Graph-based Discriminative Features Learning for Fine-grained Image Retrieval

Han SUN[1,2*], Wenxi Lang[1,2], Can Xu[3], Ningzhong Liu[1,2], Huiyu Zhou[4]

1 College of Computer Science and Technology, Nanjing University of Aeronautics and Astronautics, Nanjing 211106, China

2 MIIT Key Laboratory of Pattern Analysis and Machine Intelligence, Nanjing 211106, China

3 College of Computer Science and Engineering, Nanjing University of Science and Technology, Nanjing 210094, China

4 School of Informatics, University of Leicester, Leicester LE1 7RH, U.K.

Abstract: Fine-grained image retrieval has gradually become a hot topic in computer vision , which aims to retrieve images with the same subcategories from general visual categories. Though fine-grained image retrieval has made a breakthrough with the development of convolutional neural networks, its performance is still limited by the low discriminative feature embedding. To solve this problem, most prior works focus on mining more discriminative features with various strategies. In this paper, we propose a novel graph-based discriminative features learning network for fine-grained image retrieval (GDF-Net). We first design a global fine-grained feature aggregation module, which reconstructs the discriminative features through capturing context correlation based on a K-Nearest Neighbor graph. To reduce storage overhead and speed up retrieval, we further design a semantic hash encoding module, which generates a semantically compact hash code under the guidance of Cauchy quantization loss and

bit balance loss. Validated by extensive experiments and ablation studies, our method consistently outperforms state-of-the-art generic retrieval methods as well as fine-grained retrieval methods on three datasets, e.g., CUB Birds, Stanford Dogs and Stanford Cars.

Keywords: Fine-grained image retrieval; graph convolutional neural network; deep hashing

1 Introduction

As an emerging research topic, Fine-grained image retrieval (FGIR) aims to retrieve images belonging to a certain meta-category (e.g., dogs) and return images with the same subcategory (e.g., walker hound or basset) as a query image. Different from the classical image retrieval, for the FGIR task, images that belong to the different subcategories are similar to each other as the discriminative information among them is quite imperceptible (e.g., the texture of a dog's head usually determines whether it's walker hound or basset), while images within the same subcategory are also difficult to be distinguished due to the various pose, illumination, background, and shooting angle. To this end, distinguishing and retrieving fine-grained images is still a challenging research focus at present.

As a cornerstone, learning discriminative feature representation within and outside fine-grained categories plays a key role in FGIR. Earlier works[1, 2] mainly rely on the hand-craft features, while more recent studies[3-7] prefer to extract features from a pre-

trained Convolutional Neural Network (CNN). Although CNN's powerful semantic expression capabilities have significantly improved the feature quality, constructing a high-quality feature representation in FGIR is still an open problem, whose main challenge lies in efficiently mining and embedding discriminative local information. Most recent works[8-12] usually design specific localization strategies to perceive object's discriminative parts first. For instance, Selective Convolutional Descriptor Aggregation (SCDA) [13] first localizes the salient foreground region and irrelevant background noise with a pre-trained CNN, and then obtained a more representative feature embedding by introducing the flood-fill algorithm to filter the noise. ExchNet[14] and WSDAN[15] obtains parts of fine-grained objects based on the attention mechanism, which further designs channel-wise constraint and data enhancement to enhance the discriminativeness of parts, respectively. It seems that all the previous works try to locate and encode objects' discriminative parts and patches independently. However, what has been neglected is the global structure relationship among these local discriminative regions.

Existing works[16-18] have shown that the global structure benefits to improve the performance of image classification/recognition/segmentation tasks. More intuitively, as the Fig 1 shown, the local discriminative information (e.g., blue back, white abdomen, white and blue striped wings) can't support the model to distinguish and retrieve the Florida Jay from database accurately. However, if we further consider the size of the white area on the wings or the relative spatial relationship between the blue back and white area on the wings by manual search, the result will be significantly

improved. In theory, we consider that local spatial context and region correlations is rather helpful in distinguishing fine-grained images. On the one hand, by mining the context of the discriminative regions, more detailed information of the object can be encoded, which alleviates the local embedding issue caused by receptive field of convolutional neural network; on the other hand, by dynamically formulating the correlation among different independent discriminative regions, the final feature can not only retain the discriminativeness but also describe object's global structural relationship. All of the above issues motivate us to incorporate the local spatial context of independent discriminative regions and the correlation among these independent discriminative regions into the fine-grained image feature extracting.

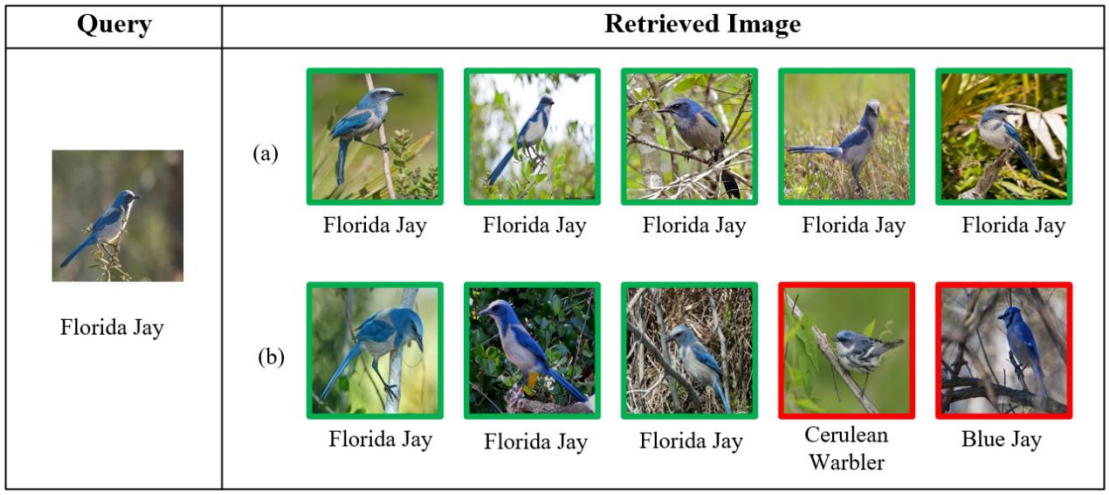


Fig. 1 The top 5 images returned. (a)based on our reconstructed feature through capturing the global spatial context correlation; (b) based on the independent local discriminative feature

Based on the above considerations, in this paper, we propose a graph-based discriminative features learning network for fine-grained image retrieval (GDF-Net), which consists of the Global Fine-grained Feature Aggregation Module (GFFAM) and Semantic Hash Encoding Module (SHECM). The GFFAM first learns attention maps

that indicating objects' discriminative parts. For each part, the GFFAM regards it as the node and constructs a graph convolutional network. By adopting a propagation strategy inspired by the K-Nearest Neighbor, the GFFAM finally learns the dynamic connection relationship among different nodes to aggregate the local discriminative features into global fine-grained features. In order to improve the retrieval speed, we also design a SHECM, which restricts the network to learn compact and high semantic hash codes based on the global fine-grained features. While ensuring the semantics of hash encoding based on classification loss, the SHECM further introduces Cauchy cross-entropy loss and bit balance loss, which can improve the compactness of the hash code and realize efficient Hamming space retrieval. In summary, the contributions of this paper are as follows:

(1) We propose an adaptive local feature fusion method based on a graph neural network. In the dissemination of the graph, the correlation among the discriminative part features can be captured effectively, which helps guide the fusion of different fine-grained part features to improve the final feature quality.

(2) We propose an extra semantic hash encoding module in the fine-grained image retrieval task, which outputs a compact hash code to improve the retrieval performance by combining the classification loss with Cauchy cross-entropy loss and bit balance loss.

(3) We evaluate the proposed model on three fine-grained benchmarks. Extensive experiments indicate our method achieves the best performance in retrieval accuracy.

2 Relate work

2.1 Fine-grained image retrieval

Fine-grained image retrieval is an emerging research hotspot that has attracted more and more research attention in recent years. Unlike general image retrieval tasks, fine-grained image retrieval aims to distinguish subcategory, which has two characteristics between database images and query images: (1) small inter-class variance. Differences among most classes are subtle and difficult to distinguish. (2) large intra-class variance, which is due to factors such as poses and viewpoints.

In the early days, FGIR relied on the use of manual features[19], and as deep learning method evolved, more and more FGIR methods were proposed. And these deep methods can be roughly divided into supervised and unsupervised methods.

For unsupervised retrieval methods, SCDA[13] proposes selective convolutional descriptor aggregation method, which first localizes objects in fine-grained images and retains useful deep descriptors for fine-grained image retrieval.

In supervised methods, CRL-WSL[20] presents a unified framework for efficient learning of discriminative features with centralized ranking loss and segmentation of target contours using learning saliency regions. DCL-NC[19] improves CRL-WSL[20] by adding a normalize-scale layer and decorrelated ranking loss. However, these two methods encode high dimensionality and may encounter problems of slow query speed and storage redundancy in practical large-scale image retrieval. ExchNet[14] uses hash learning to provide compact binary codes for fine-grained images, and designs the operation of feature swapping to make local features aligned. [21] adds a hash layer

before the classification layer as a feature-switching layer to guide the classification and achieves state-of-the-art results.

2.2 Graph convolutional networks in computer vision

Graph convolutional network(GCN)[22], a neural network that applies graph structures to learn the representation of a target node by iteratively propagating neighborhood information, has been widely used in various computer vision tasks[23-26].

In the field of object detection, Structural-RNN[23] designs a dual graph neural network that combines temporal and spatial relationships among entities to construct graphs that perform graph convolution from different views of the raw data. In the field of image classification, Lee et al.[24] introduces knowledge graphs to zero-sample learning tasks and achieves some improvement in multi-label tasks. In the field of semantic segmentation, Graph Long Short Time Memory(Graph LSTM)[25] network proposes a generalization of LSTM from sequential or multidimensional data to general graph-structured data. Wang at el.[27] uses graph convolutional neural networks on point clouds and proposes to collect features of edges by edge convolution, which contains neighborhood information, and global geometric features can be learned by stacking.

For the task of FGIR, to the best of our knowledge, few works consider introducing the graph convolutional neural networks, while it used to not only express discriminative features but also capture the complex spatial correlations among different local features in our GFFAM.

2.3 Hashing method

Current hashing methods can be classified into two categories, containing data-independent hashing and data-dependent hashing. Data-independent hashing refers to binary hash codes constructed by random projection or by hand, such as locally sensitive hash functions[28], and LSH methods usually require a long code length to guarantee retrieval performance.

Based on whether supervised information is used or not, data-dependent hashing can be classified as unsupervised and supervised hashing. For example, [29] and [30] are classical unsupervised hashing methods. Although unsupervised methods are less restrictive and more practical, exploiting the available supervised information implies better performance.

Thanks to the powerful representation capability of deep neural networks, deep models have been used for supervised hashing, proposing the synthesis of feature learning and hash coding into end-to-end deep hashing methods[31-37]. Convolutional neural network hashing (CNNH)[31] and deep pairwise supervised hashing (DPSH)[34] are representative approaches. HashNet[33] proposes to continuously approximate the sign function with the tanh activation function, thus solving the optimization complexity problem and greatly improving the retrieval performance. However, existing hashing methods are based on a linear scan of the hash to maximize the retrieval performance, thus incurring expensive overhead. Therefore, Deep Cauchy Hashing (DCH) [34] proposes a pairwise cross-entropy loss based on Cauchy distribution, which learns almost lossless hash codes by significantly penalizing similar

image pairs with Hamming distance greater than a given Hamming radius threshold. Other recent work includes greedy hash [38], Asymmetric Deep Supervised Hashing (ADSH) [39], etc.

3 Proposed Method

3.1 Overview

We propose an end-to-end Graph-propagation based discriminative features learning network for Fine-grained image retrieval. As shown in Fig 2, it consists of two key modules: the Global Fine-grained Feature Aggregation Module (GFFAM) and Semantic Hash Encoding Module (SHECM). The GFFAM explores the interdependencies among feature vectors based on a graph convolutional network, which will guide the fusion of independent discriminative elements to enhance the global fine-grained features (Section 3.2). The global features are combined in the SHECM to learn a hash embedding for the final retrieval, which we will explain in Section 3.3.

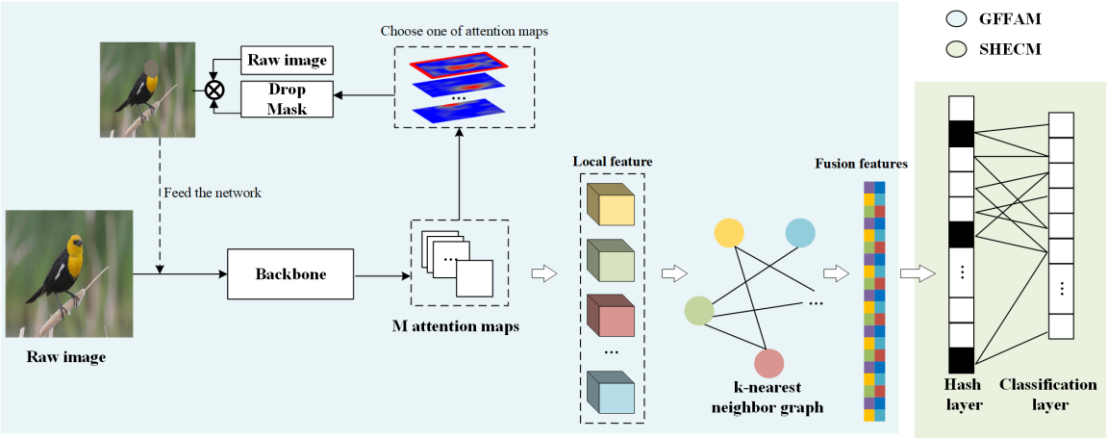


Fig.2 Framework of the proposed GDF-Net

3.2 Global Fine-grained Feature Aggregation Module

In order to ensure the discriminative ability of feature while taking both global image-level context and interaction among local features into account, we construct a graph convolution network to aggregate independent local features into global fine-grained features iteratively. The proposed GFFAM includes the following two stages. The first stage is that extracts local discriminative features which server as the node of the graph convolution network. In the second stage, the graph updates the representation of each node and propagates them to the top level for aggregation. It should be noted that, by calculating the link relationship among different nodes in the graph, the global context information and the interaction among local discriminative features are finally encoded together as the node features propagate along the connection direction.

3.2.1 Node representation based on local feature

The discriminative local features are quite important for fine-grained tasks. In constructing k-nearest neighbor graph, local features of fine-grained object are used as nodes of the graph. Therefore, we first extract the fine-grained parts with discriminative features. Specifically, in our mission with a weakly supervised image-level label, the location annotation (e.g., bounding boxes or key points) which can indicate instances' important parts is not available during the model training and testing, so we extract the discriminative local features based on attention generation strategy by calculating the category response of different parts.

Given the image X and its feature $F \in R^{H \times W \times N}$, where H, W and N represent the height, width and number of channels of the feature layers, respectively. We generate M attention maps for each image and the $A_k \in R^{H \times W}$ means the attention map of part k, which may be the wing of a bird or the head of a dog. We use a simple convolution function $f()$ to transform the feature map to attention map, which can be described as a spatial attention mapping indicating the response between original channel-wise features and category labels. In detail, the structure of f() consists of a convolution layer with kernel 1×1, a Batch Normalization layer and a Relu activation layer. The calculation method of the attention map is shown in formula (1).

$$A = f(F) = \bigcup_{k=1}^{M} A_k \qquad (1)$$

With the attention map of different local regions, we then extract the discriminative local features from these parts. Similar to the Bilinear pooling[8, 44], the M part features corresponding to these local regions can be calculated by formula (2).

$$f_k = g(A_k \odot F),\ k = 1, 2, \cdots, M \qquad (2)$$

Where f_k represents the k_{th} local feature and the $\odot$ represents the element-wise multiplication of feature map F and the k_{th} attention map, while $g()$ is the global average pooling operation.

3.2.2 Local feature enhancement

For the learned M spatial attention maps, a common phenomenon is that multiple attentions maps may focus on object's similar regions, which will greatly inhibit the diversity of the discriminative features. We further introduce an additional random dropping strategy to force the network to search for other informative local regions.

Specifically, for each training image, we first randomly select one attention map A_k from A. To improve the convergence rate of the model, the min-max normalization is adopted to smooth the value of A_k to the range of $[0, 1]$, as shown in equation (3).

$$A_k^* = \frac{A_k - \min(A_k)}{\max(A_k) - \min(A_k)} \qquad (3)$$

Here, the A_k^* represents the k_{th} attention map after augmentation. We then construct a drop mask M_d through setting the value of elements larger than the threshold of $T_d \in [0, 1]$ to 0 and the value of other elements to 1, as:

$$M_d(i,j) = \begin{cases} 0, & A_k^*(i,j) > T_d \\ 1, & otherwise \end{cases} \qquad (4)$$

where the $A_k^*(i,j)$ represents the value of element in the i_{th} row and j_{th} column belonging to the k_{th} local feature selected, and the $M_d(i,j)$ represents the value of drop mask in the corresponding position. Here, the threshold T_d is set to 0.5. With the drop mask M_d and the original image, the new masking image X_d can be obtained by an element-wised multiplication, which will be fed into the network once again and learn M new part features f_k^d. With the attention drop, we encourage the attention map to propose other discriminative parts and finally improve the accuracy of localization and the quality of feature.

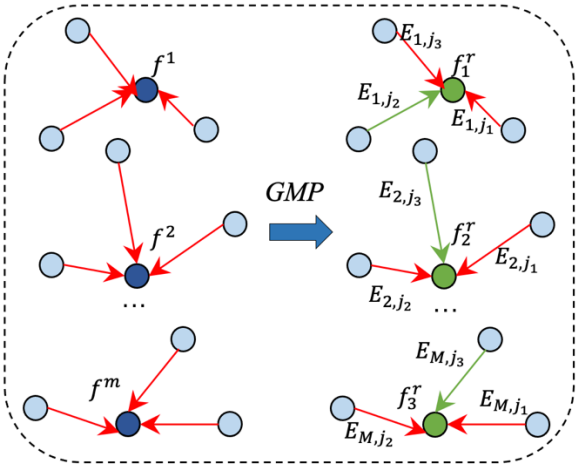


Fig.3 Example of k-nearest neighbor graph (green edge indicates the selected neighbor node).

3.2.3 Interdependency mining based on Graph-Propagation

With M local discriminative features $f_1, f_2, \ldots f_M$, we construct a directed graph $\mathrm{G} = (\mathrm{V}, \mathrm{E})$ to capture the contextual relationship among these discrete local features, which process can be described as calculating *k-nearest* neighbor of each graph node. Since the graph contains self-loops where each node also points to itself, the feature of each graph node can be updated by aggregating the features of its *k-nearest* neighbor node. Under the supervision of category label, we finally obtain M reconstructed features with the propagation of graph, which can express the contextual structural relationship among different local features while retaining high semantic discrimination.

Specially, given vertices $V = \{1, 2, \ldots, M\}$ and edges $E \subseteq V \times V$, the edge feature from i_{th} vertex to j_{th} vertex can be defined as:

$$e_{ij} = h_{\theta;\sigma}(f_i, f_i - f_j) \qquad (5)$$

$h_{\theta;\sigma}()$ is an asymmetric edge function implemented through a shared Multilayer Perceptron (MLP), which consists of a convolution layer with 1×1 kernel, a Batch Normalization layer and a Relu layer, as:

$$h_{\theta;\sigma}(f_i, f_i - f_j) = Relu(\theta \cdot f_i + \sigma(f_i - f_j)) \qquad (6)$$

Where θ and σ represent parameters of the network. As shown in equation (6), the neighborhood information from different nodes captured by $\mathrm{f_i} - \mathrm{f_j}$ can be combined with the global information captured by $\mathrm{f_i}$ gradually. Similar to image convolution, the output of i_{th} vertex can be obtained by applying a global max pooling (GMP) operation on all edge features associated with the i_{th} vertex, as:

$$f_i^r = \underset{j:(i,j)\in E}{GMP} e_{ij} \quad (7)$$

3.3 Semantic Hash Encoding Module

In the most of deep hashing methods, the hash layer is designed to encode the features and output a binary hash code, in which '1' indicates the category possesses a certain feature while '0' means it lacks this feature. Once we obtain the high discriminative features which can express the fine-grained categories, what we need to do is just encoding these features into compact hash coding to improve the retrieval efficiency. As shown in Fig.2, different from existing works[45], we design and add a semantic hash coding module before the classification layer.

For the M reconstructed context features f_i^r, the SHECM outputs a B bit hash code which can be computed as the formula (8).

$$H_i = tanh((W^H)^T f_i^r + \delta^H), \quad i = 1, 2, \cdots, M \quad (8)$$

Where $H_i \in R^B$ is the output of the hash layer with the f_i^r. The $\delta^H \in B$ and $W^H \in R^{M\times B}$ represents the bias and the weights of the hash layer, respectively. The $tanh()$ represents the active function which can be described as the formula (9).

$$\tanh(f^r) = \frac{e^x - e^{-x}}{e^x + e^{-x}} \quad (9)$$

The final hash codes can be obtained according to the formula (10). Since the value range of the $tanh()$ is $[-1,\ 1]$, the $B_i = 1$ if the $H_i \geq 0$ and $B_i = -1$ otherwise. Since the gradient of the sign function at the non-zero point may be zero, the problem of gradient vanishing arises, we only map the real-valued output H_i to hash code when test the model.

$$B_i = sgn(H_i) \quad (10)$$

3.4 Loss function

3.4.1 Loss of GFFAM

When learning the discriminative features in section 3.2.1, it's necessary that each attention map A_k belonging to same category can point to objects' similar part region. Inspired by the center loss of face recognition[37, 46], we first introduce a loss L_{ctr} to learn a feature center c_k for each local discriminative feature f_k. L_{ctr} penalizes the variances of features which comes from same parts of different objects with the same category label, which can be formulated by the equation (11).

$$L_{ctr} = \sum_{k=1}^{M+1} ||f_k - c_k||^2 \qquad (11)$$

Where c_k is the part's feature center of A_k and can be initialized from zero and updated by moving average $c_k = (1-\mu)c_k + \mu f_k$. Here, μ controls the update rate of c_k and L_{ctr} loss applies only to the original image.

Since the whole learning process of GFFAM is supervised by the category label, we adopt the cross-entropy loss L_{ce} to constraint the distance among the predicted category and the real image-label Y^*. For M original local feature f_k, we simply stack them together and then feed in a SoftMax layer to predict its category probability Y_{ori}. The loss can be calculated as:

$$L_{ori}(Y_{ori}, Y^*) = L_{ce}(Softmax(\begin{bmatrix} f_1 \\ f_2 \\ \vdots \\ f_m \end{bmatrix}),\ Y^*) \qquad (12)$$

Similarly, we also predict the category probability Y_{drop} and Y_{recn} for the M dropping feature ${f_k}^d$ and M reconstructed context feature ${f_k}^r$, respectively. The total classification loss composes of the L_{ori}, L_{drop} and L_{recn} as the formula (13)

shown.

$$L_{cls} = L_{ori}(Y_{ori}, Y^*) + L_{drop}\left(Y_{drop}, Y^*\right) + L_{recn}(Y_{recn}, Y^*) \quad (13)$$

3.4.2 Loss of SHECM

Previous deep hashing methods have used the Sigmoid function to define the probability function, however, existing hashing methods usually lack the ability to concentrate the relevant images within a small Hamming sphere, so they may perform poorly for Hamming space retrieval. Therefore, inspired by DCH[35], we use a Bayesian framework to optimize the quantization loss. We use the probability function, as:

$$\sigma\left(d\left(\boldsymbol{h}_i, \boldsymbol{h}_j\right)\right) = \frac{\gamma}{\gamma + d\left(\boldsymbol{h}_i, \boldsymbol{h}_j\right)} \quad (14)$$

where γ is the scale parameter of the Cauchy distribution. When the Hamming distance is small, this function decreases rapidly, resulting in similar points being pulled into a small Hamming radius. For a pair of binary hash codes $\boldsymbol{h}_i$ and $\boldsymbol{h}_j$ the Hamming distance is:

$$d\left(\boldsymbol{h}_i, \boldsymbol{h}_j\right) = \frac{K}{4}\left\|\frac{\boldsymbol{h}_i}{\|\boldsymbol{h}_i\|} - \frac{\boldsymbol{h}_j}{\|\boldsymbol{h}_j\|}\right\|_2^2 = \frac{K}{2}\left(1 - cos\left(\boldsymbol{h}_i, \boldsymbol{h}_j\right)\right) \quad (15)$$

Then, the Cauchy quantization loss is deduced as

$$L_q = \sum_{i=1}^{N} log\left(1 + \frac{d(|\boldsymbol{h}_i|, \mathbf{1})}{\gamma}\right) \quad (16)$$

In addition to reducing the quantization error, our approach takes into account the bit-balancing property, which means that each bit of the hash code has about a 50% chance of being 1 or -1. To produce more discriminative rather than ambiguous hash codes, we add the bit-balancing loss.

$$L_b = \sum_{i=1}^{M} \frac{1}{K} \sum_{j=1}^{K} H_i \quad (17)$$

where K is the dimensionality of vector H. The purpose of bit balance loss is to generate unbiased informative hash codes. Then, the hash loss L_{hash} is given by equation (18).

$$L_{hash} = L_q + L_b \qquad (18)$$

In summary, the total loss includes the loss of GFFAM and SHECM as the formula (19) shown.

$$L = L_{ctr} + L_{cls} + L_q + L_b \qquad (19)$$

4 Experiments

4.1 Datasets and Evaluation Metric

We conducted experiments on three classic fine-grained datasets, CUB200-2011[47], Stanford Dog[48] and Stanford Cars[49], and compare them with other fine-grained retrieval methods.CUB200-2011 contains 200 bird species with 11788 images, in which 5994 images are for training and 5794 images are for testing. Stanford dog contains 120 dog species and 20580 images, in which 12000 images are for training and 8580 images are for testing. Stanford cars contains 196 car species and 16185 images, in which 8144 images are for training and 8041 images are for testing. We follow the experimental protocol in [50], using the test images as the query set and the training images as the retrieval database for all images.

We use retrieval accuracy and Precision-Recall curve as evaluation metrics for a fair comparison with previous methods. We calculate the retrieval precision as:

$$mAP = \frac{1}{n_q}\sum_{i=1}^{n_q} AP,\ AP = \frac{1}{N_+}\sum_{k=1}^{n} \frac{{N_+}^k}{k} pos(k) \qquad (20)$$

Where n_q denotes the number of samples in query set, n denotes the number of top

returned samples. The N_+ represents the number of positive samples in the n returned samples, and the ${N_+}^k$ refers to the number of positive samples in the top k returned samples. The value of $pos(k)$ is 1 if the image at position k is positive while will be 0 otherwise.

Precision-Recall curve measures the relationship between precision and recall. Specially, for the image retrieval learning, precision represents the proportion of positive samples in all returned samples while recall refers to the proportion of correctly retrieved samples to all positive samples in the database, which can be calculated by the following formula (21) and (22).

$$Precision = \frac{TP}{TP+FP} \qquad (21)$$

$$Recall = \frac{TP}{TP+FN} \qquad (22)$$

TP, FN, and FP in formula (21) and (22) denote the positive samples of the correct classification, the positive samples of the misclassification and the negative samples of misclassification, respectively.

4.2 Implementation Details

For the GFFAM method, we implement the code based on PyTorch and train the models by preprocessing images to size 448×448 with 2 RTX 2080ti GPU. In all our experiments, we use the Resnet-50 as the backbone to extract the features and choose the output of Conv5 layers as feature maps. For the generation of attention maps, the attention maps are obtained by a 1×1 convolutional kernel with default M of 32, while the dropping threshold T_d is set to 0.5. For building k-nearest neighbor graph, the number k of nearest neighbors is 3.

Although all data sets are labeled with bounding boxes or part locations, our approach uses only category labels as monitoring information. In the training stage, the model is trained for 160 epochs by using the Stochastic Gradient Descent (SGD) with the momentum of 0.9 and batch size of 12. It's worth mentioning that the initial learning rate is set to 0.001, which has an exponential decay of 0.9 after every 2 epochs. In the inference stage, the drop branch is not used, we follow the experimental protocol in [50], using test images as the query set and training images as the retrieval database for all experiments.

4.3 State-of-the-arts Comparison

Comparisons with Classic Retrieval Algorithms. Experiments in this section, we first compare our approach with other state-of-the-art classic retrieval algorithms. For the sake of fairness, we train all the model using Resnet-50 as feature extractor with different bits of hash code on three fine-grained image datasets.

From Table 1, we can see that our GDF-Net achieves a mAP of 80.43%, 86.01% and 86.92% with 16-bit, 32-bit and 48-bit hash codes on CUB 200-2011 data set, outperforming the previous methods by a large margin in all hash bits. Compared with the second-place method of FPH, the mAP of GDF-Net increases by 29.15%, 27.69% and 25.13%, respectively. For the Stanford Dogs and Cars data set, the significant improvement trend is also obvious. The huge gap in results strongly shows the difference among the fine-grained image retrieval and common image retrieval, which also proves that specially designed discriminative feature extractor is necessary for the fine-grained image retrieval. In addition, compared with classic retrieval methods

which build hash code based on the global image feature, our GDF-Net shows a better stability for the change of hash code bits. For example, when the hash code decreases from 48-bit to 16-bit, the mAP of GDF-Net only decreases by about 6% on the CUB 200-2011 data set, while FPH decreases by more than 10%. This result demonstrates the effectiveness of the proposed method, and can further suggest that it is more efficient by mining the fine-grained local features to build the hash code.

Table 1 Mean average precision (mAP) of different hashing retrieval methods on three Fine-grained image datasets

Method	CUB 200-2011			Stanford Dogs			Stanford Cars		
	12bit	32bit	48bit	12bit	32bit	48bit	12bit	32bit	48bit
DHN	0.3711	0.4172	0.4602	0.4559	0.5290	0.5736	0.4608	0.5050	0.5574
DQN	0.3789	0.4355	0.4811	0.4676	0.5234	0.5795	0.4897	0.5444	0.5821
HashNet	0.4027	0.4712	0.5103	0.4988	0.5574	0.5981	0.5073	0.5508	0.5832
DCH	0.4602	0.5233	0.5740	0.6081	0.6567	0.6779	0.5488	0.6009	0.6175
FPH	0.5128	0.5832	0.6179	0.6312	0.6909	0.7090	/	/	/
Ours	0.8043	0.8601	0.8692	0.8193	0.8586	0.8664	0.8864	0.9181	0.9268

Comparisons with Fine-grained Retrieval Algorithms. We also conduct analytical experiments to evaluate the retrieval performance with other similar fine-grained retrieval algorithms, where the mAP results on CUB 200-2011 have been presented in Table 2. From Table 2, we can clearly observe that our approach achieves the best retrieval performance in all cases. Compared with previous works such as SCDA, CRL-WSL and DCI-NS without hash module, GDF-Net significantly improved the mAP even if the dimensionality of feature representation is quite smaller. We consider that the possible reason may be the sensitivity of fine-grained tasks to feature quality. Though the high dimensional image features contain more objects' information, it also means that the discriminative features can't play a leading role when learning hash code in Hamming space, which makes the hash codes and final retrieval results

more susceptible to be disturbed by similar objects of the same or different categories. Compared with the hash-based FGIR methods such as ExchNet and FCAENet, the best mAP of our GDF-Net outperforms by more than 15% when the hash code is 48bit. In addition, we also notice that the performance of ExchNet and FCAENet changes greatly with the hash bits, which is improved by more than 40% when the hash code increases from 12-bit to 32-bit. While for our GDF-Net, the mAP only increases by 6% and even increases less than 1% when hash code changes from 32-bit to 48-bit. The more stable results not only show the compactness and efficiency of our hash code but also verify the effectiveness of our module GFFAM from the side.

To the best of our knowledge, [21] is the closest work related to our GDF-Net. As shown in Table 2, we achieve an mAP of 86.92% with the 48bit hash code, which is 1.43% higher than [21]. And when the hash codes reduce to 32-bit, we also outperform 1.75%. We suppose the main reason for the improved results comes from the acquisition of higher quality fine-grained image features through mining the context correlation among different discriminative features, which we will analyze more detailed in the next section. Furthermore, for the sake of verifying the superiority of our method, we also evaluate our method with precision curves of 48-bit hash codes w.r.t different recall rates as shown in Fig. 4. And the larger the area enclosed by the PR curve and axes, the better retrieval performance. Experimental results on three datasets show that our GDF-Net outperforms all the methods.

Table 2 mAP of GDF-Net compared with Fine-grained image retrieval methods

Method	Model	#Dim	mAP
SCDA	VGG16	4096	0.5957

CRL-WSL	VGG16	1024	0.6590
DCI -NS	Resnet-50	1024	0.6790
ExchNet	Resnet-50	12	0.2514
		32	0.6774
		48	0.7105
FCAENet	Resnet-50	12	0.3476
		32	0.7385
		48	0.8014
[21]	Resnet-50	12	0.7901
		32	0.8426
		48	0.8549
Ours	Resnet-18	12	0.7628
		32	0.8165
		48	0.8244
	Resnet-50	12	0.8043
		32	0.8601
		48	0.8692

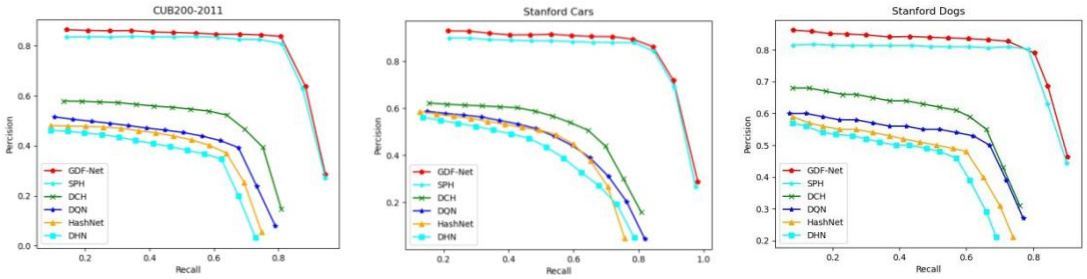


Fig. 4 Precision-recall curve with topN@48 bits on three Fine-grained image datasets

4.4 Ablation Studies

Quantitative Evaluations of GCN. As we have discussed in Section 3.2, GDF-Net constructs a KNN graph convolutional neural network to capture the correlation between different local features, which aims at reconstructing highly discriminative and contextual features for hash encoding. To evaluate the effectiveness of our KNN graph, we further conduct a set of experiments to compare the mAP between GCN and other correlation mining strategies quantitatively.

As shown in Table 3, our baseline is directly feeding M original local features f_k

into the semantic hash encoding module, which achieves a mAP of 76.84% and 82.02% for 12-bit and 48-bit on the most difficult CUB data set. To aggregate local features, we also report the results of some additional strategies, containing the application of a global average pooling(GAP)and a global max pooling(GMP) operation on all local features f_k, which achieves a significant improvement in retrieval performance. For example, GMP obtains a mAP of 84.96% with a 48-bit hash code on CUB, which is over 2% higher than the baseline. All the results verify our hypothesis that aggregating independent discriminative features are conducive to improving the feature quality and performance of fine-grained retrieval task. For our GCN, we achieve a mAP of 86.27% and 86.92% on CUB200-2011 when reconstructing the higher quality features with global average pooling and global max pooling, respectively. Compared with simple GAP/GMP, we achieve a ~2% improvement. These results show the effectiveness of our GCN which reconstruct feature under the guidance of discriminative features' *k-nearest* neighbor relationship.

Table 3 Comparison of mAP between GCN and other correlation mining strategies

Correlation mining strategy	CUB 200-2011		Stanford Dogs		Stanford Cars	
	12bit	48bit	12bit	48bit	12bit	48bit
baseline	0.7684	0.8202	0.7721	0.8234	0.8401	0.8876
GAP	0.7743	0.8325	0.7836	0.8367	0.8495	0.8941
GMP	0.7862	0.8496	0.7879	0.8482	0.8603	0.9004
GCN with GAP	0.8012	0.8627	0.8122	0.8601	0.8796	0.9203
GCN with GMP	0.8043	0.8692	0.8193	0.8664	0.8864	0.9268

Effectiveness to GCN structure. When constructing GCN, we use local discriminative features as the input of graph nodes. The initial number of nodes in the

graph network will significantly affect model performance. For this reason, we analyze the influence of nodes numbers by ablation studies on three Fine-grained benchmarks. As shown in Table 4, more initial nodes usually contribute to better performance. On CUB 200-2011 data set, GDF-Net achieve a mAP of 85.42% with 4 initial graph nodes, while the mAP increases to 86.75% and 86.92% when the initial graph nodes are set to 16 and 32. The similar rising trends is also shown on other two data sets. Considering the balance between the computational overhead and retrieval performance, we finally choose to set the number of graph nodes as 32 in our GDF-Net.

Table 4 Comparison of mAP with different numbers of initial nodes on three Fine-grained benchmarks

Numbers of initial nodes	CUB 200-2011	Stanford Dogs	Stanford Cars
4	0.8542	0.8487	0.9124
8	0.8603	0.8571	0.9196
16	0.8675	0.8652	0.9241
32	0.8692	0.8664	0.9268
64	0.8695	0.8664	0.9269

Effectiveness of Hash loss. Another important module in our GDF-Net framework is the semantic hash encoding, which designs a semantic hash coding module before the classification layer with the supervision of L_q and L_b. In order to verify the effectiveness of different losses for hash coding, we conduct a set of experiments to quantitatively evaluate the retrieval results when combining different loss functions, as shown in Table 5. For CUB, the simple baseline trains hash model without any loss, whose mAP can increase to 82.34% and 85.65% with 48-bits hash code after introducing the L_q and L_b. For our SHECM with 48-bits hash code, it can finally obtain 86.92% on CUB, 86.64% on Stanford Dogs and 92.68% on Stanford Cars. All the results show that Cauchy quantization loss and bit-balancing loss are beneficial

to generate compact hash codes.

Table 5 Comparison of mAP with different losses on three Fine-grained benchmarks

L_q	L_b	CUB 200-2011		Stanford Dogs		Stanford Cars	
		12bit	48bit	12bit	48bit	12bit	48bit
/	/	0.7123	0.8028	0.7583	0.7911	0.7867	0.8259
/	√	0.7795	0.8234	0.7689	0.8018	0.8336	0.8617
√	/	0.7998	0.8565	0.8006	0.8412	0.8719	0.9013
√	√	0.8043	0.8692	0.8193	0.8664	0.8864	0.9268

Influence of T_d and K. T_d is the dropping threshold for masking image generation, while the K is the number of neighbor nodes aggregated for graph propagation. To quantitatively analyze the impact of the above two hyperparameters on retrieval performance, we conduct additional ablation experiments on the CUB dataset and Resnet50 with 48-bits hash codes, whose results are shown in the Table 6 and Table 7, respectively. When T_d changes from 0.1 to 0.9 with a stride of 0.1, the mAP increases from 79.65% to the peaks of 86.92% and then gradually decreases to 80.31%, which show the results are sensitive to the dropping threshold. Too small threshold will discard most of the area on the original image and only remain background pixels, which is not conducive to finding other sub-discriminator features. While too large threshold makes the mined discriminative features almost unchanged, which will inhibit the diversity of local features.

Table 6 Influence analysis of T_d with Resnet50 and 48-bits hash codes on CUB

T_d	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9
mAP	0.7965	0.4689	0.7250	0.8468	0.8692	0.8620	0.8384	0.8252	0.8031

Similar to the changing trend of the dropping threshold, it turns out that considering too much neighbor node information will drastically increase the difficulty of propagation and convergence of the graph convolutional module, whose mAP will

decrease from 86.92% to 84.87% when K changes from 3 to 6. While set a smaller $K = 2$, the results will decrease about 1%, which shows the effectiveness of our GFFAM considering the spatial context information of different nodes.

Table 7 Influence analysis of K with Resnet50 and 48-bits hash codes on CUB

K	2	3	4	5	6
mAP	0.8506	0.8692	0.8681	0.8553	0.8487

4.5 Visualization Results

Visualization of Top 10 Results. Fig.5 shows the top 10 samples returned by our GDF-Net and other retrieval algorithms, including DCH[35] and Fine-grained hash retrieval algorithm [21] on the CUB Birds, Stanford dogs and Stanford Cars. Thanks to our powerful fine-grained feature extraction module, GDF-Net yields much more relevant and user-desired retrieval results than the state-of-the-art method.

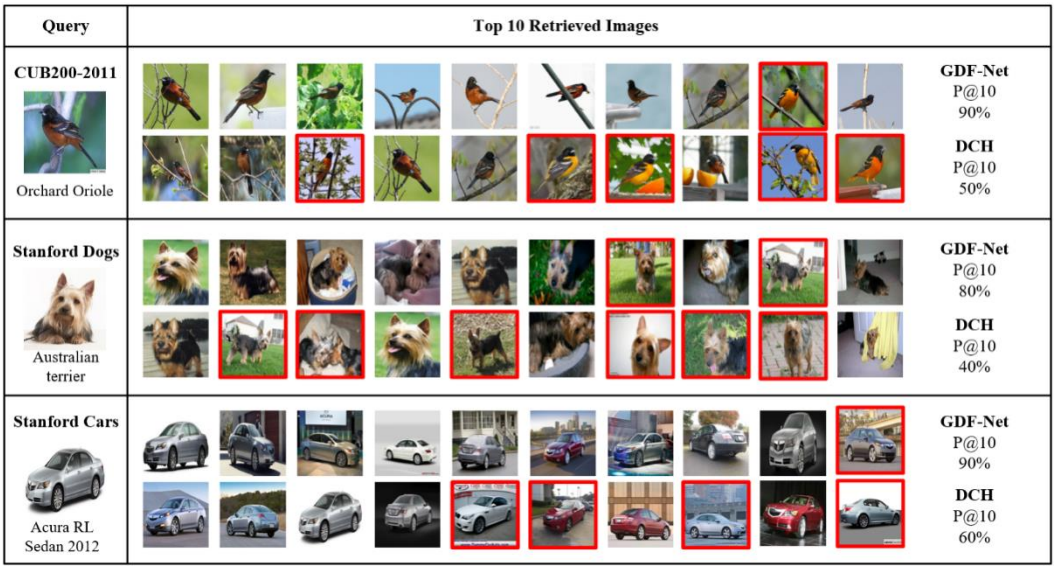
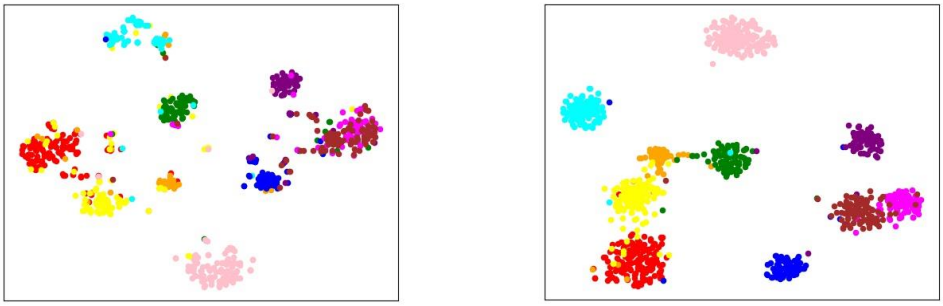


Fig.5 Examples of top 10 retrieved images and precision @ 10 on three datasets.

Visualization of hash code compactness. In order to analyze the quality of hash code, we visualize the hash code generated by GDF-Net and DCH using the t-distributed stochastic neighbor embedding (t-SNE) on Stanford Cars. As the Fig.6

shown, we sample 10 categories and find that the results of our method are better, because the results returned by DCH focus more on the corresponding global information, while the GEFAM module in GDF-Net designs a specialized feature extractor for fine-grained images.



(a) DCH (b) Our

Fig.6 The t-SNE of hash codes learned by GDF-Net and DCH on Stanford Cars

5 Conclusion

In this paper, we propose a novel deep hash fine-grained retrieval method based on graph convolutional neural network, which consists of Global Fine-grained Feature Aggregation Module (GFFAM) and Semantic Hash Encoding Module (SHECM). GFFAM constructs a *k-nearest* neighbor graph to capture the correlation among different independent discriminative features, which can guide the fusion of these local features adaptively to enhance the feature quality. To reduce the storage overhead, GFFAM designs a hash encoding layer before the final classification layer, which can generate semantic and compact hash codes with the guidance of Cauchy quantization loss and bit-balancing loss. Extensive experimental results on CUB Birds, Stanford Dogs and Stanford Cars demonstrate that our proposed method is comparable to the

state-of-art methods for fine-grained retrieval task. In the future, as we notice that fine-grained images tend to have high intra-class variances and low inter-class variances, we will further strengthen learning tolerance to improve the retrieval accuracy.

Reference

[1] L. Xie, J. Wang, B. Zhang, and Q. Tian, "Fine-grained image search," *IEEE Transactions on Multimedia*, vol. 17, no. 5, pp. 636-647, 2015.

[2] K. Lin, F. Yang, Q. Wang, and R. Piramuthu, "Adversarial learning for fine-grained image search," in *2019 IEEE International Conference on Multimedia and Expo (ICME)*, 2019: IEEE, pp. 490-495.

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

[4] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 770-778.

[5] C. Szegedy *et al.*, "Going deeper with convolutions," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2015, pp. 1-9.

[6] S. Jin, H. Yao, X. Sun, S. Zhou, L. Zhang, and X. Hua, "Deep saliency hashing for fine-grained retrieval," *IEEE Transactions on Image Processing*, vol. 29, pp. 5336-5351, 2020.

[7] J. Wang, T. Zhang, N. Sebe, and H. T. Shen, "A survey on learning to hash," *IEEE transactions on pattern analysis and machine intelligence*, vol. 40, no. 4, pp. 769-790, 2017.

[8] T.-Y. Lin, A. RoyChowdhury, and S. Maji, "Bilinear cnn models for fine-grained visual recognition," in *Proceedings of the IEEE international conference on computer vision*, 2015,

pp. 1449-1457.

[9] P. Li, J. Xie, Q. Wang, and Z. Gao, "Towards faster training of global covariance pooling networks by iterative matrix square root normalization," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2018, pp. 947-955.

[10] Z. Yang, T. Luo, D. Wang, Z. Hu, J. Gao, and L. Wang, "Learning to navigate for fine-grained classification," in *Proceedings of the European Conference on Computer Vision (ECCV)*, 2018, pp. 420-435.

[11] H. Zheng, J. Fu, T. Mei, and J. Luo, "Learning multi-attention convolutional neural network for fine-grained image recognition," in *Proceedings of the IEEE international conference on computer vision*, 2017, pp. 5209-5217.

[12] J. Fu, H. Zheng, and T. Mei, "Look closer to see better: Recurrent attention convolutional neural network for fine-grained image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 4438-4446.

[13] X.-S. Wei, J.-H. Luo, J. Wu, and Z.-H. Zhou, "Selective convolutional descriptor aggregation for fine-grained image retrieval," *IEEE Transactions on Image Processing*, vol. 26, no. 6, pp. 2868-2881, 2017.

[14] Q. Cui, Q.-Y. Jiang, X.-S. Wei, W.-J. Li, and O. Yoshie, "ExchNet: A Unified Hashing Network for Large-Scale Fine-Grained Image Retrieval," in *European Conference on Computer Vision*, 2020: Springer, pp. 189-205.

[15] T. Hu, H. Qi, Q. Huang, and Y. Lu, "See better before looking closer: Weakly supervised data augmentation network for fine-grained visual classification," *arXiv preprint arXiv:1901.09891*, 2019.

[16] W. Liu, A. Rabinovich, and A. C. Berg, "Parsenet: Looking wider to see better," *arXiv preprint arXiv:1506.04579*, 2015.

[17] M. Zhou, Y. Bai, W. Zhang, T. Zhao, and T. Mei, "Look-into-object: Self-supervised structure modeling for object recognition," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2020, pp. 11774-11783.

[18] D. Zoran, M. Chrzanowski, P.-S. Huang, S. Gowal, A. Mott, and P. Kohli, "Towards robust image classification using sequential attention models," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2020, pp. 9483-9492.

[19] X. Zheng, R. Ji, X. Sun, B. Zhang, Y. Wu, and F. Huang, "Towards optimal fine grained retrieval via decorrelated centralized loss with normalize-scale layer," in *Proceedings of the AAAI Conference on Artificial Intelligence*, 2019, vol. 33, no. 01, pp. 9291-9298.

[20] X. Zheng, R. Ji, X. Sun, Y. Wu, F. Huang, and Y. Yang, "Centralized Ranking Loss with Weakly Supervised Localization for Fine-Grained Object Retrieval," in *IJCAI*, 2018, pp. 1226-1233.

[21] H. Sun, Y. Fan, J. Shen, N. Liu, D. Liang, and H. Zhou, "A Novel Semantics-Preserving Hashing for Fine-Grained Image Retrieval," *IEEE Access*, vol. 8, pp. 26199-26209, 2020, doi: 10.1109/access.2020.2970223.

[22] M. Gori, G. Monfardini, and F. Scarselli, "A new model for learning in graph domains," in *Proceedings. 2005 IEEE International Joint Conference on Neural Networks, 2005.*, 2005, vol. 2: IEEE, pp. 729-734.

[23] A. Jain, A. R. Zamir, S. Savarese, and A. Saxena, "Structural-rnn: Deep learning on spatio-temporal graphs," in *Proceedings of the ieee conference on computer vision and pattern recognition*, 2016, pp. 5308-5317.

[24] C.-W. Lee, W. Fang, C.-K. Yeh, and Y.-C. F. Wang, "Multi-label zero-shot learning with structured knowledge graphs," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2018, pp. 1576-1585.

[25] X. Liang, X. Shen, J. Feng, L. Lin, and S. Yan, "Semantic object parsing with graph lstm," in *European Conference on Computer Vision*, 2016: Springer, pp. 125-143.

[26] M. Henaff, J. Bruna, and Y. LeCun, "Deep convolutional networks on graph-structured data," *arXiv preprint arXiv:1506.05163,* 2015.

[27] Y. Wang, Y. Sun, Z. Liu, S. E. Sarma, M. M. Bronstein, and J. M. Solomon, "Dynamic graph cnn for learning on point clouds," *Acm Transactions On Graphics (tog)*, vol. 38, no. 5, pp. 1-12, 2019.

[28] M. Datar, N. Immorlica, P. Indyk, and V. S. Mirrokni, "Locality-sensitive hashing scheme based on p-stable distributions," in *Proceedings of the twentieth annual symposium on Computational geometry*, 2004, pp. 253-262.

[29] Y. Gong, S. Lazebnik, A. Gordo, and F. Perronnin, "Iterative Quantization: A Procrustean Approach to Learning Binary Codes for Large-Scale Image Retrieval," *IEEE Transactions on Pattern Analysis & Machine Intelligence*, vol. 35, no. 12, pp. 2916-2929, 2013.

[30] H. Liu, R. Ji, Y. Wu, and W. Liu, "Towards optimal binary code learning via ordinal embedding," in *Proceedings of the AAAI Conference on Artificial Intelligence*, 2016, vol. 30, no. 1.

[31] R. Xia, Y. Pan, H. Lai, C. Liu, and S. Yan, "Supervised hashing for image retrieval via image representation learning," in *Proceedings of the AAAI conference on artificial intelligence*, 2014, vol. 28, no. 1.

[32] H. Lai, Y. Pan, Y. Liu, and S. Yan, "Simultaneous feature learning and hash coding with deep

neural networks," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2015, pp. 3270-3278.

[33] Z. Cao, M. Long, J. Wang, and P. S. Yu, "Hashnet: Deep learning to hash by continuation," in *Proceedings of the IEEE international conference on computer vision*, 2017, pp. 5608-5617.

[34] H. Zhu, M. Long, J. Wang, and Y. Cao, "Deep hashing network for efficient similarity retrieval," in *Proceedings of the AAAI Conference on Artificial Intelligence*, 2016, vol. 30, no. 1.

[35] C. Yue, M. Long, B. Liu, and J. Wang, "Deep Cauchy Hashing for Hamming Space Retrieval," in *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018.

[36] Y. Cao, M. Long, J. Wang, H. Zhu, and Q. Wen, "Deep quantization network for efficient image retrieval," in *Proceedings of the AAAI Conference on Artificial Intelligence*, 2016, vol. 30, no. 1.

[37] W.-J. Li, S. Wang, and W.-C. Kang, "Feature learning based deep supervised hashing with pairwise labels," *arXiv preprint arXiv:1511.03855*, 2015.

[38] S. Su, C. Zhang, K. Han, and Y. Tian, "Greedy hash: Towards fast optimization for accurate hash coding in cnn," in *Proceedings of the 32nd International Conference on Neural Information Processing Systems*, 2018, pp. 806-815.

[39] Q. Y. Jiang and W. J. Li, "Asymmetric Deep Supervised Hashing," 2017.

[40] J. Bruna, W. Zaremba, A. Szlam, and Y. LeCun, "Spectral networks and locally connected networks on graphs," *arXiv preprint arXiv:1312.6203*, 2013.

[41] M. Defferrard, X. Bresson, and P. Vandergheynst, "Convolutional neural networks on graphs with fast localized spectral filtering," *Advances in neural information processing systems*, vol. 29, pp. 3844-3852, 2016.

[42] T. N. Kipf and M. Welling, "Semi-supervised classification with graph convolutional networks," *arXiv preprint arXiv:1609.02907*, 2016.

[43] Y. Li, D. Tarlow, M. Brockschmidt, and R. Zemel, "Gated graph sequence neural networks," *arXiv preprint arXiv:1511.05493*, 2015.

[44] T.-Y. Lin and S. Maji, "Improved bilinear pooling with cnns," *arXiv preprint arXiv:1707.06772*, 2017.

[45] G. Zhong, H. Xu, P. Yang, S. Wang, and J. Dong, "Deep hashing learning networks," in *2016 International Joint Conference on Neural Networks (IJCNN)*, 2016: IEEE, pp. 2236-2243.

[46] Y. Wen, K. Zhang, Z. Li, and Q. Yu, "A Discriminative Feature Learning Approach for Deep Face Recognition," in *European Conference on Computer Vision*, 2016.

[47] WelinderP, BransonS, WahC, SchroffF, BelongieS, and PeronaP, "Caltech-UCSD Birds 200," *California Institute of Technology*, 2010.

[48] A. Khosla, N. Jayadevaprakash, B. Yao, and F. Li, "Novel dataset for fine-grained image categorization," 2013.

[49] J. Krause, M. Stark, J. Deng, and F. F. Li, "3D Object Representations for Fine-Grained Categorization," in *IEEE International Conference on Computer Vision Workshops*, 2014.

[50] Y. Yang, L. Geng, H. Lai, Y. Pan, and J. Yin, "Feature pyramid hashing," in *Proceedings of the 2019 on International Conference on Multimedia Retrieval*, 2019, pp. 114-122.