Universiteit Leiden
The Netherlands

Exploring deep learning for intelligent image retrieval

Chen, W.

Citation

Chen, W. (2021, October 13). *Exploring deep learning for intelligent image retrieval*. Retrieved from https://hdl.handle.net/1887/3217054

Version:	Publisher's Version
License:	Licence agreement concerning inclusion of doctoral thesis in the Institutional Repository of the University of Leiden
Downloaded from:	https://hdl.handle.net/1887/3217054

Note: To cite this publication please use the final published version (if applicable).

Chapter 2

A Comprehensive Review of Deep Image Retrieval

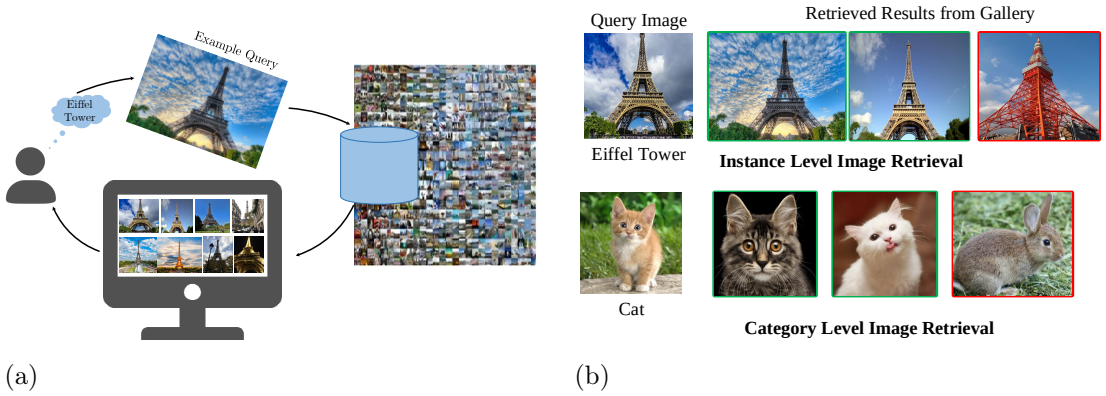
In recent years a vast amount of visual content has been generated and shared from various fields, such as social media platforms, medical images, and robotics. This abundance of content creation and sharing has introduced new challenges. In particular, searching databases for similar content, *i.e.* 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 chapter, 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 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 instance-based CBIR.

Keywords

Content based image retrieval, Deep learning, Convolutional neural networks, Literature review

This chapter is based on the following publication:

- Chen, W., Liu, Y., Wang, W., Bakker, E., Georgiou T., Fieguth P., Liu L., and Lew, M.S., "Deep Image Retrieval: A Survey." submitted to IEEE Transactions on Pattern Analysis and Machine Intelligence (*major revision*), 2021.



(a) (b)

Figure 2.1: Illustration of (a) the CBIR process and (b) categorization. The images in green frame are retrieved correctly, while the ones in red frame are matched incorrectly.

2.1 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. CBIR has been a longstanding research topic in the computer vision and multimedia community [1, 40]. With the present, exponentially increasing, amount of image and video data, the development of appropriate information systems that efficiently manage such large image collections is of utmost importance, with image searching being one of the most indispensable techniques.

A broad categorization of CBIR methodologies depends on the level of retrieval, *i.e.* instance level and category level. 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 conditions [3, 25]. In contrast, the goal of category level retrieval 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 chapter 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 [3, 25, 26, 41]. To enable accurate and efficient retrieval of massive image collections, *compact yet rich feature representations* are at the core of CBIR.

In the past two decades, remarkable progress has been made in image feature representations, which mainly consist of two important periods: feature engineering and feature learning (particularly deep learning). In the feature engineering era (*i.e.*

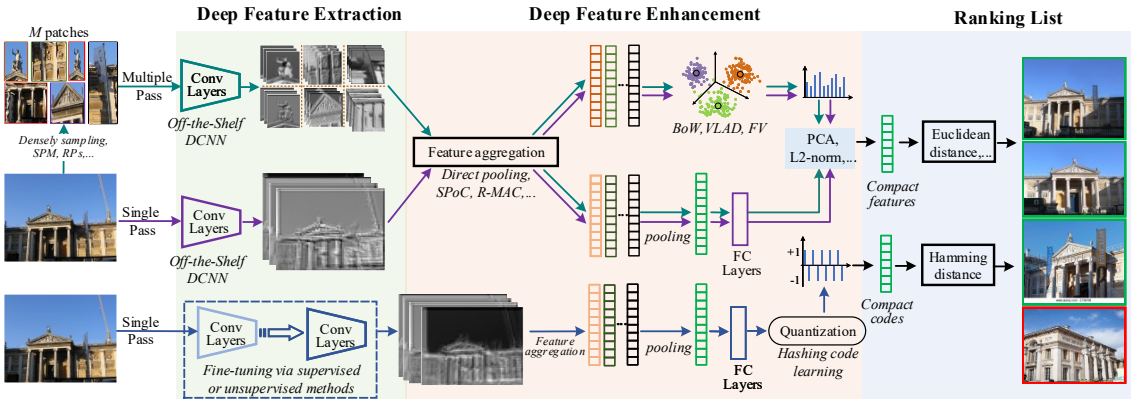


Figure 2.2: In deep image retrieval, feature embedding and aggregation methods are used to enhance the discrimination of deep features. Similarity is measured on these enhanced features using Euclidean or Hamming distances.

pre-deep learning), the field was dominated by milestone hand-engineered feature descriptors, such as the Scale-Invariant Feature Transform (SIFT) [4]. 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 [12]. Since then, deep learning has impacted a broad range of research areas, since DCNNs can learn powerful feature representations with multiple levels of abstraction directly from data. Deep learning techniques have attracted enormous attention and have brought about considerable breakthroughs in many computer vision tasks, including image classification [12, 13, 14], object detection [17], and image retrieval [26, 27, 42].

Excellent surveys for traditional image retrieval can be found in [1, 3, 40]. This chapter, in contrast, focuses on deep learning based methods. Deep learning for image retrieval is comprised of the essential stages shown in Figure 2.2 and various methods, focusing on one or more stages, have been proposed to improve retrieval accuracy and efficiency. In this chapter, we include comprehensive details about these methods, including feature fusion methods and network fine-tuning strategies *etc*, 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?*

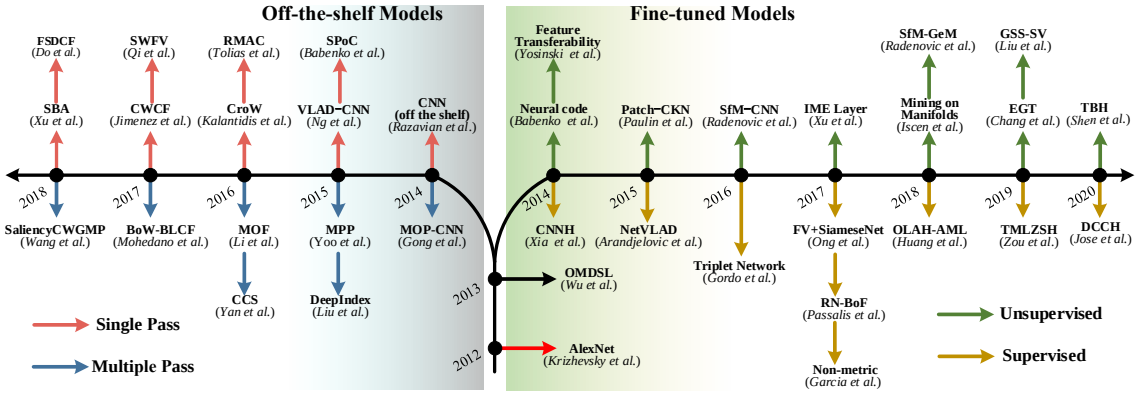


Figure 2.3: Representative methods in deep image retrieval, which are most fundamentally categorized according to whether the DCNN parameters are updated [43]. Off-the-shelf models (left) have model parameters which are not further updated or tuned when extracting features for image retrieval. The relevant methods focus on improving representations quality either by feature enhancement [26, 45, 46, 47] when using single pass schemes or by extracting representations for image patches [48] 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 [49, 50, 51, 52, 53, 54, 55] or unsupervised [32, 56, 57, 58, 59, 60]. See Sections 2.3 and 2.4 for details.

2.1.1 Summary of progress since 2012

After a highly successful image classification implementation based on AlexNet [12], 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 2.3, 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 [43] or are based on DCNNs with fixed parameters [29, 43, 44]. Regarding how to use the features effectively, researchers have proposed encoding and aggregation methods, such as R-MAC [31], CroW [26], and SPoC [25].

Recent progress for improving image retrieval can be categorized into network-level and feature-level perspectives, for which a detailed sub-categorization is shown in Figure 2.4. The network-level perspective includes network architecture improvement and network fine-tuning strategies. The feature-level perspective includes feature extraction and feature enhancement methods. Broadly this chapter will examine the four areas outlined as follows:

a. Improvements in network architectures (section 2.2.2)

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

- Deep learning for image retrieval
  - **Improvement in deep network architectures (section 2.2.2)**
    - Deepen networks: AlexNet [12], VGG [61], ResNet [13], *etc*
    - Widen networks: GoogLeNet [62], DenseNet [14], *etc*
  - Retrieval with off-the-shelf DCNN models (section 2.3)
    - **Deep feature extraction (section 2.3.1)**
      - Network feedforward scheme (section 2.3.1.1)
        - Single feedforward pass: MAC [63], R-MAC [31]
        - Multiple feedforward pass: SPM [48], RPNs [54]
      - Deep feature selection (section 2.3.1.2)
        - Fully-connected layer: layer concatenation [64]
        - Convolutional layer: SPoC [25], CroW [26]
      - Feature fusion strategy (section 2.3.1.3)
        - Layer-level fusion: MoF [65], MOP [29]
        - Model-level fusion: ConvNet fusion [61]
    - **Deep feature enhancement (section 2.3.2)**
      - Feature aggregation (section 2.3.2.1)
      - Feature embedding (section 2.3.2.2)
      - Attention mechanism (section 2.3.2.3)
        - Non-parameteric: SPoC [25], TSWVF [66]
        - Parameteric: DeepFixNet+SAM [67, 68]
      - Deep hash embedding (section 2.3.2.4)
        - Supervised hashing: Metric Learning [51]
        - Unsupervised hashing: KNN [69], k-means [70]
  - **Retrieval via learning DCNN representations (section 2.4)**
    - Supervised fine-tuning (section 2.4.1)
      - Classification-based fine-tuning (section 2.4.1.1)
      - Verification-based fine-tuning (section 2.4.1.2)
        - Transformation matrix: Non-metric [52]
        - Siamese networks: [53, 71]
        - Triplet networks: [53, 71]
    - Unsupervised fine-tuning (section 2.4.2)
      - Manifold learning sample mining: Diffusion Net [58]
      - AutoEncoder-based fine-tuning: KNN [72], GANs [60]

Figure 2.4: This chapter is organized around four key aspects in deep image retrieval, shown in boldface.

extract high-level abstract and semantic-aware features [13, 61]. It is also possible to concatenate multi-scale features in parallel, such as the Inception module in GoogLeNet [62], which we refer to as widening.

b. Deep feature extraction (section 2.3.1)

Neurons of FC layers and convolutional layers have different receptive fields, thus providing three ways to extract features: local features from convolutional layers [25, 31], global features from FC layers [48, 73] and fusions of two kinds of features [74, 75]; 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.

c. Deep feature enhancement (section 2.3.2)

Feature enhancement is used to improve feature's discriminative ability. Directly,

aggregate features can be trained simultaneously with deep networks [76]; alternatively, feature embedding methods including BoW [7], VLAD [28], and FV [8] embed local features into global ones. These methods are trained with networks separately (codebook-based) or jointly (codebook-free). Further, hashing methods [77] encode the real-valued features into binary codes to improve retrieval efficiency. The feature enhancement strategy significantly influences the efficiency of image retrieval.

d. Network fine-tuning for learning representations (section 2.4)

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 [50, 70, 78], 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.

2.1.2 Key challenges

Deep learning has been successful in learning 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 chapter 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 from images [10]. There is significant interest in learning deep features which are higher-level and semantic-aware, to better preserve the similarities of images [10]. In the past few years, various learning strategies, including feature fusion [29, 65] and feature enhancement methods [25, 31, 66] 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 [3]. Currently available deep networks are initially trained for 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

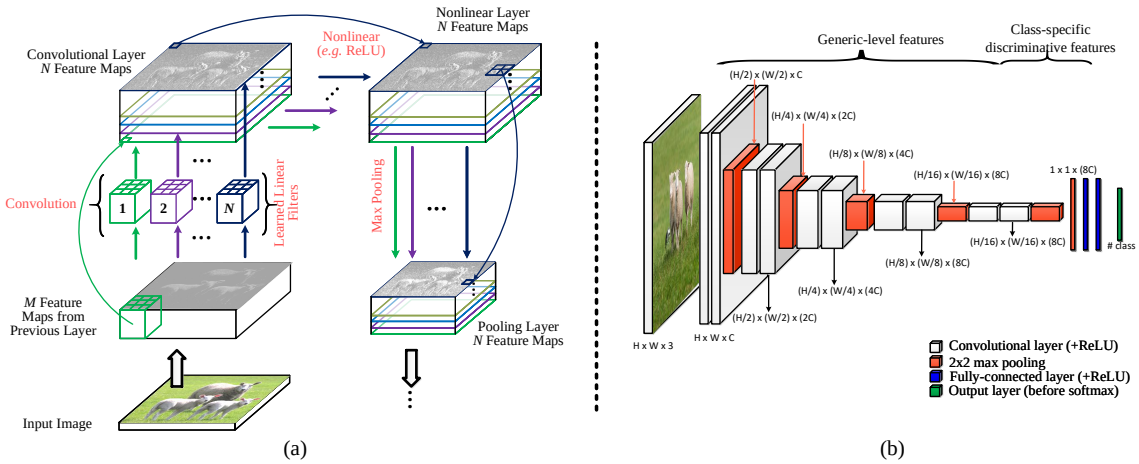


Figure 2.5: (a) Illustration of three operations that are repeatedly applied by a typical CNN [79]. (b) Generic framework of CNN.

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 and contain more semantic-aware information to support higher accuracy, yet this higher accuracy is often at the expense of efficiency. Feature enhancement methods, like hash learning, are one way to tackle this issue [50, 77], however hashing learning needs to carefully consider the loss function design, such as quantization loss [41], to obtain optimal codes for high retrieval accuracy.

2.2 Deep Convolutional Neural Networks

2.2.1 A brief introduction to deep learning

Deep learning depends on neural networks to learn features. Deep neural networks have various variants. Among them, convolutional neural networks (CNNs) are used for vision tasks. There are three types of layer in CNNs: convolutional layer, pooling layer, and fully-connected layer [79]. The convolutional layer plays a vital role in the way CNNs work, emphasizing the use of shared and learnable 2D linear filters. As illustrated in Figure 2.5(a), when a filter glides through the M feature maps from the previous layer $l-1$ each time, the outputs of the convolutions for the next layer l are calculated with its parameters $\boldsymbol{\theta}$, that includes weights $\boldsymbol{w}$ and bias $\boldsymbol{b}$:

$$\boldsymbol{x}^l = \sum_{i=1}^{M^{l-1}} \left(\boldsymbol{w}_i \boldsymbol{x}_i^{l-1} \; + \; \boldsymbol{b} \right) \tag{2.1}$$

It is important to impose a non-linear activation function $\sigma(\cdot)$ (*e.g.* ReLU) on the feature maps $\boldsymbol{x}^l$. Finally, the outputs of non-linear function is stored as inputs

for the next layer l. Usually, the number of filters applied in the previous layer determines the number of produced feature maps in the next layer. As illustrated in Figure 2.5(a), the N filters produce N feature maps.

Finally, the difference between the predictive logits of the classifier and the ground-truth label is used to compute gradients to train the network. Take supervised training as an example, a ground-truth label y^j is assigned to an input x^j, the loss function for network $f(\cdot, \boldsymbol{\theta})$ can then be formulated as:

$$J(\boldsymbol{\theta}) \;=\; \textstyle\sum_j L\big(f(x^j; \boldsymbol{\theta}),\; y^j\big) \tag{2.2}$$

During training, the gradients are computed according to the loss function $J(\boldsymbol{\theta})$ and are back-propagated to $f(\cdot, \boldsymbol{\theta})$, aiming at learning the optimal parameters $\boldsymbol{\theta}^*$:

$$\boldsymbol{\theta}^* = \underset{\boldsymbol{\theta}}{argmin}\; J(\boldsymbol{\theta}) \tag{2.3}$$

A convolutional layer represents local feature learning and yields generic features [44], as shown in Figure 2.5(b). Specifically, the first convolutional layer learns low-level features, such as edges and simple textures. Later intermediate convolutional layers learn middle-level features, such as more complex textures. The deeper convolutional layers learn high-level features, such as objects or parts of objects. Differently, the fully-connected layer, with its larger receptive field, yields global features, which usually are abstract and useful for category-specific discrimination.

The hierarchical structure of CNNs makes it successful in various computer vision tasks. Its feature learning capacity is improved significantly by stacking more convolutional layers, using different filter sizes, or concatenating more convolution operations. Among these DCNNs, there are four models that are widely used as backbone nets for image retrieval.

2.2.2 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 [12], VGG [61], GoogLeNet [62], and ResNet [13].

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. These 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 [10].

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 the reader to recent surveys [79, 80]. This chapter focuses on introducing relevant techniques including feature fusion, feature enhancement, and network fine-tuning, based on popular DCNN backbones for performing image retrieval.

2.3 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, the topic of interest in this chapter. 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 [3, 44, 81], meaning that models trained for classification do not necessarily extract 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 is more important than the 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 2.3.1) and feature enhancement (Section 2.3.2).

2.3.1 Deep feature extraction

2.3.1.1 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 [82].

The fully-connected layer has a global receptive field. After normalization and dimensionality reduction, these features are used for direct similarity measurement without further processing and admitting efficient search strategies [29, 43, 50].

Using the fully-connected layer lacks geometric invariance and spatial information, and thus the last convolutional layer can be examined instead. The research focus associated with the use of convolutional features is to improve their discrimination, where representative strategies are shown in Figure 2.6. For instance, 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.

b. Multiple feedforward pass methods.

Compared to single-pass schemes, multiple pass methods are more time-consuming [3] 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 [29, 83] or spatial pyramid model [48], as illustrated in Figure 2.7. However, these patch detection methods lack retrieval efficiency for large-scale datasets since irrelevant patches are also fed into deep networks, thus it is necessary to analyze image patches [31]. As an example, Cao *et al.* [84] 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. Region proposals can be generated using object detectors, such as selective search [85] and edge boxes [86]. Aside from using object detectors, region proposals can also be learned using deep networks, such as region proposal networks (RPNs) [17, 54] and convolutional kernel networks (CKNs) [87], and then to apply these deep networks into end-to-end fine-tuning scenarios for learning similarity [88, 89].

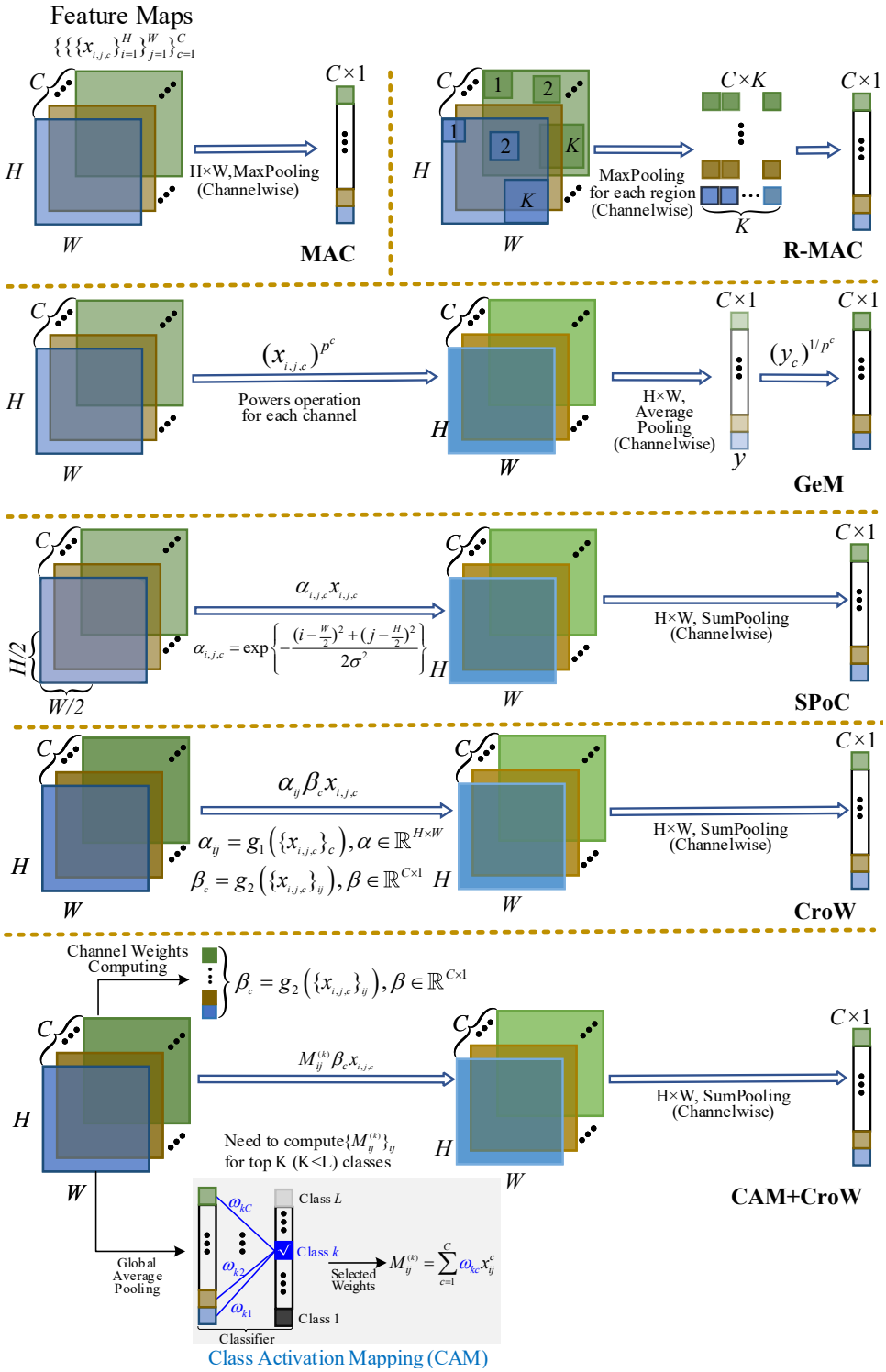


Figure 2.6: Representative methods in single feedforward frameworks, focusing on convolutional feature maps x: MAC [63], R-MAC [31], GeM pooling [57], SPoC with the Gaussian weighting scheme [25], CroW [26], and CAM+CroW [45]. Note that $g_1(\cdot)$ and $g_2(\cdot)$ represent spatial-wise and channel-wise weighting functions, respectively.

2.3.1.2 Deep feature selection

a. Extracted from fully-connected layers

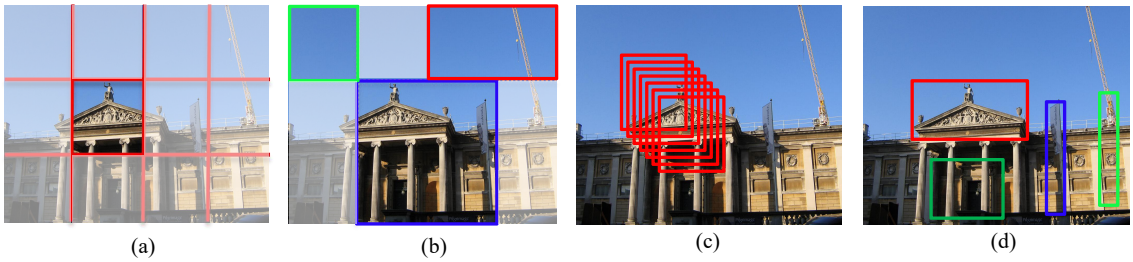


Figure 2.7: Image patch generation schemes: (a) Rigid grid; (b) Spatial pyramid modeling (SPM); (c) Dense patch sampling; (d) Region proposals (RPs) from region proposal networks.

It is straightforward to select a fully-connected layer as a feature extractor [29, 43, 50, 64]. With PCA dimensionality reduction and normalization [43], images' similarity can be measured. Only the fully-connected layer may limit the overall retrieval accuracy, Jun *et al.* [64] concatenate features from multiple fully-connected layers, and Song *et al.* [88] 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 [64].

For the first limitation, researchers focus on the inputs of networks, *i.e.*, using multiple feedforward passes [43]. 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. For this, several works introduce methods to leverage intermediate convolutional layers [25, 29, 63].

b. Extracted from convolutional layers

Features from convolutional layers (usually the last one) preserve more structural details which are especially beneficial for instance-level retrieval [63]. The neurons in a convolutional layer are connected only to a local region of the input feature maps. The smaller receptive field ensures that the produced features preserve more local structural details and are more robust to image transformations like truncation and occlusion [25]. Usually, the robustness of features is improved after pooling.

A convolutional layer arranges the spatial information well and produces location-adaptive features [90]. Various image retrieval methods use convolutional layers as local detectors [25, 30, 31, 45, 63, 90]. For instance, Razavian *et al.* [63] make the first attempt to perform spatial max pooling on the feature maps of an off-the-shelf DCNN model; Babenko *et al.* [25] propose sum-pooling convolutional features

(SPoC) to obtain compact descriptors pre-processed with a Gaussian center prior (see Figure 2.6). Ng *et al.* [90] explore the correlations between activations at different locations on the feature maps, thus improving the final feature descriptor. Yue *et al.* [30] replace BoW [7] with VLAD [28], and are the first to encode local features into VLAD representations. This idea inspired another milestone work [55] 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.

2.3.1.3 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 [64]: For instance, Yu *et al.* [91] 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.* [64] 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 (global features) and features from convolutional layers (local features) can complement each other when measuring semantic similarity and can, to some extent, guarantee retrieval performance [92].

Global features and local features can be concatenated directly [92, 93]. 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.* [65] propose a Multi-layer Orderless Fusion (MOF) approach, which is inspired by Multi-layer Orderless Pooling (MOP) [29] 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.* [92] 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.

It is worth introducing a scheme to explore *which* layer combination is better for fusion given their differences of extracting features. For instance, Chatfield *et al.* [75] demonstrate that fusing convolutional layers and fully-connected layers outperforms the methods that fuse convolutional layers only. In the end, fusing two convolutional layers with one fully-connected layer achieves the best performance.

b. Model-level fusion

It is possible to combine features on different models; such fusion focuses on model complementarity to achieve improved performance, categorized into *intra-model* and *inter-model*.

Generally, intra-model fusion suggests multiple deep models having a similar structure, while inter-model fusion involves models with more differing structures. For instance, Simonyan *et al.* [61] 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, Ding *et al.* [94] propose a selective deep ensemble 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.* [95] 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 networks have different receptive fields [48, 68, 96, 97, 98]. For instance, a two-stream attention network [68] is introduced 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 [99]. Considering the importance and necessity of inter-model fusion to bridge the gap between mid-level and high-level features, Liu *et al.* [48] combine VGG-19 and AlexNet to learn combined features, while Ozaki *et al.* [97] 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.* [98] combine ResNet and Inception V1 [62] 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 fuse all types of features from the candidate models and then learning a metric based on the concatenated features [68], 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 [49], 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.* [98] illustrate the effect of combining different numbers of features and different sizes within the ensemble. Chen *et al.* [100] 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.

2.3.2 Deep feature enhancement

2.3.2.1 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 widely used methods applied on convolutional feature maps. Sum/average pooling is less discriminative, because it considers all activated outputs from a convolutional layer, as a result it weakens the effect of highly activated features [46]. 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 [101].

Convolutional features can be directly aggregated to produce global ones by spatial pooling. For example, Razavian *et al.* [63, 83] apply max pooling on the convolutional features for retrieval. Babenko *et al.* [25] leverage sum pooling with a Gaussian weighting scheme to encode convolutional features (*i.e.* SPoC). Note that this operation usually is followed by L2 norm and PCA dimensionality reduction.

As an alternative to the holistic approach, it is also possible to pool some regions in a feature map [25, 63], such as done by R-MAC [31]. 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.

2.3.2.2 Feature embedding

Apart from direct pooling or regional pooling, it is possible to embed the convolutional features into a high dimensional space to obtain compact ones. The widely used methods include BoW, VLAD, and FV. The embedded features' dimensionality can be reduced using PCA. Note that BoW and VLAD can be extended by using other metrics, such as Hamming distance [102]. Here we briefly describe the principle of the embedding methods for the case of Euclidean distance metric.

BoW [7] is a widely adopted encoding method. BoW encoding leads to a sparse vector of occurrence. Specifically, 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 $\vec{c}_k$, one can count and normalize the number of occurrences by

$$g(\vec{c}_k) = \frac{1}{T} \sum_{t=1}^{T} \phi(\vec{x}_t, \vec{c}_k) \tag{2.4}$$

$$\phi(\vec{x}_t, \vec{c}_k) = \begin{cases} 1 & \text{if } \vec{c}_k \text{ is the closest codeword for } \vec{x}_t \\ 0 & \text{otherwise} \end{cases} \tag{2.5}$$

Thus BoW considers the number of descriptors belonging to each codebook $\vec{c}_k$ (*i.e.* 0-order feature statistics), then BoW representation is the concatenation of all mapped vectors:

$$G_{_{BoW}}(\vec{X}) = \left[\ g(\vec{c}_1), \cdots, g(\vec{c}_k), \cdots, g(\vec{c}_K)\ \right]^\top \tag{2.6}$$

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 [65, 82]. However, the embedded vectors are high dimensional and sparse, which are not well suited to large-scale datasets in terms of efficiency.

VLAD [28] 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 $(\vec{x}_t - \vec{c}_k)$:

$$g(\vec{c}_k) = \frac{1}{T}\sum_{t=1}^{T} \phi(\vec{x}_t, \vec{c}_k)(\vec{x}_t - \vec{c}_k) \tag{2.7}$$

where $\phi(\vec{x}_t, \vec{c}_k)$ as defined in (2.5). Finally, the VLAD representation is stacked by the residuals for all centroids, with dimension $(D \times K)$, *i.e.*

$$G_{_{VLAD}}(\vec{X}) = \left[\ \cdots, g(\vec{c}_k)^\top, \cdots\ \right]^\top. \tag{2.8}$$

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.* [30] embed the feature maps from the last convolutional layer into VLAD representations, which is proved to have higher effectiveness than BoW.

The FV method [8] 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 [103]. 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

$$\gamma_k(\vec{x}_t) = w_k \times p_k(\vec{x}_t) / \Big(\sum_{j=1}^{K} w_j p_j(x_t)\Big) \quad s.t. \sum_{j=1}^{K} w_k = 1 \tag{2.9}$$

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

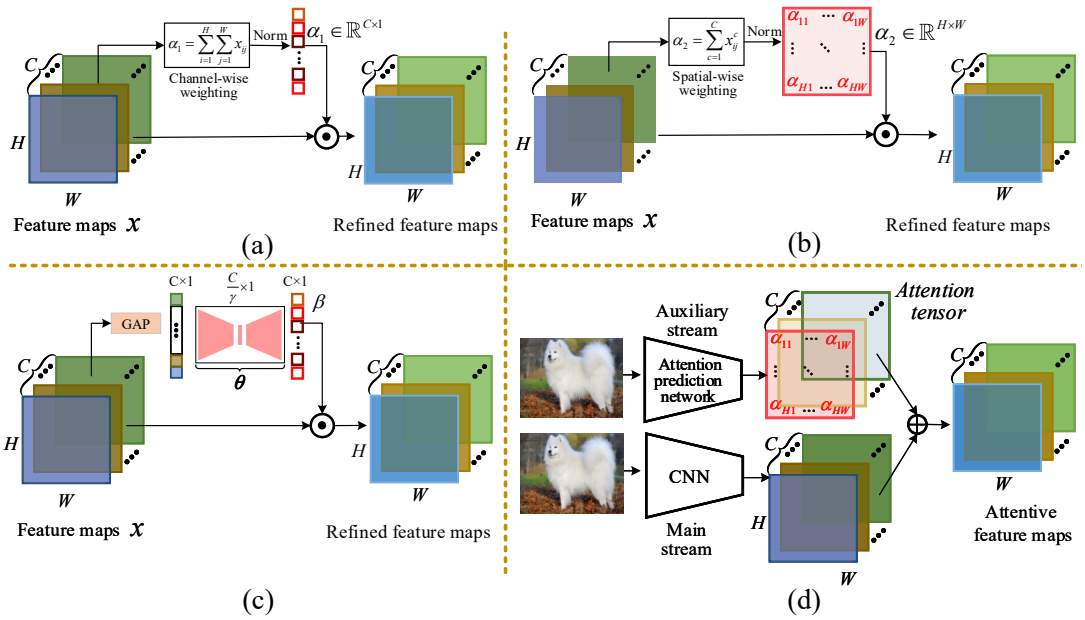


Figure 2.8: Attention mechanisms are shown, divided into two categories. (a)-(b) Non-parametric mechanisms: The attention is based on convolutional feature maps x with size $H \times W \times C$. Channel-wise attention in (a) produces a C-dimensional importance vector α_1 [26, 47]. Spatial-wise attention in (b) computes a 2-dimensional attention map α_2 [26, 45, 74, 90]. (c)-(d) Parametric mechanisms: The attention weights β are provided by a sub-network with trainable parameters (*e.g.* θ in (c)) [105, 106]. Likewise, some off-the-shelf models [99, 107] can predict the attention maps from the input image directly.

in the dictionary, which is computed as

$$
\begin{aligned}
g_{w_k} &= \frac{1}{T\sqrt{w_k}} \sum_{t=1}^{T} \Big(\gamma_k(\vec{x}_t) - w_k \Big) \\
g_{u_k} &= \frac{\gamma_k(\vec{x}_t)}{T\sqrt{w_k}} \sum_{t=1}^{T} \left(\frac{\vec{x}_t - \mu_k}{\sigma_k} \right), \\
g_{\sigma_k^2} &= \frac{\gamma_k(\vec{x}_t)}{T\sqrt{2w_k}} \sum_{t=1}^{T} \left[\left(\frac{\vec{x}_t - \mu_i}{\sigma_k} \right)^2 - 1 \right]
\end{aligned}
\tag{2.10}
$$

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

$$
G_{FV}(\vec{X}) = \left[\; g_{w_1}, \cdots, g_{w_K}, g_{u_1}, \cdots, g_{u_K}, g_{\sigma_1^2}, \cdots, g_{\sigma_K^2} \; \right]^\top \tag{2.11}
$$

The FV representation defines a kernel from a generative process and captures more statistics than BoW and VLAD. FV representations do not increase computational costs significantly but require more memory. Applying FV without memory controls may lead to suboptimal performance [104].

Discussion. Traditionally, sum pooling and max pooling are directly plugged into deep networks and the whole model is used in an end-to-end way, whereas the em-

bedding methods, including BoW, VLAD, and FV, are initially trained separately with pre-defined vocabularies [48, 108]. For these three methods, one needs to pay attention to their properties before choosing one of them to embed 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. Further, when any one of these methods is used, it is necessary to integrate them as a "layer" of deep networks 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 [30]. This idea led to a follow-up approach, NetVLAD [55], where deep networks are fine-tuned with the VLAD vector.

2.3.2.3 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 2.8, 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 2.8(a,b). For the spatial-wise pooling of Figure 2.8(b), Kalantidis *et al.* [26] 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.* [90] 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 2.8(a) are also popular non-parametric attention mechanisms. Xu *et al.* [47] rank the weighted feature maps to build the "probabilistic proposals" to further select regional features. Jimenez *et al.* [45] combine CroW and R-MAC to propose Classes Activation Maps (CAM) to weight feature maps for each class. Qi *et al.* [66] 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 2.8(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.* [105] propose stacked fully-connected layers to learn an attention model for multi-scale image patches. Similarly, Noh *et al.* [106] design a 2-layer CNN with a softplus output layer to compute scores which indicate the importance of different image regions.

Inspired by R-MAC, Kim *et al.* [109] 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 [67] explore different saliency models, including DeepFixNet [99] and Saliency Attentive Model (SAM) [107], to learn salient regions for input images. Similarly, Yang *et al.* [68] 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 [90, 95, 110].

2.3.2.4 Deep hash embedding

Real-valued features extracted by deep networks are typically high-dimensional, and therefore are not well-satisfied 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 [77, 111].

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 images are embedded as close as possible, and the hash codes of dissimilar images are as separated as possible. A hash function $h(\cdot)$ for binarizing features of an image x may be formulated as

$$b_k = h(x) = h\big(f(x;\boldsymbol{\theta})\big) \quad k = 1, \ldots, K \tag{2.12}$$

then an image can be represented by the generated hash codes $\boldsymbol{b} \in \{+1, -1\}^K$. Because hash codes are non-differentiable their optimization is difficult, so $h(\cdot)$ can be relaxed to be differentiable by using tanh or sigmoid functions [77].

When binarizing real-valued features, it is crucial (1) to preserve image similarity and (2) to improve hash code quality [77]. 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 loss function can significantly influence similarity preservation, which includes both supervised and unsupervised approaches. With the class label avail-

able, 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 [112]. 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 [111]. Similarly, Kang *et al.* [113] 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 [114], triplet loss [51, 115, 116], and adversarial learning [115, 117], 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 [118], KNN graphs [69, 72], k-means algorithms [70], and network structures such as AutoEncoders [119, 120, 121] and generative adversarial networks [60, 69, 122, 123].

Separate from the loss function, it is also important to design deep network frameworks for learning. For instance, Long *et al.* [116] 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.* [117] present coined Pair Conditional WGAN, an extension of Wasserstein generative adversarial networks, to generate more samples conditioned on the similarity information.

b. Improving hash function quality

Improving hash function 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 [77]. Bit uncorrelation implies that different bits are as independent as possible and have little redundancy of information, so that a given set of bits can aggregate more information within a given code length. In principle, bit uncorrelation can be formulated as $\boldsymbol{b}\boldsymbol{b}^\top = \boldsymbol{I}$ in which $\boldsymbol{I}$ is an identity matrix of size K. For example, it can be encouraged via regularization terms such as orthogonality [124] and mutual information [125]. Bit balance means that each bit should have a 50% chance of being +1 or -1, thereby maximizing code variance and information [77]. Mathematically, this condition is constrained by using this regularization term $\boldsymbol{b} \cdot \mathbf{1} = 0$ where $\mathbf{1}$ is a K-dimensional vector with all elements equal to 1.

2.4 Retrieval via Learning DCNN Representations

In Section 2.3, 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

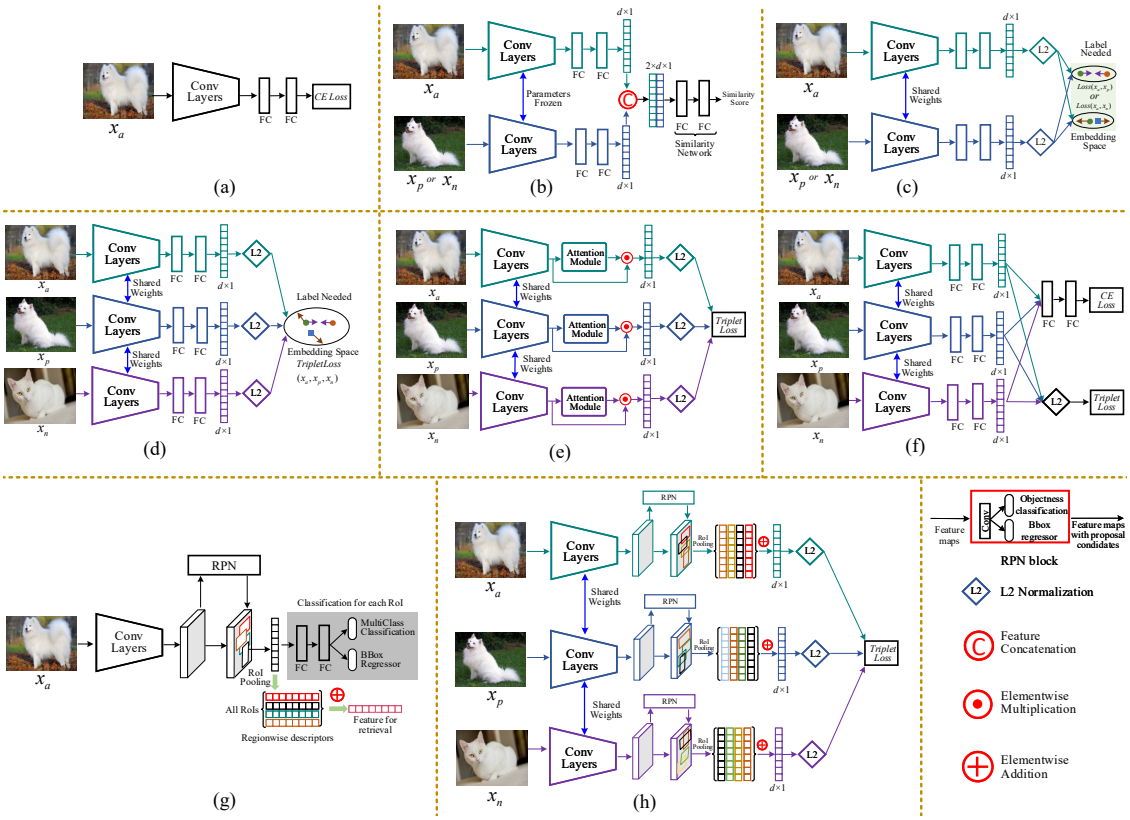


Figure 2.9: Schemes of supervised fine-tuning. Anchor, positive, and negative images are indicated by x_a, x_p, x_n, respectively. (a) classification-based; (b) using a transformation matrix for learning the similarity of image pairs; (c) Siamese networks; (d) triplet loss for fine-tuning; (e) an attention block into DCNNs to highlight regions; (f) combining classification-based and verification-based loss for fine-tuning; (g) region proposal networks (RPNs) to locate the RoI and highlight specific regions or instances; (h) inserting the RPNs of (g) into DCNNs, such that the RPNs extract regions or instances at the convolutional layer.

the strategies which were discussed. In order for models to have higher scalability and to be more effective for retrieval, a common practice is network fine-tuning, *i.e.* updating the pre-stored parameters [44, 78]. However fine-tuning does not contradict or render irrelevant feature processing methods of Section 2.3; 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.

2.4.1 Supervised fine-tuning

2.4.1.1 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 shown in Figure 2.9(a), by optimizing its parameters on the basis of a cross entropy loss L_{CE}:

$$L_{CE}(\hat{p}_i, y_i) = -\sum_{i}^{c}\Big(y_i \times log(\hat{p}_i)\Big) \tag{2.13}$$

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 [50], 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 [126], Oxford-5k, and Oxford-105k [127]. The newly-updated layers are used as global or local feature detectors for image retrieval.

A classification-based fine-tuning method improves the *model-level* adaptability for new datasets, which, to some extent, has mitigated the issue of model transfer for image retrieval. 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.

2.4.1.2 Verification-based fine-tuning

With affinity information indicating similar and dissimilar 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 [27]:

1. A pair-wise constraint, corresponding to a Siamese network as in Figure 2.9(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 2.9(e), in which anchor images are paired with both similar and dissimilar samples [27].

These verification-based learning methods are categorized into globally supervised approaches (Figure 2.9(c,d)) and locally supervised approaches (Figure 2.9(g,h)), 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

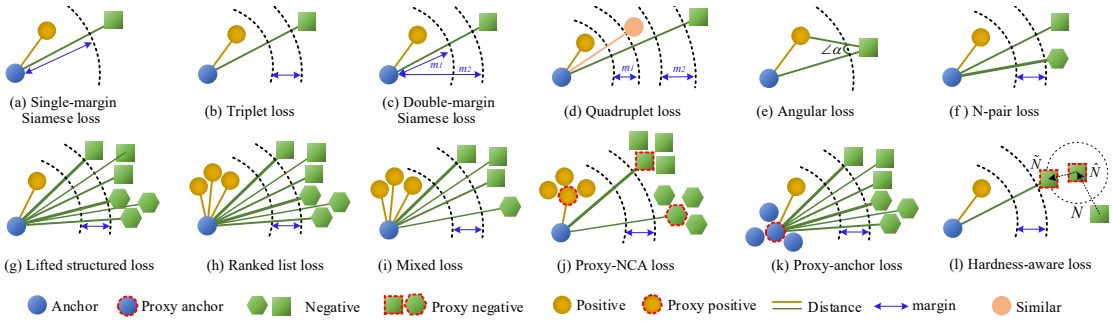


Figure 2.10: Illustrations of sample mining strategies in metric learning. Here, we illustrate three classes, where shapes indicate different classes. Multiple pairs are considered in some loss terms and assigned with distinct weights during training, indicated by different line width. (a)-(c) have been introduced in the text. (d) Quadruplet loss [128]: a sample similar to the anchor is used to construct a double margin. (e) Angular loss [129]: the angle at the negative of triple triangles is computed to obtain higher order geometric constraints. (f) N-pair loss [130]: a positive sample is identified from $N-1$ negative samples of N-1 classes. (g) Lifted structured loss [131]: the structure relationships of three positive and three negative samples are considered. (h) Ranked list loss [132]: all samples to explore intrinsic structured information are considered. (i) Mixed loss [133]: three positive and three negative samples are captured which are initially closely distributed, where another anchor-negative pair initially lies very close to the anchor. (j) Proxy-NCA loss [134]: proxy positive and negative samples for each class are computed and trained with a true anchor sample. (k) Proxy-anchor loss [135]: the anchor sample is represented by a proxy. (l) Hardness-aware loss [136]: the synthetic negative is mapped from an existing hard negative, the hard levels manipulated adaptively within a certain range.

are extracted by a network $f(\cdot)$ with parameters $\boldsymbol{\theta}$, for which we can represent the affinity information for each similar or dissimilar pair as

$$D_{ij} = D(x_i, x_j) = ||f(x_i;\boldsymbol{\theta}) - f(x_j;\boldsymbol{\theta})||_2^2 \tag{2.14}$$

a. Refining with transformation matrix.

Learning the similarity among the input samples can be implemented by optimizing the weights of a linear transformation matrix [52]. It transforms the concatenated feature pairs into a common latent space using a transformation matrix $\boldsymbol{W}\in\mathbb{R}^{2d\times 1}$, where d is the feature dimension. The similarity score of these pairs are predicted via a sub-network $S_W(x_i, x_j) = f_W(f(x_i;\boldsymbol{\theta}) \cup f(x_j;\boldsymbol{\theta}); \boldsymbol{W})$ [52, 137]. 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:

$$L_W(x_i, x_j) = |S_W(x_i, x_j) - S_{ij}\big(sim(x_i, x_j) + m\big) - (1 - S_{ij})\big(sim(x_i, x_j) - m\big)| \tag{2.15}$$

where $sim(x_i, x_j)$ can be the cosine function for guiding training $\boldsymbol{W}$ and m is a

margin. 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 2.9(b) where the weights of the two DCNNs are not necessarily shared.

b. Fine-tuning with Siamese networks.

Siamese networks represent important options in implementing metric learning for fine-tuning, as shown in Figure 2.9(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) such that $S(x_i, x_j) \in \{0, 1\}$. A Siamese loss function, illustrated in Figure 2.10(a), is formulated as

$$L_{Siam}(x_i, x_j) = \frac{1}{2}S(x_i, x_j)D(x_i, x_j) + \frac{1}{2}\big(1 - S(x_i, x_j)\big) \max\big(0,\ m - D(x_i, x_j)\big) \tag{2.16}$$

A standard Siamese network and Siamese loss are used to learn the similarity between semantically relevant samples under different scenarios. For example, Simo *et al.* [138] 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.* [53] leverage the Siamese network to learn image features which are then fed into the Fisher Vector model for further encoding. In addition, Siamese networks can also be applied to hashing learning in which the Euclidean distance formulation $D(\cdot)$ in Eq. 2.16 is replaced by the Hamming distance [114].

c. Fine-tuning with triplet networks.

Triplet networks [137] optimize similar and dissimilar pairs simultaneously. As shown in Figure 2.9(d) and Figure 2.10(b), the plain triplet networks adopt a ranking loss for training:

$$L_{Triplet}(x_a, x_p, x_n) = \max\big(0, m + D(x_a, x_p) - D(x_a, x_n))\big) \tag{2.17}$$

which indicates that the distance of an anchor-negative pair $D(x_a, x_n)$ should be larger than that of an anchor-positive pair $D(x_a, x_p)$ by a certain margin m. The triplet loss is used to learn fine-grained image features [71, 96] and for constraining hash code learning [51, 115, 116].

To focus on specific regions or objects, local supervised metric learning has been explored [58, 89, 139, 140]. In these methods, some regions or objects are extracted using region proposal networks (RPNs) [17] which subsequently can be plugged into deep networks and trained in an end-to-end manner, such as shown in Figure 2.9(g), in which Faster R-CNN [17] is fine-tuned for instance search [89]. RPNs yield the regressed bounding box coordinates of objects and are trained by the multi-class classification loss. The final networks extract better regional features by RoI pooling and perform spatial ranking for instance retrieval.

RPNs [17] enable deep models to learn regional features for particular instances or objects [54, 140]. RPNs used in the triplet formulation are shown in Figure 2.9(h). 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 RPN loss and triplet loss leads to unstable results. This is addressed in [54] by first training a CNN to produce R-MAC using a rigid grid, after which the parameters in convolutional layers are fixed and RPNs are trained to replace the rigid grid.

Attention mechanisms can also be combined with metric learning for fine-tuning [110, 139], as in Figure 2.9(e), where the attention module is typically end-to-end trainable and takes as input the convolutional feature maps. For instance, Song *et al.* [139] introduce a convolutional attention layer to explore spatial-semantic information, highlighting regions in images to significantly improve the discrimination for inter-class and intra-class features for image retrieval.

Recent studies [64, 93] have jointly optimized the triplet loss and classification loss function, as shown in Figure 2.9(f). Fine-tuned models that use only a triplet constraint may possess inferior classification accuracy for similar instances [93], since the classification loss does not predict the intra-class similarity, rather locates the relevant images at different levels. Given these considerations, it is natural to combine and optimize triplet constraint and classification loss jointly [64]. The overall joint function is formulated as

$$L_{Joint} = \alpha \cdot L_{Triplet}(x_{i,a}, x_{i,p}, x_{i,n}) + \beta \cdot L_{CE}(\hat{p}_i, y_i) \tag{2.18}$$

where the cross-entropy loss (CE loss) L_{CE} is defined in Eq. (2.13) and the triplet loss $L_{Triplet}$ in Eq. (2.17). α and β are trade-off hyper-parameters to tune the two loss functions.

An implicit drawback of the Siamese loss in Eq. 2.16 is that it may penalize similar image pairs even if the margin between these pairs is small or zero, which may degrade performance [141], since the constraint is too strong and unbalanced. At the same time, it is hard to map the features of similar pairs to the same point when images contain complex contents or scenes. To tackle this limitation, Cao *et al.* [142] adopt a double-margin Siamese loss [141], illustrated in Figure 2.10(c), to relax the penalty for similar pairs. Specifically, 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

$$\begin{aligned} L(x_i, x_j) = \frac{1}{2} S(x_i, x_j) \max\big(0, D(x_i, x_j) - m_1\big) + \\ \frac{1}{2}\big(1 - S(x_i, x_j)\big) \max\big(0, m_2 - D(x_i, x_j)\big) \end{aligned} \tag{2.19}$$

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 when the distance of a similar pair is larger than m_1. The mAP metric of retrieval is improved when using the double margin Siamese loss [141].

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.

A variety of loss functions have been proposed recently [128, 130, 131, 132, 134]. Some of these use more samples or additional constraints. For example, Chen *et al.* [128] incorporate Quadruplet samples for constraining relationships between anchor, positive, negative, and similar images. The N-pair loss [130] and the lifted structured loss [131] even define constraints on all images and employ the structural information of samples in a mini-batch.

The sampling strategy can greatly affect the feature learning and training convergence. To date, many sampling strategies such as clustering have been introduced, of which 12 are shown in Figure 2.10. Aside from sampling within a mini-batch, other work explores mining 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.* [143] 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.* [144] 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 a network. However, the possible additional computational cost during training is a core issue to be addressed.

Directly optimizing the average precision (AP) metric using the listwise AP loss [145] is one way to consider a large number of image simultaneously. Training with this loss has been demonstrated to improve retrieval performance [145, 146, 147], 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 [145, 146] or sigmoid function [147].

2.4.2 Unsupervised fine-tuning

Supervised network fine-tuning becomes infeasible when there is not enough supervisory information because such information is costly to assemble or unavailable. Given these limitations, unsupervised fine-tuning methods for image retrieval are quite necessary but less studied [148].

For unsupervised fine-tuning, two broad directions are to mine relevance among features via manifold learning to obtain ranking information, and to devise novel unsupervised frameworks (*e.g.* AutoEncoders), each discussed below.

2.4.2.1 Mining samples with manifold learning

Manifold learning focuses on capturing intrinsic correlations on the manifold structure to mine or deduce revelance, as illustrated in Figure 2.11. Initial similarities between the original extracted features are used to construct an affinity matrix, which is then re-evaluated and updated using manifold learning [149]. According to the manifold similarity in the updated affinity matrix, positive and hard negative samples are selected for metric learning using verification-based loss functions such as pair loss [58, 150], triplet loss [151, 152], or N-pair loss [148], *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 of the manifold of deep features, generally involving two steps [149] known as a diffusion process. First, the affinity matrix (Figure 2.11) 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 [59, 150, 151, 152]. Some new similarity diffusion methods have recently been proposed, like the regularized diffusion process (RDP) [153] and the regional diffusion mechanism [150]. For more details on diffusion methods we refer to the survey [149].

Most existing algorithms follow a similar principle (*e.g.* random walk [149]). The differences among methods lie primarily in three aspects:

1. **Similarity initialization**, which affects the subsequent KNN graph construction in an affinity matrix. Usually, an inner product [59, 148] or Euclidean distance [56] is directly computed for the affinities. A Guassian kernel function can be used for affinity initialization [149, 152] or Iscen *et al.* [150] consider regional similarity from image patches to build the affinity matrix.

2. **Transition matrix definition**, a row-stochastic matrix [149], determines the probabilities of transiting from one node to another in the graph. These probabilities are proportional to the affinities between nodes, which can be measured by Geodesic distance (*e.g.* the summation of weights of relevant edges).

3. **Iteration scheme**, to re-valuate and update the values in affinity matrix by the manifold similarity until some kind of convergence is achieved. Most algorithms are iteration-based [149, 151], as illustrated in Figure 2.11.

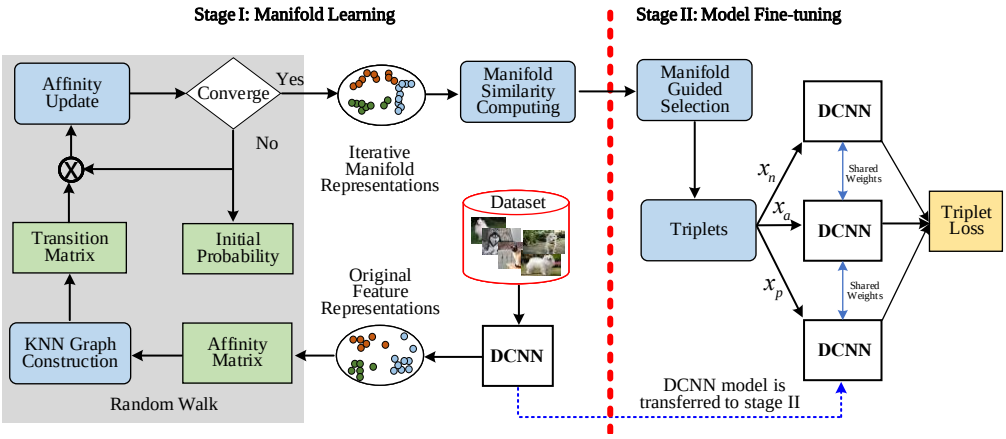


Figure 2.11: Paradigm of manifold learning for unsupervised metric learning, based on triplet loss.

Diffusion process algorithms are indispensable for unsupervised fine-tuning. Better image similarity is guaranteed when it is improved based on initialization (*e.g.* regional similarity [150] or high order information [56]). However, the diffusion process requires more computation and searching due to the iteration scheme [152], a limitation which cannot meet the efficiency requirements of image retrieval. To mitigate this, Nicolas *et al.* [148] 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 [58], which is derived from graph convolution networks [154]. Their end-to-end framework allows efficient computation during the training and testing stages.

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

It is possible to explore proximity information, to cluster in Euclidean space, splitting the training set into different groups. For example, Tzelepi *et al.* [155] explore a fully unsupervised fine-tuning method by clustering, in which the kNN algorithm is used to compute the k nearest features, then fine-tuned to minimize the squared distance between each query feature and its k nearest features. As a second example, Radenovic *et al.* [32, 57] use Structure-from-Motion (SfM) for clustering to explore sample reconstructions to select images for triplet loss. Clustering methods depend on the Euclidean distance, making it difficult to reveal the intrinsic relationship between objects.

2.4.2.2 AutoEncoder-based frameworks

An AutoEncoder is a kind of neural network that aims to reconstruct its output as closely as possible to 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. The encoder and decoder can be both be convolutional neural networks.

In an AutoEncoder, there exist different levels (*e.g.* pixel-level or instance-level) of reconstruction. These different reconstructions affect the effectiveness of an AutoEncoder, in that pixel-level reconstructions may degrade the learned features of an encoder by focusing on trivial variations in a reconstructed image, since natural images typically contains many detailed factors of location, color, and pose.

An AutoEncoder is an optional framework for supporting other methods, for example the implementation of unsupervised hash learning [60, 119, 120, 121]. Except for the reconstruction loss [60, 121], it is highly necessary to mine feature relevance to explore other objective functions. This is usually realized by using clustering algorithms [121] since features from an off-the-shelf network initially contain rich semantic information to keep their semantic structure [69, 72, 118]. For example, Gu *et al.* [121] 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.* [72] and Deng *et al.* [69] 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 Nets are also used to predict sample similarity, such as in the work of Yang *et al.* [118], which adopts a Bayes optimal classifier to assign semantic similarity labels to data pairs which have a higher similarity probability.

AutoEncoders can also be integrated into other frameworks, such as graph convolutional networks [154] and object detection models [156] to learn better binary latent variables. For example, Shen *et al.* [60] combine graph convolutional networks [154] 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 [60, 69, 122, 123]. 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 [122]. Some other losses are employed in GANs to enhance hash code learning. For instance, a distance matching regularizer is utilized to propagate the correlations between high-dimensional real-valued features and low-dimensional hash codes [157], or two loss functions that aim at promoting independence of binary codes [123]. In summary, using GANs for unsupervised hash learning is promising, but there remains much room for further exploration.

2.5 State of the Art Performance

2.5.1 Datasets

To demonstrate the effectiveness of methods, we choose four commonly-used datasets for performance comparison: Holidays, Oxford-5k (including the extended Oxford-105k), Paris-6k (including the extended Paris-106k) and UKBench.

UKBench (UKB) [158] consists of 10,200 images of objects. The whole dataset has 2,550 groups of images, each group having four images of the same object from different viewpoints or illumination conditions. Each image in the dataset can be used as a query image.

Holidays [126] 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 [127] consists of 5,062 images for 11 Oxford buildings. Each image is represented by five 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 [159] includes 6,412 images collected from Flickr. It is categorized into 12 groups about specific Paris architectures. The dataset has 500 query images for evaluation, and 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 [160]. Due to the popularity of Oxford-5k and Paris-6k, we primarily undertake performance evaluations on the original datasets.

2.5.2 Evaluation metrics

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

$$AP = \frac{\sum_{k=1}^{N} P(k) \cdot rel(k)}{R} \tag{2.20}$$

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

query images,

$$\frac{1}{Q}\sum_{q=1}^{Q} AP(q) \tag{2.21}$$

where Q is the number of query images.

Additionally, N-S score is a metric used for UKBench [158]. In this dataset, there are four relevant images for each query. The N-S score is the average, four times, for the top-four precision over the dataset.

2.5.3 Performance comparison and analysis

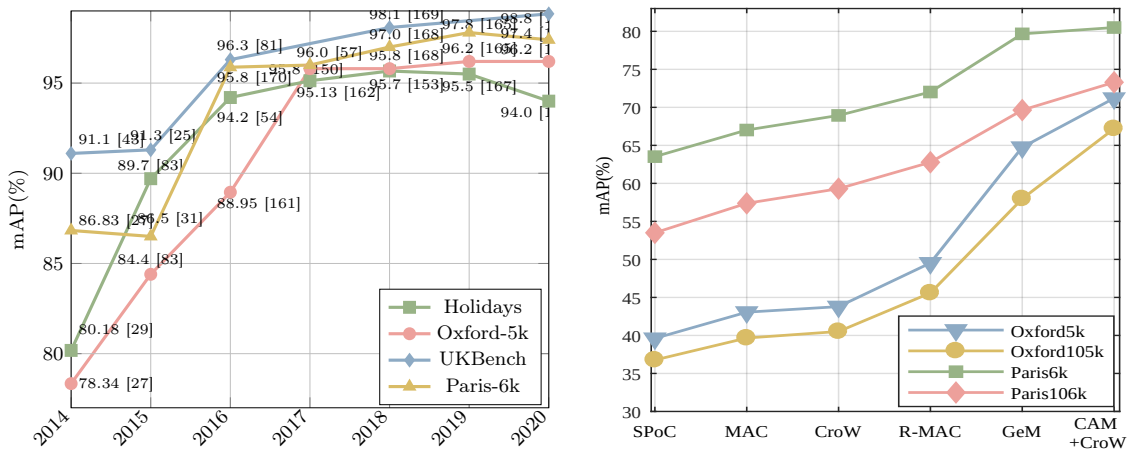
Overview. We conclude with the performance over these 4 datasets from 2014 to 2020 in Figure 2.12(a). At early period, DCNNs acted as powerful extractors and achieved good results, *e.g.* mAP is 78.34% in [27] on Oxford-5k. Subsequently, the results increased significantly when some crucial factors were adopted, including feature fusion [161, 162, 163], feature aggregation [31, 63], and network fine-tuning [153, 164]. For instance, the accuracy on UKBench reaches an mAP of 98.8% in [163] 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% [27] to 96.2% [165] on the Oxford-5k dataset when manifold learning is used to fine-tune deep networks.

We evaluate the methods using off-the-shelf models (Table 2.2) and fine-tuning networks (Table 2.3). In Table 2.2, single pass and multiple pass are analyzed, while supervised fine-tuning and unsupervised fine-tuning are compared in Table 2.3.

Evaluation for single feedforward pass. The common practice using this scheme is to enhance feature discrimination. In Table 2.2, we observe that fully-connected layers used as feature extractors may reach a lower accuracy (*e.g.* 74.7% on Holidays in [50]), compared to the counterpart convolutional layers because the fully-connected layers lack structural information. Layer-level feature fusion strategy improves retrieval accuracy. For example, Yu *et al.* [92] combined three layers (*Conv4*, *Conv5*, and *FC6*) (*e.g.* an mAP of 91.4% on Holidays), outperforming the performance of non-fusion method in [25] (*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 2.6 improve the discrimination of convolutional feature maps and perform differently in Table 2.2 (*e.g.* 66.9% of R-MAC [159], 58.9% of SPoC [25] on Oxford-5k). We view this as a critical factors and further analyze.

Evaluation for multiple feedforward pass. The methods exemplified in Figure 2.7 are reported their results in multiple pass scheme. Among them, extracting image patches densely using Overfeat [166] can reach best results on the 4 datasets [43].

Using rigid grid method reach competitive results (*e.g.* an mAP of 87.2% on Paris-6k) [108]. 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 [85], its result on UKBench is 3.81 (N-Score), higher than the one using deeper networks, such as [25, 50, 92]. Besides feeding image patches into the same network, model-level fusion also exploit complementary spatial information to improve the retrieval accuracy. For instance, as reported in [48], which combines AlexNet and VGG, the results on Holidays (81.74% of mAP) and UKBench (3.32 of N-Score) are better than these in [65] (76.75% and 3.00, respectively).



(a) (b)

Figure 2.12: (a) Performance improvement from 2014 to 2020. (b) mAP comparison of the feature aggregation methods shown in Figure 2.6.

Evaluation for supervised fine-tuning. Compared to the off-the-shelf models, fine-tuning deep networks usually improves accuracy, see Table 2.3. For instance, the result on Oxford-5k [31] by using a pre-trained VGG is improved from 66.9% to 81.5% in [53] 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 [50]), its performance may be improved with powerful DCNNs and feature enhancement methods such as the attention mechanism in [106], 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) [89]. Its results are lower than the results from [52] (88.2% and 88.2%, respectively) where a transformation matrix is used to learn

Table 2.1: 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		UKB	
	mAP	Time	mAP	Time	mAP	Time	N-S	Time
[153]	91.3 (88.4)	5.45 *ms* (809 *ms*)	-	-	95.66	3.11 *ms*	3.93	4.91 *ms*
[165]	92.6 (91.8)	2 *ms* (10 *ms*)	-	-	-	-	-	-
[150]†	85.7 (-)	20 *ms* (-)	94.1 (-)	20 *ms* (-)	-	-	-	-
[150]‡	95.8 (-)	600 *ms* (-)	96.9 (-)	700 *ms* (-)	-	-	-	-
[172]	64.9 (58.8)	0.81 *ms* (0.82 *ms*)	-	-	-	-	-	-
[57]	64.8 (57.9)	0.77 *ms* (0.73 *ms*)	-	-	-	-	-	-
[52]	55.5 (-)	0.35 *ms* (-)	71.0 (-)	0.35 *ms* (-)	-	-	-	-

visual similarity. However, when RPN is trained by using triplet loss such as [140], 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.* [53] 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 [32, 55].

Evaluation for unsupervised fine-tuning. Compared to supervised fine-tuning, unsupervised fine-tuning methods are relatively less explored. The difficulty for unsupervised fine-tuning is to mine 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 [171] achieves an mAP 88.4% on Holidays, while unsupervised fine-tuning network using the same loss function in [32, 57, 151] achieve 82.5%, 83.1%, and 87.5%, respectively. However, unsupervised fine-tuning methods can achieve a similar accuracy even outperform the supervised fine-tuning if a suited feature embedding method is used. For instance, Zhao *et al.* [152] explore 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 [140], 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 [106], while the precision is further improved to 92.0% when IME layer is used to embed features and fine-tuned in an unsupervised way [56]. Note that fine-tuning strategies are related to the type of the target retrieval datasets.

Retrieval efficiency is also an important criterion for image retrieval. Deep learning methods are usually trained and validated on large-size datasets, relying on using

GPUs. Most prior works focus more on retrieval accuracy but less on efficiency. We report the retrieval accuracy and retrieval efficiency on the 4 datasets in Table 2.1. The recorded time (in ms) indicates the average time for searching each query image. In Table 2.1, we observe some important trends. First, in general, the average retrieval time for each query image is less than $1s$. Concretely, the recorded time is up to $809ms$ on Oxford-105k in [153], whose mAP is 88.4%. The retrieval time is $600ms$ on Oxford-5k and $700ms$ on Paris-6k in [150], 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 $1ms$ in prior work [52, 57, 172], whose mAPs are lower than 70% (*i.e.* 55.5%, 64.8%, and 64.9%, respectively). In contrast, the prior approaches [150, 153, 165], reach relatively higher mAPs (*i.e.* 91.3%, 92.6%, and 95.8%, respectively), while this higher accuracy is at the expense of efficiency (more than $2ms$ even up to $600ms$). 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 discuss other important factors, including the depth of networks, retrieval feature dimension, and feature aggregation methods.

Network depth. We compare the efficacy of DCNNs depth, following the fine-tuning protocols[1] in [57]. For fair comparisons, all convolutional features from these backbone DCNNs are aggregated by MAC method [63], 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 2.13(a). We observe that the deeper networks is more beneficial for accuracy boosts, due to extracting more discriminative features.

Feature dimension. We vary the feature dimension of ResNet-50 from 32-d to 8192-d, by adding fully-connected layers on the top of pooled convolutional features. The results are shown in Figure 2.13(b). It is expected that higher-dimensional features capture much more semantics and are beneficial for retrieval. However, the performance tends to be stable when the dimension is very large. For ResNet-50, we observe that the 2048-d feature can already produce competitive results.

Feature aggregation methods. Here, we further discuss the methods of embedding convolutional feature maps, as illustrated in Figure 2.6. We use the off-the-shelf VGG (without updating parameters) on the Oxford and Paris datasets. The results are reported in Figure 2.12(b). We observe that different ways to aggregate the same off-the-shelf DCNN make differences for retrieval performance. These reported results provide a reference for feature aggregation when one uses convolutional layers for performing retrieval tasks.

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

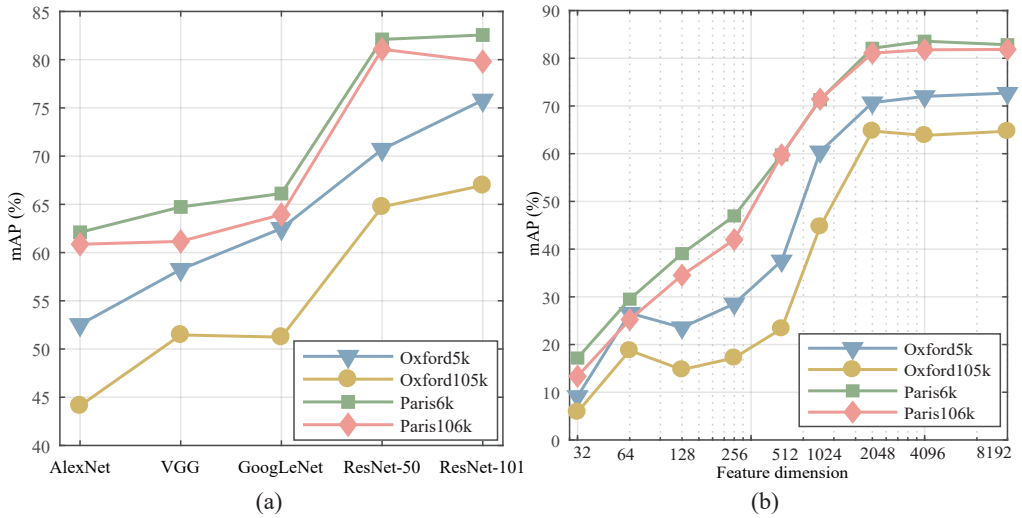


Figure 2.13: (a) The effectiveness of different DCNNs on 4 datasets. All models are fine-tuned by the same loss function. The results are tested on the convolutional features with default dimension; (b) The impact of feature dimension on retrieval performance. These features are extracted by using ResNet-50.

2.6 Chapter Conclusions

In this chapter, we reviewed 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 chapter not only provides a better understanding of image retrieval but also facilitates future research activities and application developments in this field.

Table 2.2: Performance evaluation of off-the-shelf DCNN models. "•" indicates that the models or layers are combined to learn features; "PCA_w indicates PCA with whitening on the extracted features to improve robustness; "MP" means Max Pooling; "SP" means Sum Pooling. The CNN-M network with "$*$" has an architecture similar to that of AlexNet. "-" means that the results were not reported.

Type	Method	Backbone DCNN	Output Layer	Feature Enhance.	Feature Dimension	Holidays	UKB	Oxford5k (+100k)	Paris6k (+100k)	Brief Conclusions and Highlights
Single Pass	Neural codes [50]	AlexNet	FC6	PCA	128	74.7	3.42 (N-S)	43.3 (38.6)	-	Compressed neural codes of different layers are explored. AlexNet is also fine-tuned for retrieval.
	R-MAC [31]	VGG16	Conv5	R-MAC + PCA_w	512	-	-	66.9 (61.6)	83.0 (75.7)	Adopting sliding windows with different scales on the convolutional feature maps to preserve spatial information.
	CroW [26]	VGG16	Conv5	CroW + PCA_w	256	85.1	-	68.4 (63.7)	76.5 (69.1)	The spatial- and channel-wise weighting mechanisms are utilized to highlight crucial convolutional features.
	BLCF [82]	VGG16	Conv5	BoW + PCA_w	25k	-	-	73.9 (59.3)	82.0 (64.8)	Both global features and local features are explored, demonstrating that local features have higher accuracy.
	SPoC [25]	VGG16	Conv5	SPoC + PCA_w	256	80.2	3.65 (N-S)	58.9 (57.8)	-	Exploring Gassian weighting scheme *i.e.* the centering prior, to improve the discrimination of features.
	Multi-layer CNN [92]	VGG16	FC6 • Conv4∼5	SP	4096	91.4	3.68 (N-S)	61.5 (-)	-	Layer-level feature fusion and the complementary properties of different layers are explored.
Multiple Pass	Deepindex [48]	AlexNet • VGG19	FC6-7 • FC17-18	BoW + PCA	512	81.7	3.32 (N-S)	-	75.4 (-)	Exploring layer-level and model-level fusion methods. Image patches are extracted using spatial pyramid modeling.
	MOF [65]	CNN-M* [75]	FC7 • Conv	SP or MP + BoW	20k	76.8	3.00 (N-S)	-	-	Exploring layer-level fusion scheme. Image patches are extracted using spatial pyramid modeling.
	Multi-scale CNN [63]	VGG16	Conv5	SP or MP + PCA_w	32k	89.6	95.1 (mAP)	84.3 (-)	87.9 (-)	Image patches are extracted in a dense manner. Geometric invariance is considered when aggregating patch features.
	CNNaug-ss [43]	Overfeat [166]	FC	PCA_w	15k	84.3	91.1 (mAP)	68.0 (-)	79.5 (-)	Image patches are extracted densely. Image regions at different locations with different sizes are included.
	MOP-CNN [29]	AlexNet	FC7	VLAD + PCA_w	2048	80.2	-	-	-	Image patches are extracted densely. Multi-scale patch features are further embedded into VLAD descriptors.
	CCS [73]	GoogLeNet	Conv	VLAD + PCA_w	128	84.1	3.81 (N-S)	64.8 (-)	76.8 (-)	Object proposals are extracted by RPNs. Object-level and point-level feature concatenation schemes are explored.
	OLDFP [85]	AlexNet	FC6	MP + PCA_w	512	88.5	3.81 (N-S)	60.7 (-)	66.2 (-)	Exploring the impact of proposal number. Patches are extracted by RPNs and the features are encoded in an orderless way.
	LDD [108]	VGG19	Conv5	BoW + PCA_w	500k	84.6	-	83.3 (-)	87.2 (-)	Image patches are obtained using a uniform square mesh. Patch features are encoded into BoW descriptors.

Table 2.3: Performance evaluation of methods in which DCNN models are fine-tuned, in a supervised or an unsupervised manner. "CE Loss" means the models are fine-tuned using the classification-based loss function in the form of Eq. 2.13. "Siamese Loss" is in the form of Eq. 2.16. "Regression Loss" is in the form of Eq. 2.15. "Triplet Loss" is in the form of Eq. 2.17.

Type	Method	Backbone DCNN	Output Layer	Feature Enhance.	Loss Function	Feature Dimension	Holidays	UKB	Oxford5k (+100k)	Paris6k (+100k)	Brief Conclusions and Highlights
Supervised Fine-tuning	DELF [106]	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.
	Neural codes [50]	AlexNet	FC6	PCA	CE Loss	128	78.9	3.29 (N-S)	55.7 (52.3)	-	The first work which fine-tunes deep networks for image retrieval. Compressed neural codes and different layers are explored.
	Non-metric [52]	VGG16	Conv5	PCA_w	Regression Loss	512	-	-	88.2 (82.1)	88.2 (82.9)	Visual similarity learning of similar and dissimilar pairs is performed by a neural network, optimized using regression loss.
	Faster R-CNN [89]	VGG16	Conv5	MP / SP	Regression Loss	512	-	-	75.1 (-)	80.7 (-)	RPN is fine-tuned, based on bounding box coordinates and class scores for specific region query which is region-targeted.
	SIAM-FV [53]	VGG16	Conv5	FV + PCA_w	Siamese Loss	512	-	-	81.5 (76.6)	82.4 (-)	Fisher Vector is integrated on top of VGG and is trained with VGG simultaneously.
	SIFT-CNN [171]	VGG16	Conv5	SP	Siamese Loss	512	88.4	3.91 (N-S)	-	-	SIFT features are used as supervisory information for mining positive and negative samples.
	Quartet-Net [142]	VGG16	FC6	PCA	Siamese Loss	128	71.2	87.5 (mAP)	48.5 (-)	48.8 (-)	Quartet-net learning is explored to improve feature discrimination where double-margin contrastive loss is used.
	NetVLAD [55]	VGG16	VLAD Layer	PCA_w	Triplet Loss	256	79.9	-	62.5 (-)	72.0 (-)	VLAD is integrated at the last convolutional layer of VGG16 network as a plugged layer.
	Deep Retri-eval [140]	ResNet-101	Conv5 Block	MP + PCA_w	Triplet Loss	2048	90.3	-	86.1 (82.8)	94.5 (90.6)	Dataset is cleaned automatically. Features are encoded by R-MAC. RPN is used to extract the most relevant regions.
Unsupervised Fine-tuning	MoM [151]	VGG16	Conv5	MP + PCA_w	Siamese Loss	64	87.5	-	78.2 (72.6)	85.1 (78.0)	Exploring manifold learning for mining dis/similar samples. Features are tested globally and regionally.
	GeM [57]	VGG16	Conv5	GeM Pooling	Siamese Loss	512	83.1	-	82.0 (76.9)	79.7 (72.6)	Fine-tuning CNNs on an unordered dataset. Samples are selected from an automated 3D reconstruction system.
	SfM-CNN [32]	VGG16	Conv5	PCA_w	Siamese Loss	512	82.5	-	77.0 (69.2)	83.8 (76.4)	Employing Structure-from-Motion to select positive and negative samples from unordered images.
	IME-CNN [56]	ResNet-101	IME Layer	MP	Regression Loss	2048	-	-	92.0 (87.2)	96.6 (93.3)	Graph-based manifold learning is explored within an IME layer to mine the matching and non-matching pairs in unordered datasets.
	MDP-CNN [152]	ResNet-101	Conv5 Block	SP	Triplet Loss	2048	-	-	85.4 (85.1)	96.3 (94.7)	Exploring global feature structure by modeling the manifold learning to select positive and negative pairs.