ExchNet: A Unified Hashing Network for Large-Scale Fine-Grained Image Retrieval

Quan Cui[†1,3], Qing-Yuan Jiang[†2], Xiu-Shen Wei[*3], Wu-Jun Li[2], and Osamu Yoshie[1]

[1] Graduate School of IPS, Waseda University, Japan
[2] National Key Laboratory for Novel Software Technology, Department of Computer Science and Technology, Nanjing University, China
[3] Megvii Research Nanjing, Megvii Technology, China
cui-quan@toki.waseda.jp, qyjiang24@gmail.com, weixs.gm@gmail.com, liwujun@nju.edu.cn, yoshie@waseda.jp

Abstract Retrieving content relevant images from a large-scale fine-grained dataset could suffer from intolerably slow query speed and highly redundant storage cost, due to high-dimensional real-valued embeddings which aim to distinguish subtle visual differences of fine-grained objects. In this paper, we study the novel fine-grained hashing topic to generate compact binary codes for fine-grained images, leveraging the search and storage efficiency of hash learning to alleviate the aforementioned problems. Specifically, we propose a unified end-to-end trainable network, termed as ExchNet. Based on attention mechanisms and proposed attention constraints, ExchNet can firstly obtain both local and global features to represent object parts and the whole fine-grained objects, respectively. Furthermore, to ensure the discriminative ability and semantic meaning's consistency of these part-level features across images, we design a local feature alignment approach by performing a feature exchanging operation. Later, an alternating learning algorithm is employed to optimize the whole ExchNet and then generate the final binary hash codes. Validated by extensive experiments, our ExchNet consistently outperforms state-of-the-art generic hashing methods on five fine-grained datasets. Moreover, compared with other approximate nearest neighbor methods, ExchNet achieves the best speed-up and storage reduction, revealing its efficiency and practicality.

Keywords: Fine-Grained Image Retrieval; Learning to Hash; Feature Alignment; Large-Scale Image Search.

1 Introduction

Fine-Grained Image Retrieval (FGIR) [36,42,43,31,26,19] is a practical but challenging computer vision task. It aims to retrieve images belonging to various

[†] Equal contribution.
[*] Corresponding author.

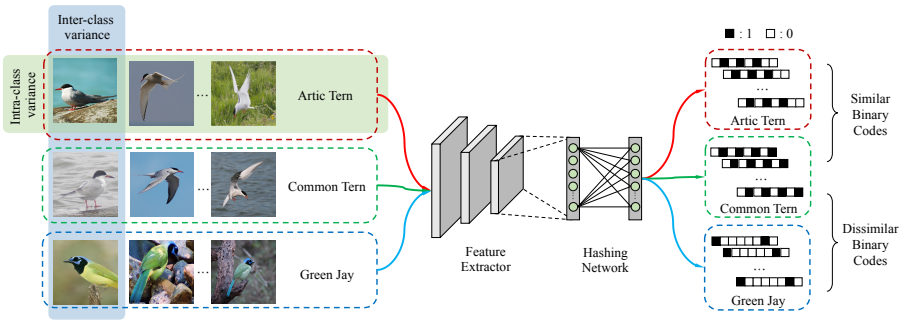


Figure 1. Illustration of the fine-grained hashing task. Fine-grained images could share large intra-class variances but small inter-class variances. Fine-grained hashing aims to generate compact binary codes with tiny Hamming distances for images of the same sub-category, as well as distinct codes for images from different sub-categories.

sub-categories of a certain meta-category (e.g., birds, cars and aircrafts) and return images with the same sub-category as the query image. In real FGIR applications, previous methods could suffer from slow query speed and redundant storage costs due to both the explosive growth of massive fine-grained data and high-dimensional real-valued features.

Learning to hash [6,10,34,35,21,17,22,16,3,14,7] has proven to be a promising solution for large-scale image retrieval because it can greatly reduce the storage cost and increase the query speed. As a representative research area of approximate nearest neighbor (ANN) search [6,13,1], hashing aims to embed data points as similarity-preserving binary codes. Recently, hashing has been successfully applied in a wide range of image retrieval tasks, e.g., face image retrieval [18], person re-identification [44,5], etc. We hereby explore the effectiveness of hashing for *fine-grained* image retrieval.

To the best of our knowledge, this is the first work to study the fine-grained hashing problem, which refers to the problem of designing hashing for fine-grained objects. As shown in Figure 1, the task is desirable to generate compact binary codes for fine-grained images sharing both large intra-class variances and small inter-class variances. To deal with the challenging task, we propose a unified end-to-end trainable network ExchNet to first learn fine-grained tailored features and then generate the final binary hash codes.

In concretely, our ExchNet consists of three main modules, including representation learning, local feature alignment and hash code learning, as shown in Figure 2. In the representation learning module, beyond obtaining the holistic image representation (i.e., global features), we also employ the attention mechanism to capture the part-level features (i.e., local features) for representing fine-grained objects' parts. Localizing parts and embedding part-level cues are crucial for fine-grained tasks, since these discriminative but subtle parts (e.g., bird heads or tails) play a major role to distinguish different sub-categories. Moreover, we also

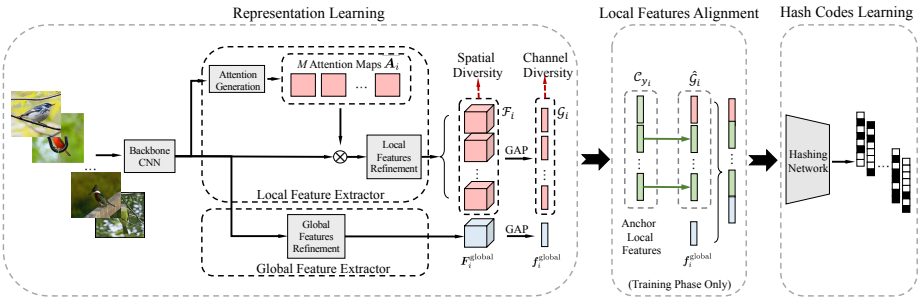


Figure 2. Framework of our proposed ExchNet, which consists of three modules. 1) The representation learning module, as well as the attention mechanism with spatial and channel diversity learning constraints, is designed to obtain both local and global features of fine-grained objects. 2) The local feature alignment module is used to align obtained local features w.r.t. object parts across different fine-grained images. 3) The hash codes learning module is performed to generate the compact binary codes.

develop two kinds of attention constraints, i.e., spatial and channel constraints, to collaboratively work together for further improving the discriminative ability of these local features. In the following, to ensure that these part-level features can correspond to their own corresponding parts across different fine-grained images, we design an anchor based feature alignment approach to align these local features. Specifically, in the local feature alignment module, we treat the anchored local features as the "prototype" w.r.t. its sub-category by averaging all the local features of that part across images. Once local features are well aligned for their own parts, even if we exchange one specific part's local feature of an input image with the same part's local feature of the prototype, the image meanings derived from the image representations and also the final hash codes should be both extremely similar. Inspired by this motivation, we perform a feature exchanging operation upon the anchored local features and other learned local features, which is illustrated in Figure 3. After that, for effectively training the network with our feature alignment fashion, we utilize an alternating algorithm to solve the hashing learning problem and update anchor features simultaneously.

To quantitatively prove both effectiveness and efficiency of our ExchNet, we conduct comprehensive experiments on five fine-grained benchmark datasets, including the large-scale ones, i.e., *NABirds* [11], *VegFru* [12] and *Food101* [23]. Particularly, compared with competing approximate nearest neighbor methods, our ExchNet achieves up to hundreds times speedup for large-scale fine-grained image retrieval without significant accuracy drops. Meanwhile, compared with state-of-the-art generic hashing methods, ExchNet could consistently outperform these methods by a large margin on all the fine-grained datasets. Additionally, ablation studies and visualization results justify the effectiveness of our tailored model designs like local feature alignment and proposed attention approach.

The contributions of this paper are summarized as follows:

- We study the novel fine-grained hashing topic to leverage the search and storage efficiency of hash codes for solving the challenging large-scale fine-grained image retrieval problem.
- We propose a unified end-to-end trainable network, i.e., ExchNet, to first learn fine-grained tailored features and then generate the final binary hash codes. Particularly, the proposed attention constraints, local feature alignment and anchor-based learning fashion contribute well to obtain discriminative fine-grained representations.
- We conduct extensive experiments on five fine-grained datasets to validate both effectiveness and efficiency of our proposed ExchNet. Especially for the results on large-scale datasets, ExchNet exhibits its outperforming retrieval performance on either speedup, memory usages and retrieval accuracy.

2 Related Work

Fine-Grained Image Retrieval Fine-Grained Image Retrieval (FGIR) is an active research topic emerged in recent years, where the database and query images could share small inter-class variance but large intra-class variance. In previous works [36], handcrafted features were initially utilized to tackle the FGIR problem. Powered by deep learning techniques, more and more deep learning based FGIR methods [36,42,33,43,31,26,19,32] were proposed. These deep methods can be roughly divided into two parts, i.e., supervised and unsupervised methods. In supervised methods, FGIR is defined as a metric learning problem. Zheng et al. [42] designed a novel ranking loss and a weakly-supervised attractive feature extraction strategy to facilitate the retrieval performance. Zheng et al. [43] improved their former work [42] with a normalize-scale layer and de-correlated ranking loss. As to unsupervised methods, Selective Convolutional Descriptor Aggregation (SCDA) [31] was proposed to localize the main object in fine-grained images firstly, and then discard the noisy background and keep useful deep descriptors for fine-grained image retrieval.

Deep Hashing Hashing methods can be divided into two categories, i.e., data-independent methods [6] and data-dependent methods [10,17], based on whether training points are used to learn hash functions. Generally speaking, data-dependent methods, also named as Learning to Hash (L2H) methods, can achieve better retrieval performance with the help of the learning on training data. With the rise of deep learning, some L2H methods integrate deep feature learning into hash frameworks and achieve promising performance. As previous work, many deep hashing methods [35,21,17,22,16,3,14,7,38,2,30,40,39] for large-scale image retrieval have been proposed. Compared with deep unsupervised hashing methods [21,7,14], deep supervised hashing methods [35,17,16,14] can achieve superior retrieval accuracy as they can fully explore the semantic information. Specifically, the previous work [35] was essentially a two-stage method which tried to learn binary codes in the first stage and employed feature learning guided by the learned binary codes in the second stage. Then, there appeared numerous

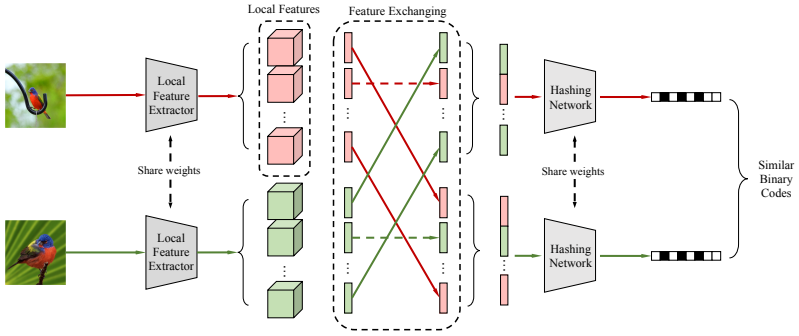


Figure 3. Key idea of our local feature alignment approach: Given an image pair of a fine-grained category, exchanging their local features of the same object parts should not change their corresponding hash codes, i.e., these hash codes should be the same as those generated without local feature exchanging and their Hamming distance should be still close also.

one-stage deep supervised hashing methods, including Deep Pairwise Supervised Hashing (DPSH) [17], Deep Supervised Hashing (DSH) [22], and Deep Cauchy Hashing (DCH) [3], which aimed to integrate feature learning and hash code learning into an end-to-end framework.

3 Methodology

The framework of our ExchNet is presented in Figure 2, which contains three key modules, i.e., the representation learning module, local feature alignment module, and hash code learning module.

3.1 Representation Learning

The learning of discriminative and meaningful local features is mutually correlated with fine-grained tasks [20,15,37,41,9], since these local features can greatly benefit the distinguishing of sub-categories with subtle visual differences deriving from the discriminative fine-grained parts (e.g., bird heads or tails) . In consequence, as shown in Figure 2, beyond the global feature extractor, we also introduce a local feature extractor in the representation learning module. Specifically, by considering model efficiency, we hereby propose to learn local features with the attention mechanism, rather than other fine-grained techniques with tremendous computation cost, e.g., second-order representations [20,15] or complicated network architectures [37,41,9].

Given an input image $\boldsymbol{x}_i$, a backbone CNN is utilized to extract a holistic deep feature $\boldsymbol{E}_i \in \mathbb{R}^{H \times W \times C}$, which serves as the appetizer for both the local feature extractor and the global feature extractor.

It is worth mentioning that the attention is engaged in the middle of the feature extractor. Since, in the shallow layers of deep neural networks, low-level

context information (e.g., colors and edges, etc.) are well preserved, which is crucial for distinguish subtle visual differences of fine-grained objects. Then, by feeding $\boldsymbol{E}_i$ into the attention generation module, M pieces of attention maps $\boldsymbol{A}_i \in \mathbb{R}^{M\times H\times W}$ are generated and we use $\boldsymbol{A}_i^j \in \mathbb{R}^{H\times W}$ to denote the attentive region of the j-th ($j \in \{1,\ldots,M\}$) part cues for $\boldsymbol{x}_i$. After that, the obtained part-level attention map $\boldsymbol{A}_i^j$ is element-wisely multiplied on $\boldsymbol{E}_i$ to select the attentive local feature corresponding to the j-th part, which is formulated as:

$$\hat{\boldsymbol{E}}_i^j = \boldsymbol{E}_i \otimes \boldsymbol{A}_i^j, \tag{1}$$

where $\hat{\boldsymbol{E}}_i^j \in \mathbb{R}^{H\times W\times C}$ represents the j-th attentive local feature of $\boldsymbol{x}_i$, and "$\otimes$" denotes the Hadamard product on each channel. For simplification, we use $\hat{\mathcal{E}}_i = \{\hat{\boldsymbol{E}}_i^1,\ldots,\hat{\boldsymbol{E}}_i^M\}$ to denote a set of local features and, subsequently, $\hat{\mathcal{E}}_i$ is fed into the later Local Features Refinement (LFR) network composed of a stack of convolution layers to embed these attentive local features into higher-level semantic meanings:

$$\mathcal{F}_i = f_{\texttt{LFR}}(\hat{\mathcal{E}}_i), \tag{2}$$

where the output of the network is denoted as $\mathcal{F}_i = \{\boldsymbol{F}_i^1,\ldots,\boldsymbol{F}_i^M\}$, which represents the final local feature maps w.r.t. high-level semantics. We denote $\boldsymbol{f}_i^j \in \mathbb{R}^{C'}$ as the local feature vector after applying global average pooling (GAP) on $\boldsymbol{F}_i^j \in \mathbb{R}^{H'\times W'\times C'}$ as:

$$\boldsymbol{f}_i^j = f_{\texttt{GAP}}(\boldsymbol{F}_i^j)\,. \tag{3}$$

On the other side, as to the global feature extractor, for $\boldsymbol{x}_i$, we directly adopt a Global Features Refinement (GFR) network composed of conventional convolutional operations to embed $\boldsymbol{E}_i$, which is presented by:

$$\boldsymbol{F}_i^{\text{global}} = f_{\texttt{GFR}}(\boldsymbol{E}_i)\,. \tag{4}$$

We use $\boldsymbol{F}_i^{\text{global}} \in \mathbb{R}^{H'\times W'\times C'}$ and $\boldsymbol{f}_i^{\text{global}} \in \mathbb{R}^{C'}$ to denote the learned global feature and the corresponding holistic feature vector after GAP, respectively.

Furthermore, to facilitate the learning of localizing local feature cues (i.e., capturing fine-grained parts), we impose the spatial diversity and channel diversity constraints over the local features in $\mathcal{F}_i$.

Specifically, it is a natural choice to increase the diversity of local features by differentiating the distributions of attention maps [41]. However, it might cause a problem that the holistic feature can not be activated in some spatial positions, while the attention map has large activation values on them due to over-applied constraints upon the learned attention maps. Instead, in our method, we design and apply constraints on the local features. In concretely, for the local feature $\boldsymbol{F}_i^j$, we obtain its "aggregation map" $\hat{\boldsymbol{A}}_i^j \in \mathbb{R}^{H'\times W'}$ by adding all C' feature maps through the channel dimension and apply the softmax function on it for converting it into a valid distribution, then flat it into a vector $\hat{\boldsymbol{a}}_i^j$. Based on the

Hellinger distance, we propose a spatial diversity induced loss as:

$$\mathcal{L}_{\mathtt{sp}}(\boldsymbol{x}_i) = 1 - \frac{1}{\sqrt{2}\binom{M}{2}} \sum_{l,k=1}^{M} \left\| \sqrt{\hat{\boldsymbol{a}}_i^l} - \sqrt{\hat{\boldsymbol{a}}_i^k} \right\|_2, \tag{5}$$

where $\binom{M}{2}$ is used to denote the combinatorial number of ways to pick 2 unordered outcomes from M possibilities. The spatial diversity constraint drives the aggregation maps to be activated in spatial positions as diverse as possible. As to the channel diversity constraint, we first convert the local feature vector $\boldsymbol{f}_i^j$ into a valid distribution, which can be formulated by

$$\boldsymbol{p}_i^j = \mathtt{softmax}(\boldsymbol{f}_i^j),\ \forall j \in \{1, \ldots, M\}. \tag{6}$$

Subsequently, we propose a constraint loss over $\{\boldsymbol{p}_i^j\}_{j=1}^M$ as:

$$\mathcal{L}_{\mathtt{cp}}(\boldsymbol{x}_i) = \left[t - \frac{1}{\sqrt{2}\binom{M}{2}} \sum_{l,k=1}^{M} \left\| \sqrt{\boldsymbol{p}_i^l} - \sqrt{\boldsymbol{p}_i^k} \right\|_2 \right]_+, \tag{7}$$

where $t \in [0, 1]$ is a hyper-parameter to adjust the diversity and $[\cdot]_+$ denotes $max(\cdot, 0)$. Equipping with the channel diversity constraint could benefit the network to depress redundancies in features through channel dimensions. Overall, our spatial diversity and channel diversity constraints can work in a collaborative way to obtain discriminative local features.

3.2 Learning to Align by Local Feature Exchanging

Upon the representation learning module, the alignment on local features is necessary for confirming that they represent and more importantly correspond to common fine-grained parts across images, which are essential to fine-grained tasks. Hence, we propose an anchor-based local features alignment approach assisted with our feature exchanging operation.

Intuitively, local features from the same object part (e.g., bird heads of a bird species) should be embedded with almost the same semantic meaning. As illustrated by Figure 3, our key idea is that, if local features were well aligned, exchanging the features of identical parts for two input images belonging to the same sub-category should not change the generated hash codes. Inspired by that, we propose a local feature alignment strategy by leveraging the feature exchanging operation, which happens between learned local features and anchored local features. As a foundation for feature exchanging, a set of dynamic anchored local features $\mathcal{C}_{y_i} = \{\boldsymbol{c}_{y_i}^1, \ldots, \boldsymbol{c}_{y_i}^M\}$ for class y_i should be maintained, in which the j-th anchored local feature $\boldsymbol{c}_{y_i}^j$ is obtained by averaging all j-th part's local features of training samples from class y_i. At the end of each training epoch, anchored local features will be recalculated and updated. Subsequently, as shown in Figure 4, for a sample $\boldsymbol{x}_i$ whose category is y_i, we exchange a half of the

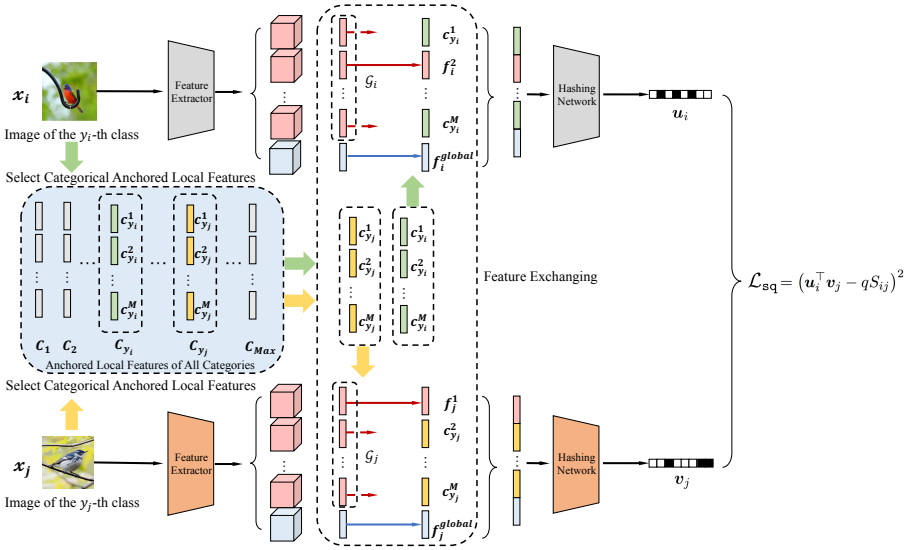


Figure 4. Our feature exchanging and hash codes learning in the training phase. According to the class indices (i.e., y_i and y_j), we first select categorical anchor features $\mathcal{C}_{y_i}$ and $\mathcal{C}_{y_j}$ for samples $\boldsymbol{x}_i$ and $\boldsymbol{x}_j$, respectively. Then, for each input image, the feature exchanging operation is conducted between its learned and anchored local features. After that, hash codes are generated with exchanged features and the learning is driven by preserving pairwise similarities of hash codes $\boldsymbol{u}_i$ and $\boldsymbol{v}_j$.

learned local features in $\mathcal{G}_i = \{\boldsymbol{f}_i^1, \ldots, \boldsymbol{f}_i^M\}$ with its corresponding anchored local features in $\mathcal{C}_{y_i} = \{\boldsymbol{c}_{y_i}^1, \ldots, \boldsymbol{c}_{y_i}^M\}$. The exchanging process can be formulated as:

$$\forall j \in \{1, \ldots, M\}, \hat{\boldsymbol{f}}_i^j \triangleq \begin{cases} \boldsymbol{f}_i^j, & \text{if } \xi_j \geq 0.5, \\ \boldsymbol{c}_{y_i}^j, & \text{otherwise}, \end{cases} \tag{8}$$

where $\xi_j \sim \mathcal{B}(0.5)$ is a random variable following the Bernoulli distribution for the j-th part. The local features after exchanging are denoted as $\hat{\mathcal{G}}_i = \{\hat{\boldsymbol{f}}_i^1, \ldots, \hat{\boldsymbol{f}}_i^M\}$ and fed into the hashing learning module for generating binary codes and computing similarity preservation losses.

3.3 Hash Code Learning

After obtaining both global features and local features, we concatenate them together and feed them into the hashing learning module. Specifically, the hashing network contains a fully connected layer and a $\texttt{sign}(\cdot)$ activation function layer. In our method, we choose an asymmetric hashing for ExchNet for its flexibility [25]. Concretely, we utilize two hash functions, defined as $g(\cdot)$ and $h(\cdot)$, to learn two different binary codes for the same training sample. The learning procedure is as

follows:

$$\boldsymbol{u}_i = g([\hat{\mathcal{G}}_i; \boldsymbol{f}_i^{\text{global}}]_{\texttt{cat}}) = \texttt{sign}(\boldsymbol{W}^{(g)}[\hat{\mathcal{G}}_i; \boldsymbol{f}_i^{\text{global}}]_{\texttt{cat}}), \tag{9}$$

$$\boldsymbol{v}_i = h([\hat{\mathcal{G}}_i; \boldsymbol{f}_i^{\text{global}}]_{\texttt{cat}}) = \texttt{sign}(\boldsymbol{W}^{(h)}[\hat{\mathcal{G}}_i; \boldsymbol{f}_i^{\text{global}}]_{\texttt{cat}}), \tag{10}$$

where $[;]_{\text{cat}}$ denotes the concatenation operator, and $\boldsymbol{u}_i, \boldsymbol{v}_i \in \{-1, +1\}^q$ denote the two different binary codes of the i-th sample. q represents the code length. $\boldsymbol{W}^{(g)}$ and $\boldsymbol{W}^{(h)}$ present the parameters of hash functions $g(\cdot)$ and $h(\cdot)$*, respectively. We denote $\boldsymbol{U} = \{\boldsymbol{u}_i\}_{i=1}^n$ and $\boldsymbol{V} = \{\boldsymbol{v}_i\}_{i=1}^n$ as learned binary codes. Inspired by [14], we only keep binary codes $\boldsymbol{v}_i$ and set hash function $h(\cdot)$ implicitly. Hence, we can perform feature learning and binary codes learning simultaneously.

To preserve the pairwise similarity, we adopt the squared loss and define the following objective function:

$$\mathcal{L}_{\texttt{sq}}(\boldsymbol{u}_i, \boldsymbol{v}_j, \boldsymbol{\mathcal{C}}) = \left(\boldsymbol{u}_i^\top \boldsymbol{v}_j - qS_{ij}\right)^2, \tag{11}$$

where $\boldsymbol{u}_i = g([\hat{\mathcal{G}}_i; \boldsymbol{f}_i^{\text{global}}]_{\texttt{cat}})$, S_{ij} is the pairwise similarity label and $\boldsymbol{\mathcal{C}} = \{\mathcal{C}_i\}_{i=1}^M$. We use Θ to denote the parameters of deep neural network and hash layer. The aforementioned process is generally illustrated by Figure 4.

Due to the zero-gradient problem caused by the $\texttt{sign}(\cdot)$ function, $L_{\texttt{sq}}(\cdot,\cdot,\cdot)$ becomes intractable to optimize. In this paper, we relax $g(\cdot) = \texttt{sign}(\cdot)$ into $\hat{g}(\cdot) = \texttt{tanh}(\cdot)$ to alleviate this problem. Then, we can derive the following loss function:

$$\hat{\mathcal{L}}_{\texttt{sq}}(\hat{\boldsymbol{u}}_i, \boldsymbol{v}_j, \boldsymbol{\mathcal{C}}) = \left(\hat{\boldsymbol{u}}_i^\top \boldsymbol{v}_j - qS_{ij}\right)^2, \tag{12}$$

where $\hat{\boldsymbol{u}}_i = \hat{g}([\hat{\mathcal{G}}_i; \boldsymbol{f}_i^{\text{global}}]_{\texttt{cat}})$ and $\boldsymbol{U}$ is relaxed as $\hat{\boldsymbol{U}} = \{\hat{\boldsymbol{u}}_i\}_{i=1}^n$.

Then, given a set of image samples $\mathcal{X} = \{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n\}$ and their pairwise labels $\boldsymbol{S} = \{S_{ij}\}_{i,j=1}^n$, we can get the following objective function by combining Equation (5), (7) and (12):

$$\begin{aligned} &\min_{\boldsymbol{V},\Theta,\boldsymbol{\mathcal{C}}} \mathcal{L}(\mathcal{X}) = \sum_{i,j=1}^{n} \hat{\mathcal{L}}_{\texttt{sq}}(\hat{\boldsymbol{u}}_i, \boldsymbol{v}_j; S_{ij}) + \lambda \sum_{i=1}^{n} \mathcal{L}_{\texttt{sp}}(\boldsymbol{x}_i) + \gamma \sum_{i=1}^{n} \mathcal{L}_{\texttt{cp}}(\boldsymbol{x}_i) \\ &\texttt{s.t.}\ \forall i \in \{1, \ldots, n\}, \hat{\boldsymbol{u}}_i = \hat{g}([\hat{\mathcal{G}}_i; \boldsymbol{f}_i^{\text{global}}]_{\texttt{cat}}), \boldsymbol{v}_j \in \{-1, +1\}^q, \end{aligned} \tag{13}$$

where S_{ij} represents the similarity between the i-th and j-th samples, q denotes the code length, λ and γ are hyper-parameters.

3.4 Learning Algorithm

To solve the optimization problem in Equation (13), we design an alternating algorithm to learn $\boldsymbol{V}$, Θ, and $\boldsymbol{\mathcal{C}}$. Specifically, we learn one parameter with the others fixed.

* We omit the bias term for simplicity.

Learn $\boldsymbol{\Theta}$ with $\boldsymbol{V}$ and $\boldsymbol{\mathcal{C}}$ fixed When $\boldsymbol{V}, \mathcal{C}$ fixed, we use back-propagation (BP) to update the parameters Θ. In particular, for input sample $\boldsymbol{x}_i$, we first calculate the following gradient:

$$\nabla_\Theta \mathcal{L}(\boldsymbol{X}) = \sum_{i,j=1}^{n} \nabla_\Theta \mathcal{L}_{\mathtt{sq}}(\hat{\boldsymbol{u}}_i, \boldsymbol{v}_j) + \lambda \sum_{i=1}^{n} \nabla_\Theta \mathcal{L}_{\mathtt{sp}}(\boldsymbol{x}_i) + \gamma \sum_{i=1}^{n} \nabla_\Theta \mathcal{L}_{\mathtt{cp}}(\boldsymbol{x}_i). \quad (14)$$

Then, we use the back-propagation algorithm to update Θ.

Learn $\boldsymbol{V}$ with $\boldsymbol{\Theta}$ and $\boldsymbol{\mathcal{C}}$ fixed When Θ, $\mathcal{C}$ are fixed, we rewrite $\mathcal{L}(\boldsymbol{V})$ as follows:

$$\mathcal{L}(\boldsymbol{V}) = \sum_{i,j=1}^{n} \left(\hat{\boldsymbol{u}}_i^\top \boldsymbol{v}_j - qS_{ij}\right)^2 = \|\widetilde{\boldsymbol{U}}\boldsymbol{V}^\top - q\boldsymbol{S}\|_F^2 \quad (15)$$

$$= \|\widetilde{\boldsymbol{U}}\boldsymbol{V}^\top\|_F^2 - 2q\mathrm{tr}(\boldsymbol{S}^\top \widetilde{\boldsymbol{U}}\boldsymbol{V}^\top) + \mathtt{const}. \quad (16)$$

Because $\boldsymbol{V}$ is defined over $\{-1, +1\}^{n\times q}$, we learn $\boldsymbol{V}$ column by column as that in ADSH [14]. Specifically, we can get the closed-form solution for the k-th column $\boldsymbol{V}_{*k}$ as follows:

$$\boldsymbol{V}_{*k} = \mathtt{sign}(\boldsymbol{V}_{/k}\widetilde{\boldsymbol{U}}_{/k}^\top \widetilde{\boldsymbol{U}}_{*k} - q\boldsymbol{Q}_{*k}), \quad (17)$$

where $\boldsymbol{Q} = \boldsymbol{S}^\top \widetilde{\boldsymbol{U}}$ and $\boldsymbol{V}_{/k}$ denotes the matrix excluding the k-th column .

Learn $\boldsymbol{\mathcal{C}}$ with $\boldsymbol{V}$ and $\boldsymbol{\Theta}$ fixed When Θ, $\boldsymbol{V}$ fixed, we use the following equation to update each $\mathcal{C}_i \in \boldsymbol{\mathcal{C}}$:

$$\forall k, \boldsymbol{c}_i^k = \frac{1}{n_i} \sum_{i=1}^{n_i} \boldsymbol{f}_i^k, \quad (18)$$

where n_i denotes the number of samples in class y_i.

3.5 Out-of-Sample Extension

When we finish the training phase, we can generate the binary code for the sample $\boldsymbol{x}_i$ by $\boldsymbol{u}_i = \mathtt{sign}(\boldsymbol{W}^{(g)}[\mathcal{G}_i; \boldsymbol{f}_i^{\mathrm{global}}]_{\mathtt{cat}})$.

4 Experiments

4.1 Datasets

For comparisons, we select two widely used fine-grained datasets, i.e., *CUB* [29] and *Aircraft* [24], as well as three popular large-scale fine-grained datasets, i.e., *NABirds* [11], *VegFru* [12], and *Food101* [23], to conduct experiments.

Specifically, *CUB* is a bird classification benchmark dataset containing 11,788 images from 200 bird species. It is officially split into 5,994 for training and 5,794 for test. *Aircraft* contains 10,000 images from 100 kinds of aircraft model variants with 6667 for training and 3333 for test. Moreover, for large-scale datasets, *NABirds* has 555 common species of birds in North America with 23,929 training images and 24,633 test images. *VegFru* is a large-scale fine-grained dataset covering vegetables and fruits from 292 categories with 29,200 for training and 116,931 for test.*Food101* contains 101 kinds of foods with 101,000 images. For each class, 250 test images are manually reviewed for correctness while 750 training images still contain some amount of noises.

4.2 Baselines and Implementation Details

Baselines For comparisons with other ANN algorithms, we select two tree-based ANN methods, i.e., BallTree [8] and KDTree [1], and one production quantization based ANN method, i.e., Product Quantization (PQ) [13]. The linear scan means that we directly perform exhaustive search based on the learned real-valued features. For comparisons with other hashing baselines, we choose eight state-of-the-art generic hashing methods. They are LSH [6], SH [34], ITQ [10], SDH [28], DPSH [17], DSH [22], HashNet [4], and ADSH [14]. Among these methods, DPSH, DSH, HashNet and ADSH are based on deep learning and others are not.

Implementation Details For comparisons with other ANN algorithms, we carry out experiments on *Food101* in which the database is the largest. We first utilize the triplet loss [27] to learn 512-D and 1024-D feature embeddings for its frequent usages in fine-grained retrieval tasks. Then, the performance of linear scan is tested on the learned features. More experimental settings about BallTree [8], KDTree [1] and PQ [13] can be found in the supplementary materials. For our ExchNet, the retrieval procedure is divided into coarse ranking to select top N as candidates and re-ranking to return top K ($K < N$) from top N candidates. We adopt the real-valued features learned with the triplet loss directly. As presented in Table 1, we report results including precision at top K (P@K), wall clock time (WC time), speed up ratio, and memory cost.

Our backbone employs the first three stages of ResNet50 and the attention generation module is the fourth stage of ResNet50 without downsample convolutions. The LFR and GFR of ExchNet are independent networks, sharing the same architecture with the fourth stage of ResNet50. The optimizer is standard mini-batch stochastic gradient descent with weight decay 1×10^{-4}. The mini-batch size M is set to 64 and the iteration times T_{max} is 100. Learning rate is set to 0.001, which is divided by 10 at the 60-th and 80-th iteration, respectively. The hyper-parameter t is set to 0.4. The number of training epochs is 20. For efficient training, we randomly sample a subset of the training set in each epoch. Specifically, for *CUB*, *Aircraft*, *Food101*, we sample 2,000 samples per epoch, while 4,000 samples are randomly selected for other datasets. To provide reliable local features for our local feature alignment strategy, in the first 50 iterations,

Table 1. Retrieval performance comparisons on the *Food101* dataset.

Method	512-dim				1024-dim			
	P@10(↑)	WCtime(↓)	Speedup(↑)	Memory(↓)	P@10(↑)	WCtime(↓)	Speedup(↑)	Memory(↓)
Linear	80.05%	9,481.03	1×	207.2MB	80.28%	22,377.96	1×	414.1MB
BallTree	77.22%	236.23	40.13×	28.1MB	77.74%	213.88	104.62×	28.1MB
KDTree	77.42%	70.16	135.13×	28.8MB	77.73%	73.57	304.14×	28.7MB
PQ	77.12%	43.49	217.99×	524.5KB	77.18%	72.47	308.74×	1.0MB
Ours	**77.69%**	**40.54**	**233.85×**	**404.0KB**	**78.06%**	**56.57**	**395.53×**	**404.0KB**

Table 2. Comparisons of retrieval accuracy (MAP) on all the fine-grained datasets.

Method	#Bits	LSH	SH	ITQ	SDH	DPSH	DSH	HashNet	ADSH	**Ours**
CUB	12bits	2.26%	5.55%	6.80%	10.52%	8.68%	4.48%	12.03%	20.03%	**25.14%**
	24bits	3.59%	6.72%	9.42%	16.95%	12.51%	7.97%	17.77%	50.33%	**58.98%**
	32bits	5.01%	7.63%	11.19%	20.43%	12.74%	7.72%	19.93%	61.68%	**67.74%**
	48bits	6.16%	8.32%	12.45%	22.23%	15.58%	11.81%	22.13%	65.43%	**71.05%**
Aircraft	12bits	1.69%	3.28%	4.38%	4.89%	8.74%	8.14%	14.91%	15.54%	**33.27%**
	24bits	2.19%	3.85%	5.28%	6.36%	10.87%	10.66%	17.75%	23.09%	**45.83%**
	32bits	2.38%	4.04%	5.82%	6.90%	13.54%	12.21%	19.42%	30.37%	**51.83%**
	48bits	2.82%	4.28%	6.05%	7.65%	13.94%	14.45%	20.32%	50.65%	**59.05%**
NABirds	12bits	0.90%	2.12%	2.53%	3.10%	2.17%	1.56%	2.34%	2.53%	**5.22%**
	24bits	1.68%	3.14%	4.22%	6.72%	4.08%	2.33%	3.29%	8.23%	**15.69%**
	32bits	2.43%	3.71%	5.38%	8.86%	3.61%	2.44%	4.52%	14.71%	**21.94%**
	48bits	3.09%	4.05%	6.10%	10.38%	3.20%	3.42%	4.97%	25.34%	**34.81%**
VegFru	12bits	1.28%	2.36%	3.05%	5.92%	6.33%	4.60%	3.70%	8.24%	**23.55%**
	24bits	2.21%	4.04%	5.51%	11.55%	9.05%	8.91%	6.24%	24.90%	**35.93%**
	32bits	3.39%	5.65%	7.48%	14.55%	10.28%	11.23%	7.83%	36.53%	**48.27%**
	48bits	4.51%	6.56%	8.74%	16.45%	9.11%	17.12%	10.29%	55.15%	**69.30%**
Food101	12bits	1.57%	4.51%	6.46%	10.21%	11.82%	6.51%	24.42%	35.64%	**45.63%**
	24bits	2.48%	5.79%	8.20%	11.44%	13.05%	8.97%	34.48%	40.93%	**55.48%**
	32bits	2.64%	5.91%	9.70%	13.36%	16.41%	13.10%	35.90%	42.89%	**56.39%**
	48bits	3.07%	6.63%	10.07%	15.55%	20.06%	17.18%	39.65%	48.81%	**64.19%**

since both local and global features are not well learned, the part-level feature exchanging operation is disabled for avoiding aligning meaningless local features.

4.3 Comparisons with other ANN Methods

To prove the practicality and effectiveness of our proposed method, comparisons with other ANN methods are presented in this section. All experiments are conducted based on hash codes of 32bits generated by our model.

In Table 1, we present the retrieval performance on the *Food101* dataset. Specifically, we present the P@10, WC time, speedup, and memory cost for all methods. We can observe that, compared with the linear search, our method can achieve up to 233× and 395× acceleration on features of 512-D and 1024-D, respectively. The memory cost of our method is also much less than tree-based methods. The best speed-up and the lowest storage usage prove the practicality of our proposed method. Meanwhile, our method can achieve state-of-the-art retrieval accuracies, which demonstrates that our ExchNet is the most effective one compared with other ANN methods. Above results illustrate our ExchNet deserves to be the optimal choice for fine-grained image retrieval.

4.4 Comparisons with State-of-the-art Hashing Methods

In Table 2, we present the mean average precision (MAP) results for comparisons with state-of-the-art hashing methods on all datasets. From Table 2, we can observe that our method can achieve the best retrieval performance in all cases. On fine-grained datasets (*CUB* and *Aircraft*) of relatively small size, almost all the generic hashing methods (except for ADSH) can not achieve a satisfactory performance, i.e., a relatively low MAP. Also, our ExchNet outperforms the most powerful baseline ADSH considerably. It can verify that given limited training data, our proposed method could still perform well. As to large-scale fine-grained datasets, the improvements become more significant. Particularly, comparing with the most powerful baselines, we achieve 12% and 14% MAP improvements on the 32 bits and 48 bits evaluation experiments of the large-scale *VegFru* dataset. Meanwhile, we achieve 14% and 16% MAP improvements on the 32 bits and 48 bits experiments of the *Food101* dataset. It shows that, with sufficient training data, we can get better retrieval results with our ExchNet on large-scale fine-grained datasets.

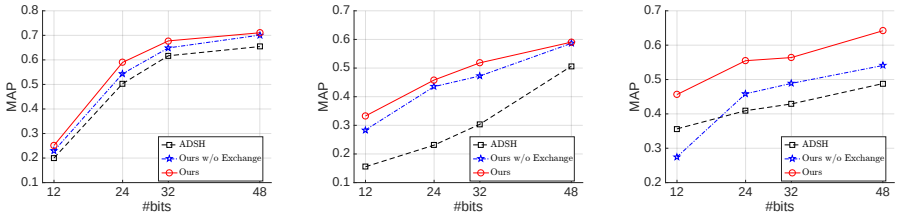
4.5 Ablation Studies

Effectiveness of the Exchanging-based Feature Alignment We verify the effectiveness of the local feature alignment approach (cf. Section 3.2) in this section. The retrieval accuracy are present in Figure 5, where "Ours w/o Exchange" means that we do not perform the feature exchanging operation (i.e., the local feature alignment) during training. Note that "Ours w/o Exchange" is degenerated to the ADSH [14] learned with our proposed representation learning architecture instead of ResNet50. Hence, we also present the results of ADSH.

It can be observed that our method can achieve the best accuracy thanks to the feature exchanging operation. Specifically, on *CUB* and *Aircraft* datasets, our proposed method with the exchanging operation performs considerably better than that without exchanging. The performance improvement on the large-scale fine-grained datasets (e.g., *Food101*) becomes more significant. Above results illustrate that our proposed local features alignment strategy is effective, especially on large-scale datasets. Moreover, even if bits of hash codes are limited, our feature alignment strategy could still benefit fine-grained retrieval greatly.

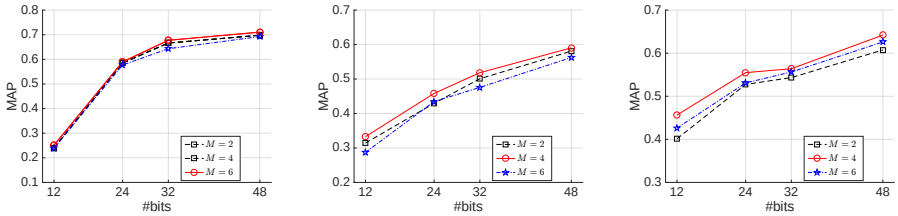
Sensitivity to Hyper Parameter M In our ExchNet, we use M to denote the number of local features, which is also the number of attention maps. In this section, we present the influence of the hyper-parameter M by ablation studies.

As presented in Figure 6, we vary M as 2, 4 and 6. From that figure, it is observed that satisfactory retrieval accuracies are achieved regardless of different M values, and the best fine-grained retrieval accuracy is obtained when $M = 4$. As analyzed, redundant local features (i.e., overmuch object parts when M is large) might cause redundancies in local feature representations, while the lack of local features (i.e., scant object parts when M is small) may result in that fine-grained images are under-represented for distinguishing subtle visual differences. Those



(a) *CUB* (b) *Aircraft* (c) *Food101*

Figure 5. Effectiveness of our feature exchanging operation.



(a) *CUB* (b) *Aircraft* (c) *Food101*

Figure 6. Influence of hyper parameter M which denotes the number of local features.

might be the reasons why M is too small or large will cause slightly accuracy drops. Moreover, comparable retrieval results of different M values show that our ExchNet is not sensitive to M.

5 Conclusions

In this paper, we studied the practical but challenging fine-grained hashing task, which aims to solve large-scale FGIR problems by leveraging the search and storage efficiency of compact hash codes. Specifically, we proposed a unified network ExchNet to obtain representative fine-grained local and global features by performing our attention approach equipped with the tailored attention constraints. Then, ExchNet utilized its local feature alignment to align these local features to their corresponding object parts across images. Later, an alternating learning algorithm was employed to return the final fine-grained binary codes. Compared with ANN methods and competing generic hash methods, experiments validated both effectiveness and efficiency of our ExchNet. In the future, we would like to explore a more challenging unsupervised fine-grained hashing topic.

Acknowledgements Quan Cui's contribution was made when he was an intern at Megvii Research Nanjing. This research was supported by the National Key Research and Development Program of China under Grant 2017YFA0700800 and "111" Program B13022. Qing-Yuan Jiang and Wu-Jun Li were supported by the NSFC-NRF Joint Research Project (No. 61861146001).

References

1. Bentley, J.L.: Multidimensional binary search trees used for associative searching. ACM Commun. **18**(9), 509–517 (1975)
2. Cakir, F., He, K., Sclaroff, S.: Hashing with binary matrix pursuit. In: ECCV. pp. 332–348 (2018)
3. Cao, Y., Long, M., Liu, B., Wang, J.: Deep cauchy hashing for hamming space retrieval. In: CVPR. pp. 1229–1237 (2018)
4. Cao, Z., Long, M., Wang, J., Yu, P.S.: Hashnet: Deep learning to hash by continuation. In: ICCV. pp. 5609–5618 (2017)
5. Chen, J., Wang, Y., Qin, J., Liu, L., Shao, L.: Fast person re-identification via cross-camera semantic binary transformation. In: CVPR. pp. 5330–5339 (2017)
6. Datar, M., Immorlica, N., Indyk, P., Mirrokni, V.S.: Locality-sensitive hashing scheme based on p-stable distributions. In: SoCG. pp. 253–262 (2004)
7. Dizaji, K.G., Zheng, F., Sadoughi, N., Yang, Y., Deng, C., Huang, H.: Unsupervised deep generative adversarial hashing network. In: CVPR. pp. 3664–3673 (2018)
8. Dolatshah, M., Hadian, A., Minaei-Bidgoli, B.: Ball*-tree: Efficient spatial indexing for constrained nearest-neighbor search in metric spaces. CoRR **abs/1511.00628** (2015)
9. Fu, J., Zheng, H., Mei, T.: Look closer to see better: Recurrent attention convolutional neural network for fine-grained image recognition. In: CVPR. pp. 4438–4446 (2017)
10. Gong, Y., Lazebnik, S.: Iterative quantization: A procrustean approach to learning binary codes. In: CVPR. pp. 817–824 (2011)
11. Horn, G.V., Branson, S., Farrell, R., Haber, S., Barry, J., Ipeirotis, P., Perona, P., Belongie, S.J.: Building a bird recognition app and large scale dataset with citizen scientists: The fine print in fine-grained dataset collection. In: CVPR. pp. 595–604 (2015)
12. Hou, S., Feng, Y., Wang, Z.: Vegfru: A domain-specific dataset for fine-grained visual categorization. In: ICCV. pp. 541–549 (2017)
13. Jégou, H., Douze, M., Schmid, C.: Product quantization for nearest neighbor search. IEEE TPAMI **33**(1), 117–128 (2011)
14. Jiang, Q.Y., Li, W.J.: Asymmetric deep supervised hashing. In: AAAI. pp. 3342–3349 (2018)
15. Li, P., Xie, J., Wang, Q., Gao, Z.: Towards faster training of global covariance pooling networks by iterative matrix square root normalization. In: CVPR. pp. 947–955 (2018)
16. Li, Q., Sun, Z., He, R., Tan, T.: Deep supervised discrete hashing. In: NeurIPS. pp. 2482–2491 (2017)
17. Li, W.J., Wang, S., Kang, W.C.: Feature learning based deep supervised hashing with pairwise labels. In: IJCAI. pp. 1711–1717 (2016)
18. Lin, J., Li, Z., Tang, J.: Discriminative deep hashing for scalable face image retrieval. In: IJCAI. pp. 2266–2272 (2017)
19. Lin, K., Yang, F., Wang, Q., Piramuthu, R.: Adversarial learning for fine-grained image search. In: ICME. pp. 490–495 (2019)
20. Lin, T.Y., RoyChowdhury, A., Maji, S.: Bilinear cnn models for fine-grained visual recognition. In: CVPR. pp. 1449–1457 (2015)
21. Liong, V.E., Lu, J., Wang, G., Moulin, P., Zhou, J.: Deep hashing for compact binary codes learning. In: CVPR. pp. 2475–2483 (2015)

22. Liu, H., Wang, R., Shan, S., Chen, X.: Deep supervised hashing for fast image retrieval. In: CVPR. pp. 2064–2072 (2016)
23. Lukas, B., Matthieu, G., Van Gool, L.: Food-101 - mining discriminative components with random forests. In: ECCV. pp. 446–461 (2014)
24. Maji, S., Rahtu, E., Kannala, J., Blaschko, M.B., Vedaldi, A.: Fine-grained visual classification of aircraft. CoRR **abs/1306.5151** (2013)
25. Neyshabur, B., Srebro, N., Salakhutdinov, R.R., Makarychev, Y., Yadollahpour, P.: The power of asymmetry in binary hashing. In: NeurIPS. pp. 2823–2831 (2013)
26. Pang, C., Li, H., Cherian, A., Yao, H.: Part-based fine-grained bird image retrieval respecting species correlation. In: ICIP. pp. 2896–2900 (2017)
27. Schroff, F., Kalenichenko, D., Philbin, J.: FaceNet: A unified embedding for face recognition and clustering. In: CVPR. pp. 815–823 (2015)
28. Shen, F., Shen, C., Liu, W., Shen, H.T.: Supervised discrete hashing. In: CVPR. pp. 37–45 (2015)
29. Wah, C., Branson, S., Welinder, P., Perona, P., Belongie, S.: The caltech-ucsd birds-200-2011 dataset (2011)
30. Wang, G., Hu, Q., Cheng, J., Hou, Z.: Semi-supervised generative adversarial hashing for image retrieval. In: ECCV. pp. 469–485 (2018)
31. Wei, X.S., Luo, J.H., Wu, J., Zhou, Z.H.: Selective convolutional descriptor aggregation for fine-grained image retrieval. IEEE TIP **26**(6), 2868–2881 (2017)
32. Wei, X.S., Wang, P., Liu, L., Shen, C., Wu, J.: Piecewise classifier mappings: Learning fine-grained learners for novel categories with few examples. IEEE TIP **28**(12), 6116–6125 (2019)
33. Wei, X.S., Xie, C.W., Wu, J., Shen, C.: Mask-CNN: Localizing parts and selecting descriptors for fine-grained bird species categorization. Pattern Recognition **76**, 704–714 (2018)
34. Weiss, Y., Torralba, A., Fergus, R.: Spectral hashing. In: NeurIPS. pp. 1753–1760 (2008)
35. Xia, R., Pan, Y., Lai, H., Liu, C., Yan, S.: Supervised hashing for image retrieval via image representation learning. In: AAAI. pp. 2156–2162 (2014)
36. Xie, L., Wang, J., Zhang, B., Tian, Q.: Fine-grained image search. IEEE Transactions on Multimedia **17**(5), 636–647 (2015)
37. Yang, Z., Luo, T., Wang, D., Hu, Z., Gao, J., Wang, L.: Learning to navigate for fine-grained classification. In: ECCV. pp. 420–435 (2018)
38. Yuan, X., Ren, L., Lu, J., Zhou, J.: Relaxation-free deep hashing via policy gradient. In: ECCV. pp. 134–150 (2018)
39. Yuan, X., Ren, L., Lu, J., Zhou, J.: Relaxation-free deep hashing via policy gradient. In: ECCV. pp. 134–150 (2018)
40. Zhang, J., Shen, F., Liu, L., Zhu, F., Yu, M., Shao, L., Heng Tao, S., Van Gool, L.: Generative domain-migration hashing for sketch-to-image retrieval. In: ECCV. pp. 297–314 (2018)
41. Zheng, H., Fu, J., Mei, T., Luo, J.: Learning multi-attention convolutional neural network for fine-grained image recognition. In: CVPR. pp. 5209–5217 (2017)
42. Zheng, X., Ji, R., Sun, X., Wu, Y., Huang, F., Yang, Y.: Centralized ranking loss with weakly supervised localization for fine-grained object retrieval. In: IJCAI. pp. 1226–1233 (2018)
43. Zheng, X., Ji, R., Sun, X., Zhang, B., Wu, Y., Huang, F.: Towards optimal fine grained retrieval via decorrelated centralized loss with normalize-scale layer. In: AAAI. vol. 33, pp. 9291–9298 (2019)
44. Zhu, F., Kong, X., Zheng, L., Fu, H., Tian, Q.: Part-based deep hashing for large-scale person re-identification. IEEE TIP **26**(10), 4806–4817 (2017)