tuning method by clustering. In this method, the KNN algorithm is used to compute the k nearest features, then the model is fine-tuned to minimize the squared distance between each query feature and its k nearest features. As another example, Radenovic *et al.* [40, 42] use Structure-from-Motion (SfM) for clustering to explore sample reconstructions to select images for triplet loss. Clustering methods depend on rigid Euclidean distance so that it is difficult to reveal the intrinsic relationship between objects.

4.2.2 AutoEncoder-based Frameworks

An AutoEncoder is a kind of neural network that aims to reconstruct its output as close as its input. In principle, an input image is encoded as features into a latent space, and these features are then reconstructed to the original input image using a decoder. Here, the encoder and decoder can be comprised of convolutional neural networks.

In an AutoEncoder, there exist different levels (*e.g.* pixel-level or instance-level) of reconstruction. These different resolutions of reconstruction affect the effectiveness of an AutoEncoder. For example, pixel-level reconstruction may degrade the learned features of an encoder by focusing on some trivial variations in a reconstructed image since a natural image usually contains a lot of trivial factors like location, color, and pose.

An AutoEncoder is an optional framework for supporting another methods. For example, unsupervised hash learning can be implemented by using AutoEncoders [45, 116, 117, 118]. Except for the reconstruction loss [45, 118], it is highly necessary to mine feature relevance to explore other objective functions. This is usually realized by using clustering algorithm [118] based on the fact that features from an off-the-shelf network initially contain rich semantic information to keep their semantic structure [55, 58, 114]. For example, Gu *et al.* [118] introduce a modified cross-entropy based on the k-means clustering algorithm where a deep model learns to cluster iteratively and yields binary codes while retaining the structures of the input data distributions. Zhou *et al.* [58] and Deng *et al.* [55] propose a self-taught hashing algorithm using a KNN graph construction to generate pseudo labels that are used to analyze and guide network training. Other techniques such as Bayes net are also used to predict sample similarity. For instance, Yang *et al.* [114] adopt a Bayes optimal classifier to assign semantic similarity labels to data pairs which have a higher similarity probability.

Fourthermore, an AutoEncoder can also be integrated into other frameworks, such as graph convolutional networks [153] and object detection model [155] to learn better binary latent variables. For eaxmple, Shen *et al*. [45] combine graph convolutional networks [153] to learn the hash codes from an AutoEncoder. In this method, the similarity matrix for graph learning is computed on the binary latent variables from the Encoder. Generative adversarial networks (GANs) are also explored in the unsupervised hashing framework [45, 55, 119, 120]. The adversarial loss in GANs is the classical objective to use. By optimizing this loss, the synthesized images generated from hash codes gradually keep semantic similarity consistent for the original images. The pixel-level and feature-level content loss are used to improve the generated image quality [119]. Some other loss items are employed in GANs to enhance the hash codes learning. For instance, a distance matching regularizer is utilized to propagate the correlations between high-dimensional real-valued features and low-dimensional hash codes. Two loss functions that aim at promoting independence of binary codes are introduced in [120]. To summarize, using GANs for unsupervised hash learning is promising, but there exist much room for further exploration.

5 The State of the Art Performance

5.1 Datasets

To demonstrate the effectiveness of these noted methods, we choose 4 commonly used datasets: Holidays, Oxford-5k (including the extended Oxford-105k), Paris-6k (including the extended Paris-106k) and UKBench for a performance comparison. Details about these datasets are given below.

UKBench [156] consists of 10,200 images of various objects. The whole dataset has 2,550 groups of images. Each group includes 4 images of the same object from different viewpoints or illumination conditions. Each image in the dataset can be used as a query image, thus the number of query images is 10,200.

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 query image for each group. In each group, the first image is used as a query image for performance evaluation.

Oxford-5k [124] consists of 5,062 images for 11 particular Oxford buildings. Each image is represented by 5 queries by a hand-drawn bounding box, thus there are 55 query Regions of Interest (RoI) in total. An additional disjoint set of 100,000 distractor images is added to obtain Oxford-100k.

Paris-6k [157] includes 6,412 images collected from Flickr. It is categorized into 12 groups about specific Paris architecture. The dataset has 500 query images for evaluation. There are also 55 queries with bounding boxes. Images are annotated with the same four types of labels as used in the Oxford-5k dataset.

Annotations and evaluation protocols in Oxford-5k and Paris-6k are updated; additional queries and distractor images are added into the two datasets, producing the *Revisited Oxford* and *Revisited Paris* datasets [158]. Due to the popularity of Oxford-5k and Paris-6k, we mainly make performance evaluations on these two original datasets.

5.2 Evaluation Metrics

Average precision. Average precision (AP) refers to the coverage area under the precision-recall curve. A larger AP means a higher precision-recall curve and better retrieval performance. AP can be calculated using Eq. (17)

$$AP = \frac{\sum_{k=1}^{N} P(k) \cdot rel(k)}{R} \tag{17}$$

where R denotes the number of relevant results for the query image. $P(k)$ is the precision of the top k retrieved images. $rel(k)$ is an indicator function equal to 1, if the item within rank k is a relevant image and 0 otherwise. N is the total number of images. Mean average precision (mAP) is adopted for the evaluation over all query images, *i.e.*, $\frac{1}{Q}\sum_{q=1}^{Q} AP(q)$, where Q is the number of query images.

N-S score is a metric initially specific for UKBench [156]. In this dataset, there are 4 relevant images for all queries, the N-S score is the average 4 times for top-4 precision over the dataset.

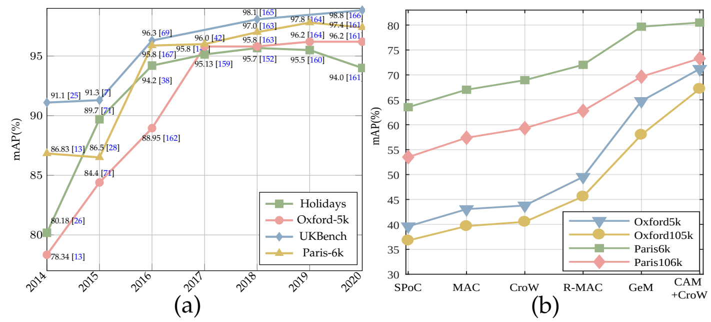

Fig. 11: (a) Performance improvement from 2014 to 2020. (b) mAP comparison of different feature aggregation methods shown in Figure 5.

5.3 Performance Comparison and Analysis

Overview. We conclude with the performance over these 4 datasets from 2014 to 2020 in Figure 11(a). At early period, DCNNs acted as powerful extractors and achieved good results, *e.g.*, mAP is 78.34% in [13] on Oxford-5k. Subsequently, the results increased significantly when some crucial factors were adopted, including feature fusion [159, 162, 166], feature aggregation [28, 48], and feature fine-tuning [152, 161]. For instance, the retrieval accuracy on UKBench reaches an mAP of 98.8% in [166] when an undirected graph is defined to fuse features and estimate their correlations. Network fine-tuning improves performance greatly. The accuracy increases steadily from 78.34% [13] to 96.2% [164] on the Oxford-5k dataset when manifold learning is used to fine-tune deep networks.

We report the results of methods using off-the-shelf models (Table 3) and fine-tuning networks (Table 4). In Table 3, single pass and multiple pass are analyzed, while supervised fine-tuning and unsupervised fine-tuning are compared in Table 4.

Evaluation for single feedforward pass. The common practice using this scheme is to enhance feature discrimination. In Table 3, we observe that fully-connected layers as feature extractors may reach a lower accuracy (*e.g.*, 74.7% on Holidays in [34]), compared to the counterpart convolutional layers. As the fully-connected layers lack structural information for instance-level retrieval. Further, layer-level feature fusion strategy improves retrieval accuracy. For example, Yu *et al*. [61] combined three layers (*Conv4*, *Conv5*, and *FC6*) (*e.g.*, an mAP of 91.4% on Holidays), outperforming the performance of non-fusion method in [7] (*e.g.*, mAP is 80.2%). Moreover, convolutional features embedded by BoW model reach a competitive performance on Oxford-5k and Paris-6k (73.9% and 82.0%, respectively), while its codebook size is 25k, which may affect the retrieval efficiency. For single pass scheme, methods shown in Figure 5 improve the discrimination of convolutional feature maps and perform differently in Table 3 (*e.g.*, 66.9% of R-MAC [157], 58.9% of SPoC [7] on Oxford-5k). We view this as a critical factors and further analyze.

Evaluation for multiple feedforward pass. The methods exemplified in Figure 6 are reported their results in multiple pass scheme. Among them, it seems that dense patch and Grid methods can achieve relatively better results, compared to other two methods. Extracting image patches densely using Overfeat [168] can reach best results on the 4 datasets [25]. Using rigid grid method reach competitive results (*e.g.*, an mAP of 87.2% on Paris-6k) [104]. 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 have a degree of purpose in processing image objects. This may be more efficient and less memory demanding. Using multiple pass scheme, spatial information is maintained better than the case using the single pass method. For example, a shallower network (AlexNet) and region proposal networks are used in [74], its result on UKBench is 3.81 (N-Score), higher than the one using deeper networks, such as [7, 34, 61]. Besides feeding image patches into the same network, model-level fusion also exploit complementary spatial information to improve the retrieval accuracy. For instance, as demonstrated in [32], which combines AlexNet and VGG, the results on the Holidays (81.74% of mAP) and UKBench (3.32 of N-Score) dataset are better than these in [50] (76.75% and 3.00, respectively).

Evaluation for supervised fine-tuning. Compared to the off-the-shelf models, fine-tuning deep networks usually improves accuracy, see Table 4. For instance, the result on Oxford-5k [28] by using a pre-trained VGG is improved from 66.9% to 81.5% in [37] when a single-margin Siamese loss is used. Similar trends can be also observed on the Paris-6k dataset. Although classification-based fine-tuning method is not excel at learning intra-class variability (*e.g.*, an mAP of 55.7% on Oxford-5k in [34]), its performance may be improved with powerful DCNNs and feature enhancement methods such as the attention mechanism in [102], leading to an mAP of 83.8% on Oxford-5k. As for verification-based fine-tuning methods, 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) [79]. Its results are lower than the results from [36] (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 [138], the effectiveness of retrieval is improved significantly where the results are 86.1% (on Oxford-5k) and 94.5% (on Paris-6k). Further, feature embedding methods are important for retrieval accuracy. For example, Ong *et al.* [37] 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 [39, 40].

Evaluation for unsupervised fine-tuning. Compared to supervised network fine-tuning, unsupervised fine-tuning methods are relatively less explored. The difficulty for unsupervised fine-tuning is to mine relevance of samples without ground-truth labels. In general, unsupervised fine-tuning methods produce lower performance than the supervised fine-tuning methods. For instance, supervised fine-tuning network by using Siamese loss in [169] achieves an mAP 88.4% on Holidays, while unsupervised fine-tuning network using the same loss function in [40, 42, 150] achieve 87.5%, 83.1%, and 82.5%, respectively. However, unsupervised fine-tuning methods can achieve a similar accuracy even outperform the supervised fine-tuning if a suited feature embedding method is used. For instance, Zhao *et al.* [151] explored global feature structure with modeling the manifold learning, producing an mAP of 85.4% (on Oxford-5k) and 96.3% (on Paris-6k). This is similar to the supervised method [138], whose results are 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 to 83.8% on Oxford-5k [102], while the precision is further improved to 92.0% when IME layer is used to embed features and fine-tuned in an unsupervised way [41]. Note that fine-tuning strategies are related to the type of the target retrieval datasets. As demonstrated in [105], fine-tuning on different datasets may hurt the final performance.

TABLE 2: Evaluations of mAP (%), N-S score, and average search time per image. "†" refers to the query time is evaluated in a global diffusion manner, while "‡" refers to the time is evaluated in a regional diffusion way.

	Oxford-5k (+100k)		Paris-6k (+100k)		Holidays		UKBench	
	mAP	Time	mAP	Time	mAP	Time	N-S	Time
[152]	91.3 (88.4)	5.45 *ms* (809 *ms*)	-	-	95.66	3.11 *ms*	3.93	4.91 *ms*
[164]	92.6 (91.8)	2 *ms* (10 *ms*)	-	-	-	-	-	-
[149]†	85.7 (-)	20 *ms* (-)	94.1 (-)	20 *ms* (-)	-	-	-	-
[149]‡	95.8 (-)	600 *ms* (-)	96.9 (-)	700 *ms* (-)	-	-	-	-
[170]	64.9 (58.8)	0.81 *ms* (0.82 *ms*)	-	-	-	-	-	-
[42]	64.8 (57.9)	0.77 *ms* (0.73 *ms*)	-	-	-	-	-	-
[36]	55.5 (-)	0.35 *ms* (-)	71.0 (-)	0.35 *ms* (-)	-	-	-	-

Retrieval efficiency is also an important criterion in deep image retrieval. Deep learning methods are usually based on large-size datasets. The training and testing of retrieval methods are mostly done on GPUs. Most prior work focus more on retrieval accuracy but less on efficiency. We report the retrieval accuracy and retrieval efficiency on 4 datasets in Table 2. The recorded time (in *ms*) indicates the average time for searching each query image. From these statistics, we observe some important trends. First, in general, the average retrieval time for each query image is less than 1*s*. Concretely, the recorded time is up to 809*ms* on Oxford-105k in [152], whose mAP is 88.4%. The retrieval time is 600*ms* on Oxford-5k and 700*ms* on Paris-6k in [149], whose time cost is caused by processing 21 regional features on each query image. Second, we observe the retrieval accuracy-efficiency balancing issue, which is significantly obvious on the Oxford-5k dataset. The average retrieval time are both less than 1*ms* in prior work [36, 42, 170], whose mAPs are lower than 70% (*i.e.*, 55.5%, 64.8%, and 64.9%, respectively). In contrast, the prior work [149, 152, 164] reach relatively higher mAPs (*i.e.*, 95.8%, 92.6%, and 91.3%, respectively), while this higher accuracy is at the expense of efficiency (more than 2*ms* even up to 600*ms*). Therefore, the trade-off of accuracy and efficiency is also an important factor to take into account in deep image retrieval, especially for large-scale datasets.

In addition, we explore other important factors which are common for retrieval, including the depth of networks, retrieval feature dimension, and feature aggregation methods.

The depth of networks. We compare the efficacy of DCNNs depth, following the DCNN fine-tuning protocols [1] in [42]. For

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

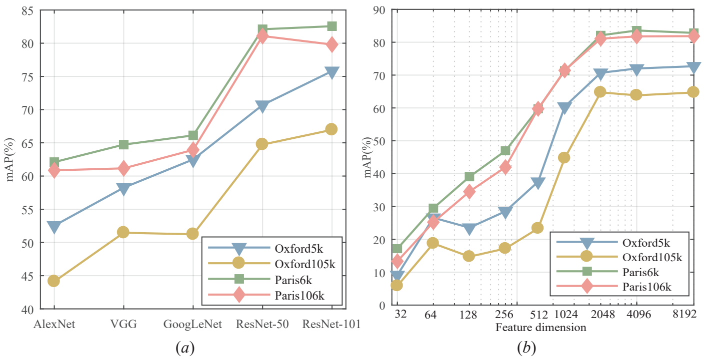

Fig. 12: (a) The effectiveness of different DCNNs on 4 datasets. All models are fine-tuned by the same loss function. The results are tested on the convolutional features with default dimension; (b) The impact of feature dimension on retrieval performance. These features are extracted by using ResNet-50.

fair comparisons, all convolutional feature maps from these networks are aggregated by MAC method [48], and fine-tuned by using the same learning rate. That means, the adopted methods are the same except the DCNNs have different depths. We use the default feature dimension (*i.e.* AlexNet (256-d), VGG (512-d), GoogLeNet (1024-d), ResNet-50/101 (2048-d)). The results are reported in Figure 12(a). We observe that the deeper networks is more beneficial for accuracy improvement, by extracting more discriminative features.

Feature dimension. We focus on varying the feature dimension of ResNet-50 from 32-d to 8092-d, by adding a fully-collected layers on the top of pooled convolutional features. The results are shown in Figure 12(b). It is expected that higher-dimension features capture much more semantics and are helpful for retrieval. The accuracy is achieving to a bottleneck as the dimension is increasing. For ResNet-50, we observe that the default feature dimension can produce competitive results.

Feature aggregation methods. For this factor, we mainly focus on the methods embedding convolutional feature maps, as depicted 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 different ways to aggregate the same off-the-shelf DCNN make difference for retrieval performance. The reported results provide a reference for feature aggregation when using convolutional layers for performing retrieval tasks.

6 Conclusions and Future Directions

In this survey, we reviewed the recent works on deep learning methods for image retrieval, and categorized it into deep image retrieval of off-the-shelf models and fine-tuned models according to the parameter updates of deep networks. Concretely, the off-the-shelf group is concerned with obtaining high-quality features by freezing the pre-stored parameters where network feedforward schemes, layer selection, and feature fusion methods are presented. While fine-tuned based methods deal with updating networks with optimal parameters for feature learning in both supervised and unsupervised approaches. For each group, we presented the corresponding methods and compared their differences. The corresponding experimental results are collected and analyzed for all the categorized works.

Deep learning has shown significant progress and spotlighted its capacity for image retrieval. Despite the great success, there are still many unsolved problems. Here, we introduce some promising trends as future research directions. We hope that this survey not only provides a better understanding of image retrieval but also facilitates future research activities and application developments in this field.

(1) *Zero-shot Learning for Image Retrieval*.

The popularity of media platforms and the rapid development of novel techniques makes it very convenient for people to share their images. As a result, the number of images increases immensely. In this case, there often exist “*unseen*“ images or categories. However, most datasets are static and offer a limited amount of objects and categories for feature learning. Thus, the retrieval algorithms or systems may suffer from the scarcity of the appropriate training data for these unseen images. Therefore, it is needed to extend conventional image retrieval methods to a zero-shot learning scenario, where we can retrieve both seen and unseen categories from the system. Furthermore, combined with unsupervised methods, the zero-shot learning algorithms can significantly improve the flexibility and generalization of image retrieval systems.

(2) *End-to-End Unsupervised Retrieval*.

Using supervisory information, network training or fine-tuning is more likely to mitigate the semantic gap. However, the sophisticated supervised learning algorithms are in most cases not very general because there is usually not enough supervisory information available. Thereby, it is necessary to explore unsupervised image retrieval, which has been studied less [147]. Therefore, as a solution, the earlier noted manifold learning is a way to mine the samples using relevance context information. The self-supervision information is learned based on graph discovery in the manifold space. However, the whole training process is not end-to-end yet. Currently, graph convolutional networks [153] have been used to replace the diffusion process for end-to-end training [43].

(3) *Incremental Image Retrieval*.

Current image retrieval focuses on static datasets and is not suited for incremental scenarios [171, 172]. That is, most of these approaches assume that images from all categories are available during training. This assumption may be restrictive in real-world applications as new categories are constantly emerging. Repetitive fine-tuning on both old and new images is time-consuming and inefficient, while fine-tuning only on the new images may lead to catastrophic forgetting, thereby resulting in severe degradation of the retrieval performance for the old categories. Therefore, one practical direction would be to build an up-to-date retrieval model to handle incremental streams of new categories, while retaining its previous performance on existing categories without forgetting.

(4) *Deploy Image Retrieval for Practical Applications*.

Existing image retrieval technologies are trained and evaluated on standard benchmarks such as the Oxford and Paris datasets, and various metric learning methods are explored for retrieval on fine-grained datasets. However, these technologies are still far from the real-world applications such as face search, fashion search, person re-identification, shopping recommendation system, or medical image retrieval. In these practical applications, the purpose of image retrieval, may not just be retrieving images for general content on standard benchmarks, but also for more refined information. It is challenging to deploy image retrieval for specific scenario. For example, as a specific instance search topic, person re-identification systems may encounter images with low-resolution or with inferior quality due to inadequate illumination. Existing techniques such as Attention mechanisms and the region proposal networks *etc*. can be adopted to guarantee performance. On the other hand, it is valuable to explore multi-modal retrieval in practical applications. That means, image retrieval can also be combined with other auxiliary modalities such as words, phrases, and sentences to meet different retrieval expectations of users.

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. (TOMM)*, 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] 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.

[5] 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.

[6] 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.

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

[8] 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.

[9] Y. Cao, M. Long, J. Wang, H. Zhu, and Q. Wen, "Deep quantization network for efficient image retrieval." in *AAAI*, 2016, pp. 3457–3463.

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

[11] R. Furuta, N. Inoue, and T. Yamasaki, "Efficient and interactive spatial-semantic image retrieval," in *MMM*, 2018, pp. 190–202.

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

[13] 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.

[14] 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.

[15] 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.

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

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

[18] 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.

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

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

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

[22] G. Huang, Z. Liu, L. Van Der Maaten, and K. Q. Weinberger, "Densely connected convolutional networks," in *CVPR*, 2017, pp. 4700–4708.

[23] 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.

[24] J. Long, E. Shelhamer, and T. Darrell, "Fully convolutional networks for semantic segmentation," in *CVPR*, 2015, pp. 3431–3440.

[25] 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.

[26] Y. Gong, L. Wang, R. Guo, and S. Lazebnik, "Multi-scale orderless pooling of deep convolutional activation features," in *ECCV*, 2014, pp. 392–407.

[27] J. Yosinski, J. Clune, Y. Bengio, and H. Lipson, "How transferable are features in deep neural networks?" in *NeurIPS*, 2014, pp. 3320–3328.

[28] G. Tolias, R. Sicre, and H. Jégou, "Particular object retrieval with integral max-pooling of CNN activations," in *ICLR*, 2015, pp. 1–12.

[29] A. Jiménez, J. M. Alvarez, and X. Giró Nieto, "Class-weighted convolutional features for visual instance search," in *BMVC*, 2017, pp. 1–12.

[30] 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. (TOMM)*, vol. 15, no. 2, pp. 1–22, 2019.

[31] 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.

[32] Y. Liu, Y. Guo, S. Wu, and M. S. Lew, "Deepindex for accurate and efficient image retrieval," in *ICMR*, 2015, pp. 43–50.

[33] P. Wu, S. C. Hoi, H. Xia, P. Zhao, D. Wang, and C. Miao, "Online multimodal deep similarity learning with application to image retrieval," in *ACM MM*, 2013, pp. 153–162.

[34] A. Babenko, A. Slesarev, A. Chigorin, and V. Lempitsky, "Neural codes for image retrieval," in *ECCV*, 2014, pp. 584–599.

[35] C.-Q. Huang, S.-M. Yang, Y. Pan, and H.-J. Lai, "Object-location-aware hashing for multi-label image retrieval via automatic mask learning," *IEEE Trans. Image Process.*, vol. 27, no. 9, pp. 4490–4502, 2018.

[36] N. Garcia and G. Vogiatzis, "Learning non-metric visual similarity for image retrieval," *Image Vis. Comput.*, vol. 82, pp. 18–25, 2019.

[37] E.-J. Ong, S. Husain, and M. Bober, "Siamese network of deep fisher-vector descriptors for image retrieval," *arXiv preprint arXiv:1702.00338*, 2017.

[38] A. Gordo, J. Almazán, J. Revaud, and D. Larlus, "Deep image retrieval: Learning global representations for image search," in *ECCV*, 2016, pp. 241–257.

[39] R. Arandjelovic, P. Gronat, A. Torii, T. Pajdla, and J. Sivic, "NetVLAD: CNN architecture for weakly supervised place recognition," in *CVPR*, 2016, pp. 5297–5307.

[40] 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.

[41] 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, 2018.

[42] 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, 2017.

[43] 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.

[44] C. Chang, G. Yu, C. Liu, and M. Volkovs, "Explore-exploit graph traversal for image retrieval," in *CVPR*, 2019, pp. 9423–9431.

[45] Y. Shen, J. Qin, J. Chen, M. Yu, L. Liu, F. Zhu, F. Shen, and L. Shao, "Auto-encoding twin-bottleneck hashing," in *CVPR*, 2020, pp. 2818–2827.

[46] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *arXiv preprint arXiv:1409.1556*, 2014.

[47] 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.

[48] 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.

[49] H. Jun, B. Ko, Y. Kim, I. Kim, and J. Kim, "Combination of multiple global descriptors for image retrieval," *arXiv preprint arXiv:1903.10663*, 2019.

[50] 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.

[51] C. Qi, C. Shi, J. Xu, C. Wang, and B. Xiao, "Spatial weighted fisher vector for image retrieval," in *ICME*, 2017, pp. 463–468.

[52] 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.

[53] 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.
[54] H. Liu, R. Wang, S. Shan, and X. Chen, "Deep supervised hashing for fast image retrieval," in *CVPR*, 2016, pp. 2064–2072.
[55] C. Deng, E. Yang, T. Liu, J. Li, W. Liu, and D. Tao, "Unsupervised semantic-preserving adversarial hashing for image search," *IEEE Trans. Image Process.*, vol. 28, no. 8, pp. 4032–4044, 2019.
[56] H. Hu, K. Wang, C. Lv, J. Wu, and Z. Yang, "Semi-supervised metric learning-based anchor graph hashing for large-scale image retrieval," *IEEE Trans. Image Process.*, pp. 739–754, 2018.
[57] D. Deng, R. Wang, H. Wu, H. He, Q. Li, and X. Luo, "Learning deep similarity models with focus ranking for fabric image retrieval," *Image Vis. Comput.*, vol. 70, pp. 11–20, 2018.
[58] K. Zhou, Y. Liu, J. Song, L. Yan, F. Zou, and F. Shen, "Deep self-taught hashing for image retrieval," in *ACM MM*, 2015, pp. 1215–1218.
[59] Y. Li, Z. Miao, J. Wang, and Y. Zhang, "Nonlinear embedding neural codes for visual instance retrieval," *Neurocomputing*, vol. 275, pp. 1275–1281, 2018.
[60] 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.
[61] 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.
[62] X.-S. Wei, J.-H. Luo, J. Wu, and Z.-H. Zhou, "Selective convolutional descriptor aggregation for fine-grained image retrieval," *IEEE Trans. Image Process.*, vol. 26, no. 6, pp. 2868–2881, 2017.
[63] J. Sivic and A. Zisserman, "Video google: A text retrieval approach to object matching in videos," in *CVPR*, 2003, pp. 1470–1477.
[64] H. Jégou, M. Douze, C. Schmid, and P. Pérez, "Aggregating local descriptors into a compact image representation," in *CVPR*, 2010, pp. 3304–3311.
[65] T. Jaakkola and D. Haussler, "Exploiting generative models in discriminative classifiers," in *NeurIPS*, 1999, pp. 487–493.
[66] M. Oquab, L. Bottou, I. Laptev, and J. Sivic, "Learning and transferring mid-level image representations using convolutional neural networks," in *CVPR*, 2014, pp. 1717–1724.
[67] 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.
[68] 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.
[69] 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.
[70] 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.
[71] A. Sharif Razavian, J. Sullivan, A. Maki, and S. Carlsson, "A baseline for visual instance retrieval with deep convolutional networks," in *ICLR*, 2015.
[72] D. Xu, X. Alameda-Pineda, J. Song, E. Ricci, and N. Sebe, "Cross-paced representation learning with partial curricula for sketch-based image retrieval," *IEEE Trans. Image Process.*, vol. 27, no. 9, pp. 4410–4421, 2018.
[73] 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.
[74] K. Reddy Mopuri and R. Venkatesh Babu, "Object level deep feature pooling for compact image representation," in *CVPR Workshops*, 2015, pp. 62–70.
[75] C. L. Zitnick and P. Dollár, "Edge boxes: Locating object proposals from edges," in *ECCV*, 2014, pp. 391–405.
[76] L. Xie, R. Hong, B. Zhang, and Q. Tian, "Image classification and retrieval are one," in *ICMR*, 2015, pp. 3–10.
[77] J. Mairal, P. Koniusz, Z. Harchaoui, and C. Schmid, "Convolutional kernel networks," in *NeurIPS*, 2014, pp. 2627–2635.
[78] J. Song, Q. Yu, Y.-Z. Song, T. Xiang, and T. M. Hospedales, "Deep spatial-semantic attention for fine-grained sketch-based image retrieval." in *ICCV*, 2017, pp. 5552–5561.
[79] 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.
[80] 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.
[81] L. Zheng, Y. Zhao, S. Wang, J. Wang, and Q. Tian, "Good practice in CNN feature transfer," *CoRR*, vol. abs/1604.00133, 2016.
[82] T. Ng, V. Balntas, Y. Tian, and K. Mikolajczyk, "SOLAR: Second-order loss and attention for image retrieval," in *ECCV*, 2020, pp. 253–270.
[83] P. Kulkarni, J. Zepeda, F. Jurie, P. Perez, and L. Chevallier, "Hybrid multi-layer deep CNN/aggregator feature for image classification," in *ICASSP*, 2015, pp. 1379–1383.
[84] D. Yu, Y. Liu, Y. Pang, Z. Li, and H. Li, "A multi-layer deep fusion convolutional neural network for sketch based image retrieval," *Neurocomputing*, vol. 296, pp. 23–32, 2018.
[85] C. Shen, C. Zhou, Z. Jin, W. Chu, R. Jiang, Y. Chen, and X.-S. Hua, "Learning feature embedding with strong neural activations for fine-grained retrieval," in *ACM MM*, 2017, pp. 424–432.
[86] B. Cao, A. Araujo, and J. Sim, "Unifying deep local and global features for efficient image search," in *ECCV*, 2020, pp. 726–743.
[87] K. Chatfield, K. Simonyan, A. Vedaldi, and A. Zisserman, "Return of the devil in the details: Delving deep into convolutional nets," in *BMVC*, 2014.
[88] 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.
[89] Z. Ding, L. Song, X. Zhang, and Z. Xu, "Selective deep ensemble for instance retrieval," *Multimed. Tools. Appl.*, pp. 1–17, 2018.
[90] W. Kim, B. Goyal, K. Chawla, J. Lee, and K. Kwon, "Attention-based ensemble for deep metric learning," in *ECCV*, 2018, pp. 736–751.
[91] T. Bui, L. Ribeiro, M. Ponti, and J. Collomosse, "Sketching out the details: Sketch-based image retrieval using convolutional neural networks with multi-stage regression," *Comput. Graph.*, vol. 71, pp. 77–87, 2018.
[92] K. Ozaki and S. Yokoo, "Large-scale landmark retrieval/recognition under a noisy and diverse dataset," in *CVPR Workshop*, 2019.
[93] H. Xuan, R. Souvenir, and R. Pless, "Deep randomized ensembles for metric learning," in *ECCV*, 2018, pp. 723–734.
[94] J. Pan, E. Sayrol, X. Giro-i Nieto, K. McGuinness, and N. E. O'Connor, "Shallow and deep convolutional networks for saliency prediction," in *CVPR*, 2016, pp. 598–606.
[95] B.-C. Chen, L. S. Davis, and S.-N. Lim, "An analysis of object embeddings for image retrieval," *arXiv preprint arXiv:1905.11903*.
[96] Y.-L. Boureau, J. Ponce, and Y. LeCun, "A theoretical analysis of feature pooling in visual recognition," in *ICML*, 2010, pp. 111–118.
[97] 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. (TOMM)*, vol. 10, no. 3, pp. 1–20, 2014.
[98] 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.
[99] 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.
[100] D.-j. Jeong, S. Choo, W. Seo, and N. I. Cho, "Regional deep feature aggregation for image retrieval," in *ICASSP*, 2017, pp. 1737–1741.
[101] G. Li and Y. Yu, "Visual saliency detection based on multiscale deep CNN features," *IEEE Trans. Image Process.*, vol. 25, no. 11, pp. 5012–5024, 2016.
[102] 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.
[103] M. Cornia, L. Baraldi, G. Serra, and R. Cucchiara, "Predicting human eye fixations via an lstm-based saliency attentive model," *IEEE Trans. Image Process.*, vol. 27, no. 10, pp. 5142–5154, 2018.
[104] 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.
[105] V. Chandrasekhar, J. Lin, O. Morere, H. Goh, and A. Veillard, "A practical guide to CNNs and fisher vectors for image instance retrieval," *Signal Process.*, vol. 128, pp. 426–439, 2016.
[106] J. Kim and S.-E. Yoon, "Regional attention based deep feature for image retrieval," in *BMVC*, 2018, pp. 209–223.
[107] B. Chen and W. Deng, "Hybrid-attention based decoupled metric learning for zero-shot image retrieval," in *CVPR*, 2019, pp. 2750–2759.
[108] C. Deng, E. Yang, T. Liu, and D. Tao, "Two-stream deep hashing with class-specific centers for supervised image search," *IEEE Trans. Neural. Netw. Learn. Syst.*, 2019.
[109] L. Liu, F. Shen, Y. Shen, X. Liu, and L. Shao, "Deep sketch hashing: Fast free-hand sketch-based image retrieval," in *CVPR*, 2017, pp. 2862–2871.
[110] R. Kang, Y. Cao, M. Long, J. Wang, and P. S. Yu, "Maximum-margin hamming hashing," in *ICCV*, 2019, pp. 8252–8261.
[111] F. Zhao, Y. Huang, L. Wang, and T. Tan, "Deep semantic ranking based hashing for multi-label image retrieval," in *CVPR*, 2015, pp. 1556–1564.
[112] F. Long, T. Yao, Q. Dai, X. Tian, J. Luo, and T. Mei, "Deep domain adaptation hashing with adversarial learning," in *ACM SIGIR*, 2018, pp. 725–734.
[113] Y. Cao, B. Liu, M. Long, J. Wang, and M. KLiss, "Hashgan: Deep

learning to hash with pair conditional wasserstein gan," in *CVPR*, 2018, pp. 1287–1296.
[114] E. Yang, T. Liu, C. Deng, W. Liu, and D. Tao, "Distillhash: Unsupervised deep hashing by distilling data pairs," in *CVPR*, 2019, pp. 2946–2955.
[115] J. Zhang and Y. Peng, "Ssdh: semi-supervised deep hashing for large scale image retrieval," *IEEE Trans. Circuits Syst. Video Technol.*, 2017.
[116] M. A. Carreira-Perpinán and R. Raziperchikolaei, "Hashing with binary autoencoders," in *CVPR*, 2015, pp. 557–566.
[117] 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.
[118] Y. Gu, S. Wang, H. Zhang, Y. Yao, W. Yang, and L. Liu, "Clustering-driven unsupervised deep hashing for image retrieval," *Neurocomputing*, vol. 368, pp. 114–123, 2019.
[119] J. Song, "Binary generative adversarial networks for image retrieval," in *AAAI*, 2017.
[120] K. G. Dizaji, F. Zheng, N. S. Nourabadi, Y. Yang, C. Deng, and H. Huang, "Unsupervised deep generative adversarial hashing network," in *CVPR*, 2018, pp. 3664–3673.
[121] V. Erin Liong, J. Lu, G. Wang, P. Moulin, and J. Zhou, "Deep hashing for compact binary codes learning," in *CVPR*, 2015, pp. 2475–2483.
[122] F. Cakir, K. He, S. A. Bargal, and S. Sclaroff, "Hashing with mutual information," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 41, no. 10, pp. 2424–2437, 2019.
[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] W. Chen, X. Chen, J. Zhang, and K. Huang, "Beyond triplet loss: a deep quadruplet network for person re-identification," in *CVPR*, 2017, pp. 403–412.
[126] J. Wang, F. Zhou, S. Wen, X. Liu, and Y. Lin, "Deep metric learning with angular loss," in *ICCV*, 2017, pp. 2593–2601.
[127] K. Sohn, "Improved deep metric learning with multi-class n-pair loss objective," in *NeurIPS*, 2016, pp. 1857–1865.
[128] H. Oh Song, Y. Xiang, S. Jegelka, and S. Savarese, "Deep metric learning via lifted structured feature embedding," in *CVPR*, 2016, pp. 4004–4012.
[129] X. Wang, Y. Hua, E. Kodirov, G. Hu, R. Garnier, and N. M. Robertson, "Ranked list loss for deep metric learning," in *CVPR*, 2019, pp. 5207–5216.
[130] L. Chen and Y. He, "Dress fashionably: Learn fashion collocation with deep mixed-category metric learning," in *AAAI*, 2018, pp. 2103–2110.
[131] Y. Movshovitz-Attias, A. Toshev, T. K. Leung, S. Ioffe, and S. Singh, "No fuss distance metric learning using proxies," in *ICCV*, 2017, pp. 360–368.
[132] S. Kim, D. Kim, M. Cho, and S. Kwak, "Proxy anchor loss for deep metric learning," in *CVPR*, 2020, pp. 3238–3247.
[133] W. Zheng, Z. Chen, J. Lu, and J. Zhou, "Hardness-aware deep metric learning," in *CVPR*, 2019, pp. 72–81.
[134] 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.
[135] E. Simo-Serra, E. Trulls, L. Ferraz, I. Kokkinos, P. Fua, and F. Moreno-Noguer, "Discriminative learning of deep convolutional feature point descriptors," in *ICCV*, 2015, pp. 118–126.
[136] M. Tzelepi and A. Tefas, "Exploiting supervised learning for finetuning deep CNNs in content based image retrieval," in *ICPR*, 2016, pp. 2918–2923.
[137] J. Song, T. He, L. Gao, X. Xu, and H. T. Shen, "Deep region hashing for efficient large-scale instance search from images," *arXiv preprint arXiv:1701.07901*, 2017.
[138] 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.
[139] 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.
[140] 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.
[141] J. Lu, J. Hu, and J. Zhou, "Deep metric learning for visual understanding: An overview of recent advances," *IEEE Signal Process. Mag.*, vol. 34, no. 6, pp. 76–84, 2017.
[142] X. Wang, H. Zhang, W. Huang, and M. R. Scott, "Cross-batch memory for embedding learning," in *CVPR*, 2020, pp. 6388–6397.
[143] B. Harwood, B. Kumar, G. Carneiro, I. Reid, T. Drummond *et al.*, "Smart mining for deep metric learning," in *ICCV*, 2017, pp. 2821–2829.
[144] K. He, Y. Lu, and S. Sclaroff, "Local descriptors optimized for average precision," in *CVPR*, 2018, pp. 596–605.
[145] 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.
[146] 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.
[147] N. Aziere and S. Todorovic, "Ensemble deep manifold similarity learning using hard proxies," in *CVPR*, 2019, pp. 7299–7307.
[148] M. Donoser and H. Bischof, "Diffusion processes for retrieval revisited," in *CVPR*, 2013, pp. 1320–1327.
[149] 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.
[150] A. Iscen, G. Tolias, Y. Avrithis, and O. Chum, "Mining on manifolds: Metric learning without labels," in *CVPR*, 2018, pp. 7642–7651.
[151] 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.
[152] 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.
[153] T. N. Kipf and M. Welling, "Semi-supervised classification with graph convolutional networks," in *ICLR*, 2017.
[154] T. Maria and T. Anastasios, "Deep convolutional image retrieval: A general framework," *Signal Process. Image Commun.*, vol. 63, pp. 30–43, 2018.
[155] R.-C. Tu, X.-L. Mao, B.-S. Feng, and S.-Y. Yu, "Object detection based deep unsupervised hashing," in *IJCAI*, 2019, pp. 3606–3612.
[156] D. Nister and H. Stewenius, "Scalable recognition with a vocabulary tree," in *CVPR*, 2006, pp. 2161–2168.
[157] 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.
[158] F. Radenovic, A. Iscen, G. Tolias, Y. Avrithis, and O. Chum, "Revisiting oxford and paris: Large-scale image retrieval benchmarking," in *CVPR*, 2018.
[159] A. Alzu'bi, A. Amira, and N. Ramzan, "Content-based image retrieval with compact deep convolutional features," *Neurocomputing*, vol. 249, pp. 95–105, 2017.
[160] 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.
[161] L. T. Alemu and M. Pelillo, "Multi-feature fusion for image retrieval using constrained dominant sets," *Image Vis Comput*, vol. 94, p. 103862, 2020.
[162] 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.
[163] A. Iscen, Y. Avrithis, G. Tolias, T. Furon, and O. Chum, "Fast spectral ranking for similarity search," in *CVPR*, 2018, pp. 7632–7641.
[164] 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.
[165] 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.
[166] L. P. Valem and D. C. G. Pedronette, "Graph-based selective rank fusion for unsupervised image retrieval," *Pattern Recognit Lett*, 2020.
[167] 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.
[168] 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.
[169] 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.
[170] Q. Wang, J. Lai, L. Claesen, Z. Yang, L. Lei, and W. Liu, "A novel feature representation: Aggregating convolution kernels for image retrieval," *Neural Networks*, vol. 130, pp. 1–10, 2020.
[171] D. Wu, Q. Dai, J. Liu, B. Li, and W. Wang, "Deep incremental hashing network for efficient image retrieval," in *CVPR*, 2019, pp. 9069–9077.
[172] 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.

TABLE 3: mAP (%) evaluation on four benchmarks. "&" means the models or the layers are combined together to learn features. "PCA_w indicates PCA with whitening is performed on the extracted features. "MP" means Max Pooling. "SP" means Sum Pooling. The patch extraction methods, including SPM, RP, *etc.* are illustrated in Figure 6. The network with "♣" has a similar architecture with AlexNet. "(-)" means corresponding results were not reported.

	Methods Type		Used in	Base DCNN	Output Layer	Feat. Enhance.	Feat. Dim.	Holidays	UKB	Oxford5k (+100k)	Paris6k (+100k)	Brief Conclusions and Highlights
(1) Retrieval with Off-the-Shelf DCNN Models	Single Feed		[34]	AlexNet	FC6	PCA	128	74.7	3.42 (N-S)	43.3 (38.6)	-	A milestone work in which AlexNet is fine-tuned on the landmarks for retrieval. Performance of compressed neural codes of different layers are investigated.
	Single Feed		[28]	VGG16	Conv5	R-MAC + PCA_w	512	-	-	66.9 (61.6)	83.0 (75.7)	Adopting the sliding windows with different scales on the convolutional feature maps to preserve spatial information. Re-ranking methods are also used to improve the retrieval accuracy further.
	Single Feed		[10]	VGG16	Conv5	CroW + PCA_w	256	85.1	-	68.4 (63.7)	76.5 (69.1)	The spatial- and channel- wise weighting mechanism are utilized to highlight crucial convolutional features. Different feature dimensions are tested on 3 datasets. Query expansion is explored to improve retrieval accuracy.
	Single Feed		[70]	VGG16	Conv5	BoW + PCA_w	25k	-	-	73.9 (59.3)	82.0 (64.8)	An important work encodes the convolutional activations as local features into BoW descriptors. The global features and local features are explored in the work where local features have higher accuracy than global ones.
	Single Feed		[7]	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 feature representation. Full query image is fed into networks when the Oxford-5k and Oxford-105k datasets are used.
	Single Feed		[61]	VGG16	FC6 & Conv4 & Conv5	SP	4096	91.4	3.68 (N-S)	61.5 (-)	-	Exploring combinations of different fully-connected layers and convolutional layers in different deep networks which demonstrates the complementary properties between high-layer and low-layer for image retrieval.
	Single Feed		[31]	VGG16	Conv5	SP+PCA_w	4096	-	-	86.1 (80.4)	79.1 (73.6)	Ranking "probabilistic proposals" on the feature maps in an unsupervised manner to weight then select regional representations. Global semantic representations are concatenated from all regional representations.
	Multiple Feed	SPM	[32]	AlexNet & VGG19	FC6-7 FC17-18	BoW + PCA	512	81.74*	3.32* (N-S)	-	75.35*	Exploring layer-level and model-level fusion (combination) methods to extract features, deep features are further embedded into BoW descriptors separately. $*$ indicates the results are from "fc7+fc18" combination.
	Multiple Feed	SPM	[50]	CNN-M♣ [87]	Conv3 & Conv5 & FC7	SP or MP + BoW	20k	76.75	3.00 (N-S)	-	-	Exploring the combinations of different fully-connected layers and convolutional layers within the same deep network. Hamming embedding method is also investigated to improve the retrieval results significantly.
	Multiple Feed	Dense Patch	[48]	VGG16	Conv5	SP or MP + PCA_w	32k	89.6	95.1 (mAP)	84.3 (-)	87.9 (-)	Investigating an efficient pipeline for visual instance retrieval. Image sub-patches are extracted in a dense manner. Geometric invariance is taken into consideration when spatial max pooling is used to aggregate patch features.
	Multiple Feed	Dense Patch	[25]	Overfeat [168]	FC	PCA_w	15k	84.3	91.1 (mAP)	68.0 (-)	79.5 (-)	A milestone work which explores deep network to accomplish image retrieval. Image sub-patches are extracted at different locations with different sizes. Retrieval results are based on the sum of all patch features.
	Multiple Feed	Dense Patch	[26]	AlexNet	FC7	VLAD + PCA_w	2048	80.2	-	-	-	The image patches are extracted in a dense manner which are more dense than the extraction method used in [25]. Multi-scale patch features are further embedded into VLAD descriptors.
	Multiple Feed	RP	[60]	Goog-LeNet	Conv5	VLAD + PCA_w	128	84.13	3.81 (N-S)	64.84 (-)	76.76 (-)	Fusing complementary CNN and SIFT features for retrieval which includes 3 level contents. Features are learned for object proposals. Object-level and point-level features concatenation schemes are explored.
	Multiple Feed	RP	[100]	VGG19	Conv5	SP + PCA_w	512	88.6	-	74.1 (-)	80.2 (-)	Exploring to extract the spatial- and channel-wise weights to highlight important which mitigates the feature burstiness issue. Images are retrieved in a region-by-region manner.
	Multiple Feed	RP	[74]	AlexNet	FC6	MP + PCA_w	512	88.46	3.81 (N-S)	60.71 (-)	66.23 (-)	Exploring the impact of proposal number. Features of patch are aggregated in an orderless manner. Fine-tuning methods are also explored.
	Multiple Feed	Grid	[104]	VGG19	Conv5	BoW + PCA_w	500k	84.6	-	83.3 (-)	87.2 (-)	Image sub-patches are obtained using uniform square mesh. Patch features are encoded into BoW descriptors. Voting scores between matched local descriptors are used to solve the burstiness issue.

TABLE 4: Continued for Table 3. "On Sup." means the metric learning methods are in a supervised manner while "On Unsup." means in an unsupervised manner. "Classification based" means the models are fine-tuned using the classification-based loss function (CE loss) in Eq. 10. "S-M constra. Loss" refers the single margin contrastive loss in Eq. 13, while "D-M constra. Loss" denotes the double margin contrastive loss in Eq. 16. Regression loss is in the form of Eq. 12. Triplet loss is in the form of Eq. 14.

	Methods	Type	Used in	Base DCNN	Output Layer	Feat. Enhance.	Loss Func.	Feat. Dim.	Holidays	UKB	Oxford5k (+100k)	Paris6k (+100k)	Brief Conclusions and Highlights
(2) Retrieval via Learning DCNN Representations	On Sup.	Classification based	[102]	ResNet-101	Conv4 Block	Attention + PCA_w	CE Loss	2048	-	-	83.8 (82.6)	85.0 (81.7)	Exploring the FCN to construct feature pyramids of different sizes. The ResNet-101 is fine-tuned and tested on the large-scale Google-Landmarks dataset, also evaluated on the Holidays and Oxford datasets.
	On Sup.	Classification based	[34]	AlexNet	FC6	PCA	CE Loss	128	78.9	3.29 (N-S)	55.7 (52.3)	-	The first work about fine-tuning the pre-trained network for image retrieval using classification-based loss function. Compressed neural codes and different layers in deep networks are investigated.
	On Sup.	Matrix Trans.	[36]	VGG16	Conv5	PCA_w	Regression Loss	512	-	-	88.2 (82.1)	88.2 (82.9)	Visual similarity learning of similar and dissimilar pairs are performed by a neural network. The neural network is optimized by using regression loss, while the parameters of the backbone network are not updated.
	On Sup.	Fine-tuned RPN	[79]	VGG16	Conv5	MP and SP	Regression Loss	512	-	-	75.1 (-)	80.7 (-)	RPN is fine-tuned, which is based on output bounding box coordinates and class scores, for specific region query, the resultant features are region-targeted and suitable for object retrieval.
	On Sup.	Siamese Net.	[37]	VGG16	Conv5	FV + PCA_w	S-M Contra. Loss	512	-	-	81.5 (76.6)	82.4 (-)	Integrating Fisher Vector method on the top of VGG to encode features in an end-to-end manner instead of max pooling, the parameters in GMM model and the backbone network are optimized simultaneously.
	On Sup.	Siamese Net.	[169]	VGG16	Conv5	SP	S-M Contra. Loss	512	88.4	3.91 (N-S)	-	-	SIFT features are used as supervisory information for mining positive and negative samples to fine-tune ResNet50 within a siamese structure, the learning objective loss functions are SIFT-based and CNN-based.
	On Sup.	Siamese Net.	[140]	VGG16	FC6	PCA	D-M Contra. 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 for fine-tuning. Quartet-net helps avoid over-fitting when training on large-scale datasets.
	On Sup.	Triplet Net.	[39]	VGG16	VLAD Layer	PCA_w	Triplet Loss	256	79.9	-	62.5 (-)	72.0 (-)	A milestone work in which VLAD is integrated at the last convolutional layer of VGG16 network as a plugged layer, opens up the possibility of end-to-end learning for other ranking tasks.
	On Sup.	Triplet Net.	[138]	ResNet-101	Conv5 Block	MP + PCA_w	Triplet Loss	2048	90.3	-	86.1 (82.8)	94.5 (90.6)	Using RPN to fine-tune the backbone networks which include VGG and ResNet-101. Comparison between VGG and ResNet-101 demonstrates the deeper network achieves better retrieval results.
	On Unsup.	Siamese Net.	[150]	VGG16	Conv5	MP + PCA_w	S-M Contra. Loss	64	87.5	-	78.2 (72.6)	85.1 (78.0)	Exploring manifold learning for mining positive and negative samples in a fully unsupervised manner. The method is tested for features in global and regional kind, and surpasses the fully supervised approaches.
	On Unsup.	Siamese Net.	[42]	VGG16	Conv5	GeM Pooling	S-M Contra. Loss	512	83.1	-	82.0 (76.9)	79.7 (72.6)	Fine-tuning CNNs on an unordered dataset in a fully automated manner. Positive and negative samples are selected using Structure-from-Motion and Reconstructed 3D models. Multi-scale strategy is adopted.
	On Unsup.	Siamese Net.	[40]	VGG16	Conv5	PCA_w	S-M Contra. Loss	512	82.5	-	77.0 (69.2)	83.8 (76.4)	Employing Structure-from-Motion to select positive and negative samples from an unordered images which demonstrates the learned whitening has better results than PCA whitening. AlexNet and VGG are compared.
	On Unsup.	Siamese Net.	[41]	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 unodered datasets. Image regions are extracted at 3 different scales to obtain pyramid features.
	On Unsup.	Triplet Net.	[151]	ResNet-101	Conv5 Block	SP	Triplet Loss	2048	-	-	85.4 (85.1)	96.3 (94.7)	Exploring global feature structure with modeling the manifold learning to select positive and negative pairs. It effectively improves retrieval without the aid of additional labels, and needs to access to database in advance.