# Meta-Hashing for Remote Sensing Image Retrieval

Xu Tang, *Senior Member, IEEE*, Yuqun Yang, *Student Member, IEEE*, Jingjing Ma, *Member, IEEE*, Yiu-Ming Cheung, *Fellow, IEEE*, Chao Liu, Fang Liu, *Member, IEEE*, Xiangrong Zhang, *Senior Member, IEEE*, and Licheng Jiao, *Fellow, IEEE*

***Abstract*— With the explosive growth of the volume and resolution of high-resolution remote-sensing (HRRS) images, the management of them becomes a challenging task. The traditional content-based remote-sensing image retrieval (CBRSIR) technologies cannot meet what we expect due to the large volume of image archives and complex contents within HRRS images. As a successful approximate nearest neighborhood (ANN) search technique, Hash learning has received wide attention, especially when deep convolutional neural networks (DCNNs) appear. Due to DCNNs' strong capacity of feature learning, many DCNN-based hashing methods have been proposed and achieved good performance for large-scale CBRSIR tasks. Nevertheless, their limitation is that a large of labeled training samples should be collected for training the deep models. To overcome this limitation, this article, therefore, develops a new supervised hash learning method for the large-scale HRRS CBRSIR task based on meta-learning, which could achieve well-retrieval performance with a few labeled training samples. First, taking the characteristics of HRRS into account, we develop a self-adaptive convolution (SAP-Conv) block and design a hashing net based on the block. SAP-Conv can learn robust features from HRRS images by exploring their multiscale information. Second, to enhance the generalization of the hashing net under a few labeled training samples, the hash learning is formulated in a meta-way, and we name it meta-hashing. Meta-hashing can effectively preserve the similarities between support and query set, and the similarities between samples within support set by the developed loss function. To further improve the performance of meta-hashing, we expand it to a dynamic version named dynamic-meta-hashing, in which the numbers of support and query are changeable in the training phase. Experimental results counted on the three widely used HRRS datasets demonstrate our dynamic-meta-hashing and meta-hashing can achieve promising performance in large-scale HRRS CBRSIR tasks based on a few training samples. Our source codes are available at https://github.com/TangXu-Group/Meta-hashing.**

***Index Terms*— Content-based remote-sensing images retrieval (CBRSIR), meta-learning, supervised hash learning.**

Manuscript received July 1, 2021; revised October 13, 2021 and November 28, 2021; accepted December 14, 2021. Date of publication December 16, 2021; date of current version March 1, 2022. This work was supported in part by the National Natural Science Foundation of China under Grant 62171332, Grant 61801351, Grant 61802190, and Grant 61772400; in part by the Key Research and Development Program of Shaanxi under Grant 2021GY-035; in part by the Fund of National Key Laboratory of Science and Technology on Remote Sensing Information and Imagery Analysis, Beijing Research Institute of Uranium Geology, under Grant 6142A010301; in part by the China Postdoctoral Science Foundation Funded Project under Grant 2017M620441; in part by the Hong Kong Scholars Program under Grant XJ2019037; in part by the Fundamental Research Funds for the Central Universities under Grant 30919011281 and Grant JSGP202101; in part by the General Research Fund of Research Grants Council of Hong Kong under Project 12201321; in part by the NSFC under Grant 61672444; in part by the Hong Kong Baptist University under Grant RC-FNRA-IG/18-19/SCI/03 and Grant RC-IRCMs/18-19/SCI/01; in part by the Innovation and Technology Fund (ITF) of Innovation and Technology Commission (ITC) of the Government of the Hong Kong SAR under Project ITS/339/18; and in part by the Xidian University Artificial Intelligence School Innovation Fund Project under Grant YJS2115. *(Corresponding authors: Jingjing Ma; Yiu-Ming Cheung.)*

Xu Tang is with the Key Laboratory of Intelligent Perception and Image Understanding, Ministry of Education, School of Artificial Intelligence, Xidian University, Xi'an 710071, China, and also with the Department of Computer Science, Hong Kong Baptist University, Hong Kong, SAR, China (e-mail: tangxu128@gmail.com).

Yuqun Yang, Jingjing Ma, Chao Liu, Xiangrong Zhang, and Licheng Jiao are with the Key Laboratory of Intelligent Perception and Image Understanding, Ministry of Education, School of Artificial Intelligence, Xidian University, Xi'an 710071, China (e-mail: jjma@ieee.org).

Yiu-Ming Cheung is with the Department of Computer Science, Hong Kong Baptist University, Hong Kong, SAR, China (e-mail: ymc@comp.hkbu.edu.hk).

Fang Liu is with the Key Laboratory of Intelligent Perception and Systems for High-Dimensional Information, Ministry of Education, School of Computer Science and Engineering, Nanjing University of Science and Technology, Nanjing 210023, China.

This article has supplementary downloadable material available at https://doi.org/10.1109/TGRS.2021.3136159, provided by the authors.

Digital Object Identifier 10.1109/TGRS.2021.3136159

## I. Introduction

With the development of sensor technologies, the volume of remote-sensing (RS) images grows explosively and the spatial resolution of RS images increases drastically. These high-resolution RS (HRRS) images bring abundant earth observation information which could help scholars to study our planet deeply [1]–[6]. However, it is a tough task to manage the large-scale HRRS image archives well. To deal with this issue, content-based RS image retrieval (CBRSIR) attracts researchers' attention.

As a useful management tool, CBRSIR aims at organizing RS images according to their contents [7], [8]. In general, a complete CBRSIR system contains a feature extraction/learning module, a similarity matching module, and a re-ranking module [9]. Three modules focus on mapping RS images into a proper feature space, measuring the similarities between RS images, and adjusting the retrieval results according to the users' opinions, respectively. The retrieval precision and speed are often adopted to evaluate the performance of a CBRSIR system. The traditional CBRSIR methods extract hand-crafted visual features from RS images and adopt the exhaustive search scheme to accomplish the retrieval process [10], [11]. Nevertheless, they cannot achieve satisfactory results in the large-scale HRRS CBRSIR scenario since: 1) the hand-crafted features are not able to fully explore the complex contents within HRRS images and 2) the exhausting search is a time-consuming matching scheme. Recently,

the deep convolutional neural networks (DCNNs) [12]–[14] provide new opportunities for CBRSIR. The high-level features learned by DCNNs help the CBRSIR methods to achieve remarkable retrieval precision. However, these deep features are dense and high-dimensional. When the scale of HRRS archives is large, the time costs of the exhaustive search are unacceptable. To overcome this limitation, approximate nearest neighbor (ANN) search [15] entries the CBRSIR community.

Unlike the exhaustive search, ANN search only focuses on the neighbors around the query image to enhance retrieval efficiency. Among many solutions of the ANN search, hashing is a useful one, which aims at mapping the images into the binary discrete feature space so that the ANN search can be completed efficiently and accurately [16]. In the last decades, various hashing methods have been developed, and they perform positively in different fields [17]. Especially when DCNN appears, the performance of the deep hashing is impressive [18]. In the RS community, deep hashing also draws researchers' eyes. Many deep hashing methods have been proposed to deal with large-scale CBRISR tasks [19]. They accomplish retrieval from different aspects, such as robust hashing features learning [20]–[23], and cross-model RS images understanding [24], [25]. Also, these RS-oriented deep hashing methods can be broadly classified into two groups, i.e., supervised and unsupervised. Although they have demonstrated some level of success in these application domains, they have the following limitations. For unsupervised techniques, the behavior of learned hash codes is always far from satisfactory since they lack sufficient semantic information during the learning process. For supervised methods, a bulk of labeled data should be collected to train the deep hashing network, which is impractical in most RS applications. When the volume of data with prior knowledge is limited, the deep hashing models cannot be trained well for CBRSIR tasks. Moreover, their generalization is not strong enough.

Considering the limitation mentioned above, is it possible to develop a deep hashing method that can be trained by a few samples and has a strong generalization ability? Fortunately, the current hot topic, few-shot learning (FSL), can help us to construct a deep hashing model at a small sample cost. As a sub-area of machine learning, FSL aims to train a model using a limited number of supervised samples [26]. To achieve this goal, many methods and learning algorithms have been proposed for FSL, such as meta-learning [27], [28], multi-task learning [29], [30], and embedding learning [31], [32]. Meta-learning methods accomplish FSL in an episode way, which could enhance the model's generalization. By discovering the task-generic and task-specific knowledge, multi-task learning approaches learn different tasks at the same time, which can be adopted to complete FSL naturally. The key point of embedding learning is mapping samples into a low-dimensional space, in which the inter-class and intra-class samples could be compacted and dispersed, respectively. Due to this property, the embedding learning methods require fewer samples, which match the FSL smoothly.

Based on the above discussion, we propose a new deep hashing network with the consideration of characteristics of HRRS images in this article. Besides, a meta-learning algorithm, named *meta-hashing*, is developed to train our model under the FSL paradigm, which could reduce the burden of collecting labeled data and enhance the model's generalization simultaneously. Specifically, the proposed hashing network consists of a DCNN and a hashing layer. First, to fully explore the multi-scale information and complex contents from HRRS images, we develop a new self-adaptive pyramid convolution (SAP-Conv) block and embed it into ResNet18 [33]. To map the learned deep features into the hash codes, we embed a hashing layer at the top of DCNN. Second, rather than collecting a large number of labeled data to train our model, the meta-hashing algorithm uses multiple episodes with limited labeled HRRS images to obtain a transferable hashing model. In addition, a specific hashing loss function is defined to guarantee the usefulness of the learned hash codes, which strictly follows the rules of meta-learning and hash learning. Third, a dynamic $N$-way $K$-shot strategy is designed to further improve the performance of meta-hashing. Besides the image retrieval, the proposed methods can be used for RS image content understanding applications, such as scene classification, land covers searching, etc. Furthermore, their helpfulness would be highlighted when the volume of labeled data is small. From this point, our model is more practical than the traditional hashing techniques introduced in the RS community.

The main contributions of our method are as follows.

1) To complete the hash learning task for HRRS images, we proposed an end-to-end deep hashing network. It can fully explore the multi-scale information by the developed SAP-Conv block which is embedded in the hashing network.
2) To train the proposed hashing network under the framework of FSL, we formulate the hash learning in a meta way (i.e., meta-hashing). Through expanding data-based learning to task-based learning, not only the number of labeled samples can be reduced but also the generalization ability of the trained hashing model can be enhanced. As far as we know, it is the first time to use meta-learning in the CBRSIR community.
3) To satisfy the rules of hash learning under the meta-learning paradigm, we first define the distance metric for measuring the data cells. Then, the specific loss function is constructed. It can: 1) preserve the similarity relationships between HRRS images from the original space to the hashing space and 2) compact and separate the intra-class/inter-class HRRS images.
4) To further improve the performance of meta-hashing, we expand the basic $N$-way $K$-shot strategy to the dynamic version. By adjusting values of $N$ and $K$ dynamically during the meta-learning process, the transferability of the trained model can be enhanced.

The rest of this manuscript is organized as follows. The published literature related to hashing in RS and FSL is reviewed in Section II. The proposed deep hashing network and meta-hashing algorithm are introduced in Section III. Section IV displays the experiments and relevant discussion. The conclusion is exhibited in Section V.

## II. Related Work

### A. Hashing in Remote Sensing

In this section, to review the existing RS-oriented hashing methods logically, we divide them into two groups according to if there is a deep neural network in hash learning or not, and we name them nondeep and deep hashing.

The nondeep hashing methods focus on adopting or developing specific hashing algorithms/functions to map the RS images' visual features into the binary hash codes. As early as 2015, Demir and Bruzzone [34] applied two classical hashing methods, the kernel-based unsupervised/supervised hashing [35], [36], to encode RS images. Compared with the typical visual features, although the retrieval precision based on the obtained hash codes is slightly weaker, the retrieval efficiency is enhanced distinctly. This confirms hash learning is feasible in the RS community. To improve the performance of hashing for RS images, a partial randomness hashing method was proposed in the literature [37]. In this method, a part of parameters of hashing functions was generated randomly and the rest of the parameters were trained by RS images. Through this co-play scheme, the obtained hash functions are effective. The methods mentioned above only apply some existing hashing approaches to encode the HRRS images directly. Although they can achieve an acceptable precision at a low cost of time consumption in CBRSIR tasks, the characteristics of RS images (such as objects in HRRS images are diverse in type and huge in volume) are not taken into account. To overcome this shortcoming, a method based on multiple hash codes fusion scheme was proposed [38]. It first generates different hash codes with the consideration of various properties of RS images. Then, a multi-hash-code-matching strategy is used to get the retrieval results. Another nondeep hashing method was introduced in [39], in which the probabilistic latent model is utilized to explore the semantics from RS images. By adding the high-level information, the behavior of the final hash codes is enhanced.

Although the nondeep hashing methods are feasible in practice and reasonable in theory, their performance is limited by the pre-obtained visual features. In contrast, deep hashing approaches integrate feature learning and hash mapping by the neural network. Due to the hierarchical structure and nonlinear fitting capacity, deep hashing achieves high retrieval precision and dominates the large-scale CBRSIR community. The first deep hashing method for RS images may be the deep hashing neural network (DHNN) which was proposed in [20]. DHNN consists of a deep feature learning network and a hash learning network, which focus on learning the high-level semantics from RS images and the hash codes, respectively. Through developing the hashing function with the binary quantization and pairwise similarity constraints, the discrete binary hash codes can be extracted from RS images in an end-to-end manner. Based on DHNN, another deep hashing network was presented in the literature [25] to deal with the cross-source large-scale CBRSIR task. The mentioned two hashing networks are developed based on feedforward DCNN. Although their performance is promising, the distribution gaps between continuous deep features and discrete hash codes are not considered. To overcome this limitation, Tang *et al.* [21] proposed a semi-supervised deep adversarial hashing (SDAH) model. In SDAH, the residual autoencoder is developed to complete the feature learning. To bridge the gap between two kinds of feature distributions, an adversarial network is added. Its "true" input is a manual data which has discrete binary values (0 and 1) and follows the uniform distribution. Through the adversarial regularization, the continuous deep features can be converted into the hash codes smoothly. Another deep adversarial hashing network was introduced in [22], which is named feature and hash (FAH) learning. To explore information from RS images from different aspects, FAH develops the specific feature learning network, which contains the multi-scale and attention branches. Both the global, local, and multi-scale knowledge can be represented by the learned deep features. Meanwhile, the hashing network based on the generative adversarial model is constructed to generate the hash codes naturally.

### B. FSL in Neural Networks

FSL is a challenging research topic in machine learning, especially in the current deep era. The limited labeled samples in FSL cannot provide enough information to train a deep neural network commonly. Therefore, some new models (e.g., meta-learning) have been proposed to train the deep neural network under the FSL scenario. In this section, we divide the FSL methods in the neural network into two parts for review. One is meta-learning-based methods and the other one is non-meta-learning-based methods.

The meta-learning-based FSL (ML-FSL) methods train the deep neural network in the episode manner. Through task-based learning, the generalized network can be trained with a few numbers of labeled data. ML-FSL methods are popular in many research areas. Snell *et al.* [40] proposed the prototypical networks to deal with the few-shot classification task, where a metric space is learned by a meta-learning algorithm for performing the classification. A meta-learning-based few-shot segmentation network was developed in the literature [41]. The authors use small subsets of training images to mimic the few-shot settings. In each subset, one image is regarded as the query data and the others are regarded as the support data. Through training the DCNN using the support data, the query image's segmentation result can be obtained by the cosine distances between its feature map and class vectors corresponding to different support sets. An ML-FSL-based object detection method was introduced in [42]. In this method, the CentreNet [43] is decomposed into class-generic and class-specific parts. The class-generic part is trained regularly while the class-specific part is trained in a meta-way. Then, the object detection is accomplished by an incremental scheme.

The non-meta-learning-based FSL (NML-FSL) methods are diverse. Many models can be used to complete FSL. A few-shot domain adaption method was introduced in [44] under the framework of multi-task learning, where two variational autoencoders are developed to mine the generic information from the source and target tasks. A dynamic conditional

network (DCN) was proposed for FSL according to the theory of embedding learning [45]. DCN contains two subsets. One consists of a set of filters and the other consists of the adaptive weights that are used to combine the filters linearly. Combining them together, the task embedding can be learned for conditional FSL. Under the generative modeling framework, a generative adversarial network (GAN)-based FSL method was presented in the literature [46]. Due to the property of GAN, a sharp decision boundary can be learned for the few-shot classification task in a semi-supervised way.

Although the number of FSL studies is few, the significance of FSL is shown gradually in the RS community. An FSL method was proposed to deal with hyperspectral image (HSI) classification in [47]. Apart from constructing a specific DCNN for HSIs, a meta-learning algorithm is also developed to train the DCNN. The effectiveness and the transferability of the trained model are demonstrated by the cross-source experiments. Here, the cross-source experiments denote that the training and testing HSIs are collected by different sensors. Another HSI classification FSL method was introduced in [48]. To handle the unseen category issue, a two-phase relation network is developed. The network is trained in the meta-way, which guarantees the generation of the model and reduces the number of labeled data. In addition, transfer learning and fine-tuning technologies are often considered in FSL. Rostami *et al.* [49] proposed a deep transfer learning method for few-shot synthetic aperture radar (SAR) image classification. Two encoders are trained in optical RS and SAR domains, respectively. Then, by transfer learning, the SAR image classification can be completed with a few labeled samples. A transfer learning and deep convolutional neural network (TL-DeCNN) was proposed in [50] to classify HRRS scenes based on few shot samples. Through transferring the weights of three typical DCNNs, the RS scene classification network can be fine-tuned by a few labeled samples. In [51], a few-shot classification method was proposed for HSIs. Combining the multitask transfer learning technique and the fine-tune technology, the proposed Dirichlet-net can improve classification performance.

## III. Proposed Method

### A. Preliminaries

Before explaining the details of the proposed hashing net and meta-hashing algorithm, some preliminaries are introduced in this section, including supervised hash learning and meta-learning.

*1) Supervised Hash Learning:* Supervised hash learning focus on learning a mapping function $\mathcal{H}_\theta(\cdot)$ with sign activation function which is used to map images $\mathbf{x}$ into hash codes $\mathbf{b} = \text{sign}(\mathcal{H}_\theta(\mathbf{x}))$, where $\theta$ is the parameters of hash model. For hash learning, a key point is similarity preserving which means the relationship between images should be maintained into their hash codes. To this end, many methods optimize a global objective based on triplet or contrastive loss to find the optimal parameters $\theta$. Here, we adopt the contrastive function [52] to accomplish the main idea of hash learning

$$\begin{aligned}\mathcal{L}_{\text{contrastive}} = \sum_{i,j}\big\{ & I_{ij}\text{distance}\big(\mathbf{b}_i, \mathbf{b}_j\big) \\ & +\big(1 - I_{ij}\big)\max\big(m - \text{distance}\big(\mathbf{b}_i, \mathbf{b}_j\big), 0\big)\big\} \\ & \text{s.t. } \mathbf{b}_i, \mathbf{b}_j \in \{-1, 1\}^L \end{aligned} \tag{1}$$

where $I_{ij}$ is the supervised indicator, $I_{ij} = 1$ ($I_{ij} = 0$) means the $i$th and the $j$th images are similar (dissimilar), $\mathbf{b}_i$ and $\mathbf{b}_j$ are the hash codes of $i$th and the $j$th images, distance$(\mathbf{b}_i, \mathbf{b}_j)$ means the distance (i.e., Hamming distance) between $\mathbf{b}_i$ and $\mathbf{b}_j$ which is used to evaluate the similarity in the hashing space, $m > 0$ is a margin parameter, and $L$ is the length of hash codes. Since the hash codes are discrete, directly optimizing the hash learning model is difficult. Therefore, many hash learning methods will relax the discrete hash codes into continuous value by $\tanh(\cdot)$ activation function for facilitating the optimization. Here, we use $\tilde{\mathbf{b}}_i$ to indicate the real-valued hash codes and $\tilde{\mathbf{b}}_i = \tanh(\mathcal{H}_\theta(\mathbf{x}_i))$. In this case, the Euclidean distance is always selected to replace Hamming distance during the model training.

The supervised hash learning methods could obtain effective hash model by minimizing the above objective. However, since the intra-class diversity and inter-class similarity caused by the complex types and varying resolutions of the objects within HRRS images, many of the supervised hash learning methods need to be optimized by enough labeled samples to obtain well generalization for the large scale remained samples.

*2) Meta-Learning:* As a famous supervised method based on meta-learning, ML-FSL has achieved encouraging performance in many applications, such as [40] and [41]. Many of them decompose a single training objective with overall observation of training samples into multiple sub-tasks. For simple, we use $\mathcal{T}_p \sim \mathcal{P}(\mathcal{T})$ to denote the sub-task $\mathcal{T}_p$ which is sampled from task distribution $\mathcal{P}(\mathcal{T})$. Suppose there are $M$-classes in training set. In each one sub-task $\mathcal{T}_p$, there are $N$-classes ($N < M$) mini training set which contains support set $\{\mathbf{x}^i_{S_r}\}^{i=1,2,\ldots,K}_{r=1,2,\ldots,N}$ and query set $\{\mathbf{x}^j_{Q_r}\}^{j=1,2,\ldots,K_q}_{r=1,2,\ldots,N}$, where $K$ and $K_q$ are the number of support and query samples per class. One talks about the $N$-way $K$-shot learning. Then, they employ a network with parameters $\phi$ to produce a distribution over classes for a query point $\mathbf{x}^j_{Q_r}$ based on a softmax over distances to the class center in the embedding space

$$p_\phi\Big(y = r|\mathbf{x}^j_{Q_r}\Big) = \frac{\exp\Big\{-\text{distance}\Big(f_\theta\Big(\mathbf{x}^j_{Q_r}\Big), \mathbf{c}_r\Big)\Big\}}{\sum_{r'}^{N}\exp\Big\{-\text{distance}\Big(f_\theta\Big(\mathbf{x}^j_{Q_r}\Big), \mathbf{c}_{r'}\Big)\Big\}}. \tag{2}$$

The $r$-class center can be calculated as the mean vector of the $i$-class support points, $\mathbf{c}_r = (1/K)\sum_i^K f_\theta(\mathbf{x}^i_{S_r})$. Finally, the learning proceeds can be completed by minimizing the negative log-probability $\mathcal{L}_\phi = -(1/K \times N)\sum_j^K\sum_r^N \log\{p_\phi(y = r|\mathbf{x}^j_{Q_r})\}$.

Through learning in each one sub-task $\mathcal{T}_p \sim \mathcal{P}(\mathcal{T})$, many of the FSL methods could achieve well generalization for large-scale remained samples based on few-labeled training samples. To explain the $N$-way $K$-shot learning clearly, we exhibit a

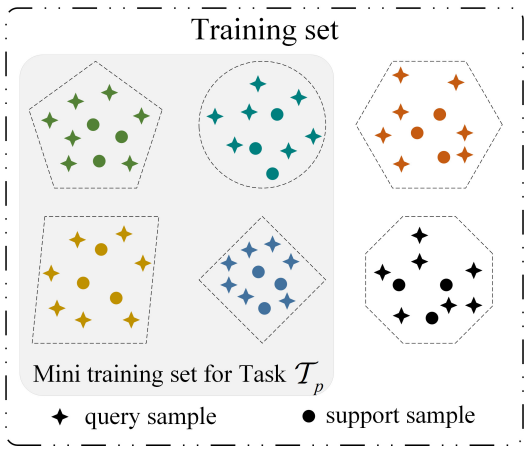


Fig. 1. Schematic of four-way three-shot learning. There are six classes in training set. Four classes data (with the gray background) are selected to be mini training set for finishing sub-task $\mathcal{T}_p$. Each class data contain the support set with three samples and the query set with seven samples.

four-way three-shot example in Fig. 1. There are six classes and each class has ten samples. We select four of them randomly to construct the mini training set for sub-task $\mathcal{T}_p$. Three samples within each class are chosen as the support samples, and the rest of samples are selected to be query samples. Through varying the combinations within the mini training set, a set of sub-tasks $\mathcal{P}(\mathcal{T})$ can be established to train the model at a small cost of labeled samples.

### *B. Problem We Want to Solve*

As we mentioned before, the volume and resolution of HRRS images are increasing with the development of the earth observation technique, which introduces some new challenges for HRRS image retrieval. First, the volume of HRRS images is large, while the number of labeled samples is relatively small. Therefore, how to learn an effective deep hashing model with a few labeled samples is a critical problem that we need to tackle. Second, the complex types and resolutions of HRRS images will enrich the complexity of contents and increase the intra-class diversity and inter-class similarity. Therefore, our new deep hashing network should consider the diverse contents within HRRS images and the intra- and inter-class relationships between HRRS images. To achieve the goals mentioned above, we develop the hashing net and meta-hashing method. First, we design the hashing net for learning dense features with the consideration of complex contents within HRRS images. Second, the meta-hashing algorithm is developed to learn a practical and transferable hash model using a few labeled training samples. Also, different distance metrics are designed to separate/impact the inter-/intra-class images.

### *C. Hashing Net*

Taking the characteristics of RS images into account, we develop hashing net in this section. First, we design a new SAP-Conv block which is good at capturing the multi-scale features. Second, we embed the SAP-Conv in the widely used ResNet18 [33] and add the hashing layer to construct the hashing net. In detail, we replace the last $3 \times 3$ convolutional operation in ResNet18. To map the obtained multi-scale features into hash codes, we add a hashing layer with $\tanh(\cdot)$ activation function on the top of hashing net. Note that, we also add a classifier layer on the hashing layer since our previous work [22] has proved the semantic information could increase the discrimination of learned hash codes. The detailed architecture of hashing net is described in the Section IV. In the following paragraphs of this section, we focus on introducing the SAP-Conv block.

The main idea of the SAP-Conv block is to explore the multi-scale information without increasing the trained parameters. This will profit the network to learn robust features when the number of training samples is few. The architecture of SAP-Conv is shown in Fig. 2, which mainly contains two components, i.e., pyramid convolution and scale self-adaptive scheme. The pyramid convolution aims at learning multi-scale features by the same filters. Then, the contributions of different features corresponding to different scales are integrated using the scale self-adaptive scheme for any specific tasks.

*Pyramid convolution* is designed based on dilated convolution [53]. Simply, we use the following formulation to denote the dilated convolution:

$$\mathbf{f}_d = \mathbf{f}_{\text{in}} \otimes_d \mathbf{w} \tag{3}$$

where $d$ indicates the dilation rate, $\mathbf{f}_d$ is the outputs, $\mathbf{f}_{\text{in}}$ denotes the input features maps, and $\mathbf{w}$ means the parameters of convolutional kernel. When $d = 1$, the dilated convolution equals standard convolution. When $d > 1$, the dilated convolution has an enlarged receptive field compared with the standard convolution. Taking $3 \times 3$ filters and $d = 1, 2, 3$ as examples, the filters' receptive fields are $3 \times 3$, $5 \times 5$ and $7 \times 7$, respectively, and the number of three filters' parameters are equal to that of $3 \times 3$. Based on these properties, which the dilated convolution enlarges the receptive fields without increasing parameters, we develop the pyramid convolution here. The proposed model utilizes the same filters to convolute the input feature maps in parallel paths with different dilation rates. It can be formulated as

$$\mathbf{f}_d^{(i)} = \mathbf{f}_{\text{in}} \otimes_{d=i} \mathbf{w}, \quad i = [1, 2, 3, \ldots, D] \tag{4}$$

where $D$ is the number of scale level.

It is noted that there are two points we want to touch on. First, in this article, the size of feature maps $\mathbf{f}_d^{(i)}$is equal to $C' \times H' \times W'$ through the padding operation before dilated convolutions. The reason is that the multi-scale information of each pixel in $\mathbf{x}$ can be reflected by the same position in $\mathbf{f}_d^{(i)}$. Hence, the pyramid convolution would not introduce any additional parameters. Second, the concept of the receptive field can be understood from macro- and micro-perspectives. From a macro-standpoint [54], the receptive field of a CNN is the region of the input space. From a micro-view [53], the CNN is constructed with the network unit, and each unit is a convolution kernel. Thus, the receptive field of the convolution kernel is kernel size. In this article, similar to [53], the receptive field here is discussed from the micro-perspective. In other

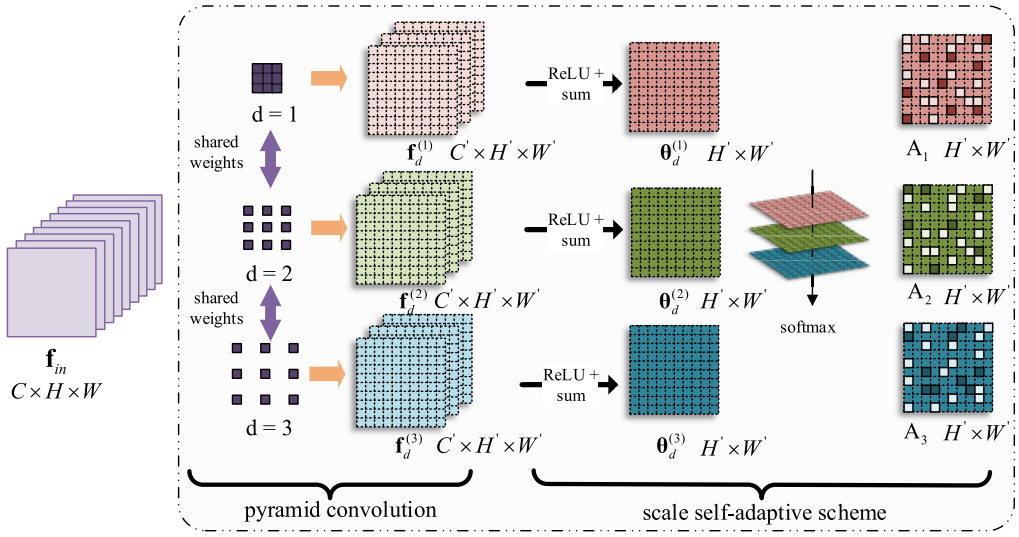


Fig. 2. Framework of proposed SAP-Conv block.

words, the receptive field is decided by the convolutional kernels.

*Scale self-adaptive scheme* is developed to effectively integrate the multi-scale features $\mathbf{f}_d^{(i)}$. Its main idea is that each position $p(j,k)$ of the fused features has different coefficients for combining $\mathbf{f}_d^{(i)}$ together. Then, we have the following function:

$$\mathbf{f}_{self}(j,k) = \sum_{i=1}^{D} \alpha_i(j,k)\mathbf{f}_d^{(i)}(j,k)$$
$$\text{s.t.} \sum_{i=1}^{D} \alpha_i(j,k) = 1, 0 \le \alpha_i(j,k) \le 1 \tag{5}$$

where $\mathbf{f}_{\text{self}}(j,k)$ denotes the fused features in $p(j,k)$, and $(j,k) \in (H', W')$. In addition, the coefficients $\alpha_i(j,k)$ denotes the contribution of $\mathbf{f}_d^{(i)}(j,k)$ to the $\mathbf{f}_{\text{self}}(j,k)$.

For calculating the coefficients $\alpha_i(j,k)$, there are three steps. First, the ReLU activation function is applied on these obtained multi-scale features to add the nonlinearity

$$\mathbf{g}_d^{(i)} = \text{ReLU}\left(\mathbf{f}_d^{(i)}\right) \tag{6}$$

where $\mathbf{g}_d^{(i)} \in \mathbb{R}^{C' \times H' \times W'}$. Second, for each $\mathbf{g}_d^{(i)}$, we combine each position's information by summing them together along with the channel. This operation can be denoted as

$$\theta_d^{(i)} = \sum_{c=1}^{C'} \mathbf{g}_d^{(i)} \tag{7}$$

where $\theta_d^{(i)} \in \mathbb{R}^{H' \times W'}$. Since the sum operation for each scale features $\mathbf{f}_d^{(i)}$ will lead to different value range, therefore, $\theta_d^{(i)}$ is normalized into the same scale $(0,1)$ for effectively comparing. At last, for each position $p(j,k)$ of the fused features, a softmax function is employed for highlighting the important scale features and weakening the others. This can be formulated as

$$\alpha_i(j,k) = \frac{e^{\theta_d^{(i)}(j,k)}}{\sum_{n=1}^{D} e^{\theta_d^{(n)}(j,k)}}. \tag{8}$$

where $(j,k) \in (H', W')$. Then, for calculating simply, $\mathbf{A}_i \in \mathbb{R}^{H' \times W'}$ are obtained by stacking $\alpha_i(j,k)$ together. we name $\mathbf{A}_i$ scale attention map since it indicates the different contributions of $\mathbf{f}_d^{(i)}$. When each scale attention map $\mathbf{A}_i$ is obtained, the multi-scale features $\mathbf{f}_d^{(i)}$ can be fused as follows:

$$\mathbf{f} = \sum_{i=1}^{D} \mathbf{A}_i \odot \mathbf{f}_d^{(i)} \tag{9}$$

where $\mathbf{f} \in \mathbb{R}^{C' \times H' \times W'}$, $\odot$ means the elementwise product. Then, in accordance with the traditional CNNs, $\mathbf{f}$ can be passed with the general operation, i.e., activation functions and pooling.

Note that, there is another point we want to further explore. As we present above, $\theta_d^{(i)}$ is normalized into $(0,1)$ for effectively comparing them. However, the softmax function in this value range could not highlight the important scale features enough. For example, suppose $D = 3$, $\theta_d^{(1)}(j,k) = 1$, $\theta_d^{(2)}(j,k) = 0.01$ and $\theta_d^{(3)}(j,k) = 0.01$, the coefficients can be calculated as, $\alpha_1(j,k) \approx 0.5737$, $\alpha_2(j,k) \approx 0.2132$, $\alpha_3(j,k) \approx 0.2132$. For improving the ability of highlighting important multi-scale features, the following strategy is introduced:

$$\sigma_d^{(i)} = \frac{1}{-\log_2 \theta_d^{(i)}} \tag{10}$$

$$\alpha_i(j,k) = \frac{e^{\sigma_d^{(i)}(j,k)}}{\sum_{n=1}^{D} e^{\sigma_d^{(n)}(j,k)}}. \tag{11}$$

By this strategy, suppose $D = 3$, $\theta_d^{(1)}(j,k) = 1$, $\theta_d^{(2)}(j,k) = 0.01$ and $\theta_d^{(3)}(j,k) = 0.01$, the coefficients can be calculated as, $\alpha_1(j,k) \approx 1$ , $\alpha_2(j,k) \approx 0$ , $\alpha_3(j,k) \approx 0$ . As the results show, the most important features $\mathbf{f}_d^{(1)}(j,k)$ has the most contribution.

The proposed scale self-adaptive scheme has a similar function to the popular spatial attention scheme. However, the normal spatial attention mechanisms aim at emphasizing the important spatial information to explore the multi-scale

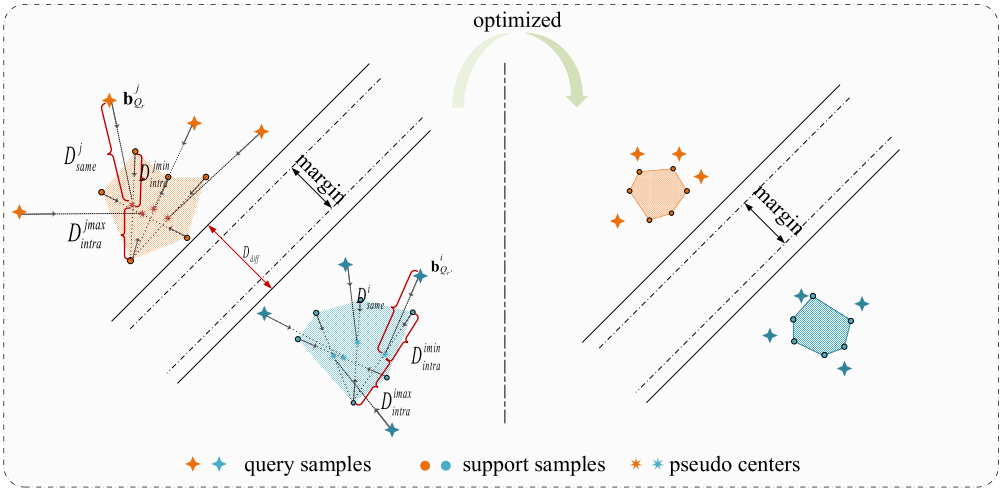


Fig. 3. Schematic of the distances in meta-hashing. The points with same color means they belong to same class. Optimizing these distances could make the similar data points to be closer and dissimilar data points to be farther that margin.

knowledge. At the same time, our scale self-adaptive scheme pays more attention to fuse features with different scales.

### D. Meta-Hashing

To complete the learning hash model based on few-labeled samples, we develop the meta-hashing algorithm. For the sake of clarity, we will introduce meta-hashing in two aspects, including learning in one sub-task $\mathcal{T}_p$ and learning in task distribution $\mathcal{P}(\mathcal{T})$.

*1) Learning in One Sub-Task $\mathcal{T}_p$:* In accordance with the $N$-way $K$-shot learning, there are the support set $\{\mathbf{x}_{S_r}^i\}_{r=1,2,\ldots,N}^{i=1,2,\ldots,K}$ and query set $\{\mathbf{x}_{Q_r}^j\}_{r=1,2,\ldots,N}^{j=1,2,\ldots,K_q}$ in the task $\mathcal{T}_p$, where $\{\mathbf{x}_{S_r}\} \cap \{\mathbf{x}_{Q_r}\} = \emptyset$. Their real-valued hash codes can be denoted by $\{\tilde{\mathbf{b}}_{S_r}^i\}_{r=1,2,\ldots,N}^{i=1,2,\ldots,K}$ and $\{\tilde{\mathbf{b}}_{Q_r}^j\}_{r=1,2,\ldots,N}^{j=1,2,\ldots,K_q}$. To maintain images' relationships into hashing space, we expect the distances of same classes samples are smaller than that of other dissimilar classes samples by a margin in the hashing space. To tackle the impact caused by the intra-class diversity and inter-class similarity, we minimize three kinds of distances, including the intra-distances $D_{\text{intra}}$ between the same classes support samples and pseudo centers, the intra-distances $D_{\text{same}}$ between the same classes query samples and pseudo centers, and the inter-distances $D_{\text{diff}}$ between different classes support samples and query samples. For convenient interpretation, we have shown them in Fig. 3.

*a) $D_{\text{intra}}$ and $D_{\text{same}}$:* Before computing the distance $D_{\text{intra}}$ and $D_{\text{same}}$, we need to define the pseudo centers. For each one query sample's real-valued hash codes $\tilde{\mathbf{b}}_{Q_r}^j$, we calculate a pseudo center $\tilde{\mathbf{c}}_r^j$ by the following three steps. First, we compute the distances between $\tilde{\mathbf{b}}_{Q_r}^j$ and $\{\tilde{\mathbf{b}}_{S_r}^i\}^{i=1,2,\ldots,K}$, which indicated by $\{d_r^{ji}\}^{i=1,2,\ldots,K}$. Second, finding the nearest and farthest samples ($\tilde{\mathbf{b}}_{S_r}^{j\min}$ and $\tilde{\mathbf{b}}_{S_r}^{j\max}$) through indexing the samples with minimum and maximum distances in $\{d_r^{ji}\}^{i=1,2,\ldots,K}$. Finally, the pseudo center $\tilde{\mathbf{c}}_r^i$ is calculated as

$$\tilde{\mathbf{c}}_r^j = \frac{\tilde{\mathbf{b}}_{S_r}^{j\min} + \tilde{\mathbf{b}}_{S_r}^{jmax}}{2}. \tag{12}$$

Once we obtain the pseudo center $\tilde{\mathbf{c}}_r^j$, three $r$th class data points $\tilde{\mathbf{b}}_{Q_r}^j$, $\tilde{\mathbf{b}}_{S_r}^{j\max}$, $\tilde{\mathbf{b}}_{S_r}^{j\min}$ could be used to compute the distance $D_{\text{intra}}^j$ and $D_{\text{same}}^j$ as follows:

$$D_{\text{intra}}^j = \underbrace{\|\tilde{\mathbf{b}}_{S_r}^{j\min} - \tilde{\mathbf{c}}_r^j\|_2^2}_{D_{intra}^{j\min}} + \underbrace{\|\tilde{\mathbf{b}}_{S_r}^{j\max} - \tilde{\mathbf{c}}_r^j\|_2^2}_{D_{\text{intra}}^{jmax}} \tag{13}$$

$$D_{\text{same}}^j = \|\tilde{\mathbf{b}}_{Q_r}^j - \tilde{\mathbf{c}}_r^j\|_2^2. \tag{14}$$

Note that, there are $K_q$ query samples belonging to $r$th class in task $\mathcal{T}_p$. Such that, for $r$-class, we can obtain the $D_{\text{intra}}(r)$ and $D_{\text{same}}(r)$ as follows:

$$D_{\text{intra}}(r) = \frac{1}{K_q}\sum_{j}^{K_q} D_{\text{intra}}^j \tag{15}$$

$$D_{\text{same}}(r) = \frac{1}{K_q}\sum_{j}^{K_q} D_{\text{same}}^j. \tag{16}$$

In the end, for task $\mathcal{T}_p$, we minimize the following objective function to regularize all the similar samples to be mapped closer in hashing space:

$$\mathcal{L}_{\text{same}} = \frac{1}{N}\sum_{r}^{N}\{D_{intra}(r) + D_{\text{same}}(r)\}. \tag{17}$$

*b) $D_{\text{diff}}$:* To maintain the relationship between dissimilar samples, we employ a penalty term on the dissimilar samples when the distance between their hash codes is smaller than a margin $m$. For clarity, we also introduce how to compute the $D_{\text{diff}}^j$ for the particular query sample $\tilde{\mathbf{b}}_{Q_r}^j$ first. Then the objective function can be calculated by the mean value of all the query samples' $D_{\text{diff}}^j$ in task $\mathcal{T}_p$.

For the $r$-class query sample $\tilde{\mathbf{b}}_{Q_r}^j$, we compute all the distances between $\tilde{\mathbf{b}}_{Q_r}^j$ and dissimilar support samples $\tilde{\mathbf{b}}_{S_{r'}}^j$. $\{d_{rr'}^{ji}\}_{r'\neq r, r'=1,2,\ldots,N}^{i=1,2,\ldots,K}$ is adopted to denote the computed distances. In general, we can obtain the objective function for $\mathbf{b}_{Q_r}^j$ as follows:

$$\mathcal{L} = \frac{1}{(N-1)\times K}\sum_{r'}^{N}\sum_{i}^{K}\max\left(m - d_{rr'}^{ji}, 0\right). \tag{18}$$

Minimizing this objective function is useful for regularizing the dissimilar data points into dissimilar hash codes. However,

since each $d_{rr'}^{ji}$ has the same contribution in (9), the hash learning model cannot concentrate on the "hard" samples which mean the samples are from different classes but have high inter-class similarities. Intuitively, if the model can map "hard" samples into correct hash codes, the other samples can also be mapped correctly.

To make the model focus on the "hard" samples, we formulate the objective function based on distances between the "hard" samples. For the query $\mathbf{b}_{Q_r}^{j}$ and $r'$-class support set, we find the minimum distance, as follows:

$$d_{rr'}^{j\min} = \min\left\{d_{rr'}^{j1}, d_{rr'}^{j2}, \ldots, d_{rr'}^{jK}\right\}. \tag{19}$$

Then, for the $r$-class query and $r'$-class support sets, we can obtain all the "hard" samples' distances as follows:

$$D_{\text{diff}}\left(rr'\right) = \frac{1}{K_q}\sum_{j}^{K_q}\max\left(m - d_{rr'}^{j\min}, 0\right). \tag{20}$$

In the end, for task $\mathcal{T}_p$, we minimize the following objective to regularize the dissimilar samples to be mapped into hashing space with the distances that are bigger than $m$:

$$\mathcal{L}_{\text{diff}} = \frac{1}{N\times(N-1)}\sum_{r}^{N}\sum_{r'}^{N}D_{\text{diff}}\left(rr'\right). \tag{21}$$

*2) Learning in Task $\mathcal{T}_p \sim \mathcal{P}(\mathcal{T})$:* Suppose the number of tasks is $T$. For each task, we randomly select $N$ classes from dataset, where $N < M$. For each class, we randomly sample $K$ and $K_q$ images as the support set and query set from the training set. In addition, as our developed hashing net that adds the classifier layer on the hashing layer, we utilize the cross-entropy loss function to boost the discrimination of learned hash codes. Hence, we optimize the hash learning model in each task through minimizing the following objective:

$$\mathcal{L}_{\text{similarity}} = \mathcal{L}_{\text{same}} + \mathcal{L}_{\text{diff}} + \alpha\mathcal{L}_{\text{cross-entropy}} \tag{22}$$

where $\alpha$ is the hyper-parameter which is used to control the contribution of the last item.

As we mentioned before, we use $\tanh(\cdot)$ activation function on the lash layer for relaxing the discrete hash codes into real-valued hash codes. Therefore, to make the real-valued hash codes close to the discrete codes, we simply use large margin $m$. Once the hash model is completely optimized, we replace the $\tanh(\cdot)$ by $\text{sign}(\cdot)$ activation function for generating discrete hash codes. For clarity, we provide Algorithm 1 to detail the procedure of meta-hashing.

### E. Dynamic-Meta-Hashing

Generally, the $N$ and $K$ are fixed in ML-FSL methods. In this article, we expand our meta-hashing algorithm into a dynamic version, named *dynamic-meta-hashing*, for boosting the robustness of hashing model. It means the $N$ and $K$ are randomly changing when the training process is conducted in multiple sub-tasks. Note that, the changeable $N$ and $K$ should be smaller than the number of all classes $M$ and the numbers of samples per classes in training set. The detail the procedure of dynamic-meta-hashing is displayed in Algorithm 1. The main idea of the proposed dynamic meta-hashing is changing the number of classes $N$ and support samples $K$ could introduce different inter-class similarity and intra-class diversity. Intuitively, with increasing the number of classes $N$, the higher inter-class similarity will be introduced in the current tasks. Meanwhile, when increasing the number $K$, the intra-class diversity will impact the hash model a lot. Therefore, dynamically adjusting $N$ and $K$ could boost the robustness and performance of meta-hashing. The positive experimental results also confirm this idea. The implementation details are displayed in Section IV-B.

**Algorithm 1** Meta-Hashing and Dynamic-Meta-Hashing

**Input:** Training image set, the number of classes sampled per task $N$, the number of support and query samples per class $K$ and $K_q$ (their summation equals the number of training samples per class), maximum number of task $T$, margin parameter $m$.

**Output:** The optimized meta-hashing model with parameter $\theta$;

Initialize $\theta$;
**for** $t = 1, 2, \ldots, T$ **do**
  **if** dynamic-meta-hashing **then**
    //dynamic-meta-hashing
    Randomly fix $N$, $K$ and $K_q$;
  **end if**
    Randomly select $N$ classes from dataset;
    Randomly sample $K$ and $K_q$ images per class as the support set and query set;
    Compute the distances between support and query samples using (15), (16) and (20);
    Optimize $\theta$ by minimizing (22);
**end for**
Return $\theta$

## IV. Experiments

### A. Datasets

To validate the performance of our methods, including the meta-hashing and dynamic meta-hashing, we conduct extensive experiments on three published HRRS datasets.

*1) UCMerced:* This dataset was published by the University of California at Merced (UCMerced) [55]. There are 21 scene categories and each category contains 100 HRRS images with size of $256 \times 256$. The spatial resolution is 0.3 m/pixel in the RGB color space. Some examples and the semantic categories of the UCMerced dataset are displayed in Fig. 4.

*2) AID:* The aerial image dataset (AID) was introduced by the Wuhan University [56] which is made up of 30 aerial scene classes and has 10 000 images. The number of samples in each category varies from 200 to 420, the spatial resolution of RS images changes from 0.5 to 8 m, and the sizes of images are $600 \times 600$. Some examples and their semantics of this dataset are shown in Fig. 5.

*3) NWPU:* This dataset was published by Northwestern Polytechnical University (NWPU) [57] that contains 31 500 RS images which have been divided into 45 scene categories. Each category includes 700 images with size of $256 \times 256$ in RGB color space. The spatial resolution varies from about 30 to


Fig. 4. Snapshot of different scenes in the UCMerced dataset.


Fig. 5. Snapshot of different scenes in the AID dataset.

0.2 m/pixel. The highly overlapping contents make the dataset more challenging in the retrieval task, and some examples and their classes are displayed in Fig. 6.

### *B. Experimental Settings*

*1) Implementation Details:* We accomplish the meta-hashing and dynamic-meta-hashing methods on the developed hashing network whose architecture is displayed in Table I. As shown in this table, we replace the last $3 \times 3$ convolution of ResNet18 by our SAP-Conv block and set the scale level $D = 3$. Besides, we initial the ResNet18 (except the classifier layer) by the pre-trained parameters which are obtained using the ImageNet dataset [58], and other parts of our hashing net (i.e., hashing layer, SAP-Conv block, and the classifier layer) are initialized randomly. For validating the generation of our methods, we construct three different training sets for each dataset. Specifically, for the UCMerced, AID, and NWPU datasets, we randomly select five, eight, and ten images per class to construct three kinds of training sets. Corresponding to each training sets, the reminded samples are adopted as the query sets for completing the retrieval tasks based on all samples in these datasets. Here, the PyTorch platform [59] and the Adam optimization algorithm [60] are adopted to accomplish the training process. In the following experiments, unless otherwise stated, the maximum number of task $T$, the weight decay parameter, and the initial learning rate are set to be 10 000, 0.0005, and 0.0001, respectively. Also, the values of learning rate are divided by ten when the number of tasks is in multiples of 5000. By default, we set the margin parameter $m = 24, 32, 40, 48$ when the hash codes lengths are 24, 32, 40, 48, and the parameters $\alpha = 1.0$ for the rest of experiments. The influence of these free parameters is discussed in Section IV-E.

Note that, for meta-hashing, we fix $N = 5$ for all experiments. For dynamic-meta-hashing, the number of $N$ are

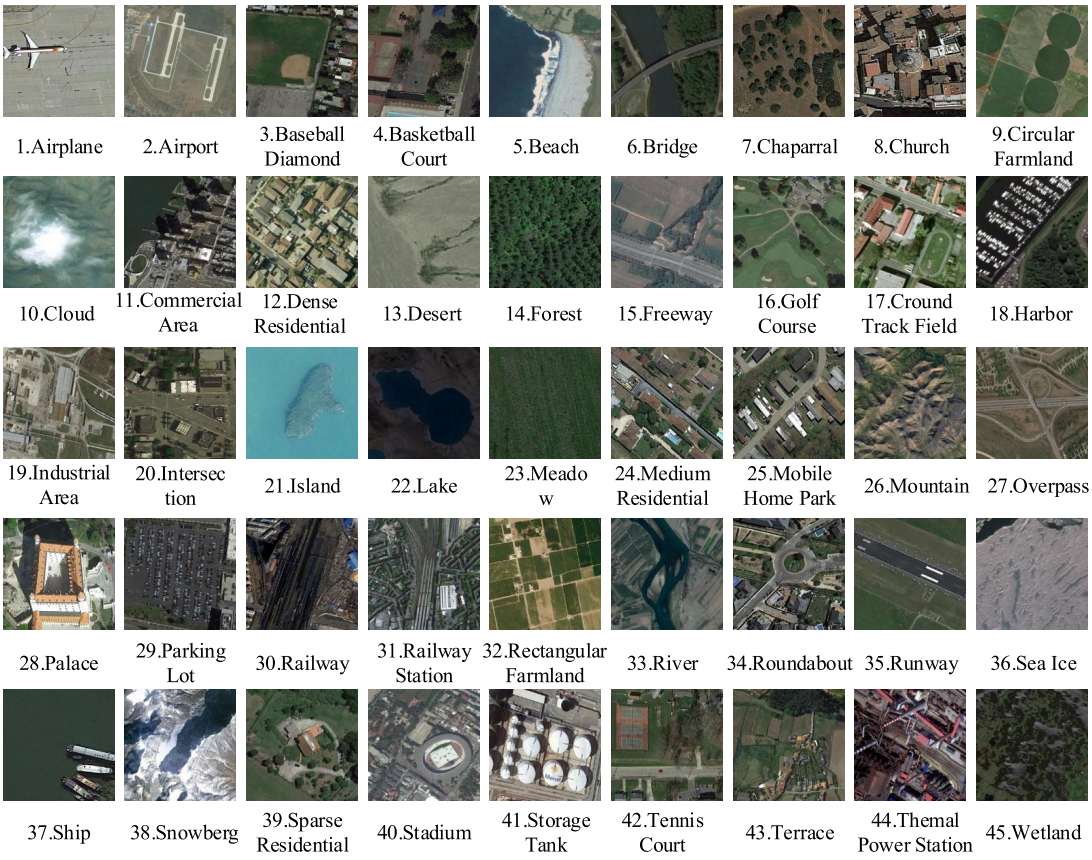


Fig. 6. Snapshot of different scenes in the NWPU dataset.

TABLE I

ARCHITECTURES FOR HASHING NET. BUILDING BLOCKS ARE SHOWN IN BRACKETS, WITH THE NUMBERS OF BLOCKS STACKED

| layer name | output size | Hashing Net |
|---|---|---|
| conv1 | 112×112 | 7×7, 64, stride 2 |
| conv2_x | 56×56 | 3×3 max pool, stride 2 |
| | | $\begin{bmatrix} 3\times3,\ 64 \\ 3\times3,\ 64 \end{bmatrix}\times2$ |
| conv3_x | 28×28 | $\begin{bmatrix} 3\times3,\ 128 \\ 3\times3,\ 128 \end{bmatrix}\times2$ |
| conv4_x | 14×14 | $\begin{bmatrix} 3\times3,\ 256 \\ 3\times3,\ 256 \end{bmatrix}\times2$ |
| conv5_x | 7×7 | $\begin{bmatrix} 3\times3,\ 512 \\ \text{SAP-Conv}(3\times3),\ \text{D=3},\ 512 \end{bmatrix}\times2$ |
| | 1×1 | average pool |
| hashing layer | $L$ | $L$-d fc layer with $tanh(\cdot)$ |
| classifier layer | $M$ | $M$-d softmax |

randomly selected in [5, 6, 7, 8, 9, 10] with each steps for all experiments. Considering the different number of training samples, we summarize the configurations about the number of support set $K$ and query set $K_q$ in Table II.

*2) Compared Methods:* To the best of our knowledge, the number of hashing methods for CBRSIR under the paradigm the FSL or meta-learning is few. Such that, to verify if our methods are effective or not, we select several different kinds of hash learning methods as the compared methods, including supervised hashing, semi-supervised hashing, and fine-tune-based hashing methods. Specifically, supervised methods contain deep supervised hashing (DSH) [61] and feature and hash learning (FAH) [22]. For the semi-supervised methods, they are semi-supervised deep hashing (SSDH) [62] and semi-supervised deep adversarial hashing (SDAH) [21]. The fine-tune-based method is metric and hash-code learning network (MHCLN) [63].

We further adopt two metric learning-based methods for comparison with our proposed solutions. One is the prototypical network [40]. Here, the last fully connected layers' outputs of the prototypical network are used to complete the retrieval task. The other one is the deep meta-metric learning (DMML) [64], in which the specific distance metric between samples is learned in the meta manner and the obtained metric is chosen to accomplish our retrieval task.

For the sake of fairness, all of the compared methods' dense feature learning parts are replaced by our hashing net (i.e., ResNet18 with SAP-Conv block). For DSH, FAH, SDAH, SSDH, and MHCLN, the mini-batch size is set at 64, total iterations are equal to 10 000, the initial learning rate is 0.0001. We also divide the learning rate by ten every 5000 iterations. For prototypical network and DMML, we set $N$ and $K$ to

TABLE II
CONFIGURATIONS ABOUT THE NUMBER OF SUPPORT SET $K$ AND QUERY SET $K_q$

| Data set | The number of training samples per class | $K$ and $K_q$ for meta-hashing | $K$ and $K_q$ for dynamic-meta-hashing |
|---|---|---|---|
| UCMerced/AID/NWPU | 5 | $K = 3$, $K_q = 2$ | $K = [2, 3]$, $K_q = 5 - K$ |
| | 8 | $K = 4$, $K_q = 4$ | $K = [3, 4, 5]$, $K_q = 8 - K$ |
| | 10 | $K = 5$, $K_q = 5$ | $K = [3, 4, 5, 6, 7]$, $K_q = 10 - K$ |

be the same with our meta-hashing (see Table II). Since the supervised methods (such as DSH and FAH) cannot be trained using the limited labeled samples, the data augmentations (including random cropping, horizontal flipping, and vertical flipping) are adopted to enrich the datasets ensure different methods' behavior. The diversity rather than the volume of training samples is enhanced after the data augmentation.

Finally, all of the experiments are conducted on an HP-Z840-Workstation with Xeon(R) CPU E5-2650, TITAN Xp, 256G RAM, and Ubuntu OS 16.04.

*3) Evaluation Metrics:* Two widely used evaluation metrics are selected to assess the retrieval performance [22], including the mean average precision (MAP) and precision–recall curve (PR-curve). MAP reflects the overall retrieval accuracy with the consideration of position information, while PR-curve represents the variation between the retrieval precision and recall [65]. Here, the MAP can be calculated as follows:

$$\text{MAP} = \frac{1}{|Q|}\sum_{i=1}^{|Q|}\frac{1}{N_i}\sum_{i=1}^{N_i}\text{precision}(R_{ik}) \tag{23}$$

where $|Q|$ is the size of the query set. Specifically, $q_i \in Q$ is a query image, $N_i$ is the number of relevant images to $q_i$ in the retrieval database. Suppose the relevant images are ordered as $\{r_i, r_2, \ldots, r_{N_i}\}$, $R_{ik}$ is the set of ranked results from the top result until we get to $r_k$ point. Note that, we count the MAP values on the top 20 retrieval results as an ideal retrieval system should rank more relevant results on the top positions. Here, relevant results mean the retrieved HRRS images have the same semantics as the query images. Specifically, for the hash learning methods (i.e., dynamic-meta-hashing, meta-hashing, FAH, DSH, SSDH, and SDAH), we rank the relevant results based on Hamming distance. For the metric learning methods (i.e., DMML and protopical network), the retrieval results are ranked based on Euclidean distance.

### C. Overall Performance

*1) Mean Average Precision:* In this section, we report the MAP scores of our model and other compared methods. As described above, we construct three kinds of training sets for each dataset. Here, we recall them UCMerced-5, UCMerced-8, UCMerced-10, AID-5, AID-8, AID-10, NWPU-5, NWPU-8, and NWPU-10 for short. The MAP scores of different methods are summarized in Table III. In this table, the best results are bold for clarity. From observing these results, we can find that our models can achieve the best performance.

1) *Compared With Conventional Hashing Methods:* It is easy for us to can find that the meta-learning-based methods (i.e., dynamic-meta-hashing, meta-hashing, DMML, and prototypical network) outperform traditional hashing methods across different hash codes with diverse bits and datasets in most cases, especially when the number of training samples is small. Taking the results based on UCMereced-5 (24-bits) as examples, compared with the best conventional method (SSDH), the improvements of meta-learning-based methods are 5.93% (dynamic-meta-hashing), 6.04% (meta-hashing), 5.29% (DMML), and 1.36% (prototypical network), respectively. The promising results show that the meta-learning can bring a good generalization based on a few labeled training samples. It is worth noting that the performance of conventional methods is enhanced with the increasing of the volume of training samples. Taking the UCMerced-10 (24 bits) as examples, the best method (SSDH) among the conventional methods outperforms DMML and prototypical network by 0.32% and 3.48%, respectively. The reason behind this is that the key idea of DMML and prototypical network is to minimize the distances between query samples and support set. However, they do not take the relationships between the samples within support sets into account. Back to our developed methods (dynamic-meta-hashing and meta-hashing), benefiting from the designed strategies for preserving the similarities between the samples within support sets and the resemblance between the query and support samples, their performance is more superior than that of the conventional hashing methods even though the number of training samples is increasing. For instance, for the UCMerced-10 (24 bits), the dynamic-meta-hashing and meta-hashing outperform SSDH by 2.01% and 1.07%, respectively.
2) *Compared With Meta-Learning-Based Methods:* The behavior of the proposed models (dynamic-meta-hashing and meta-hashing comparison) is stronger than that of other meta-learning-based methods in most cases (DMML and prototypical network). Taking AID-5 (32-bits) as an example, the improvements of dynamic-meta-hashing are 2.16% (DMML) and 4.17% (prototypical network), while the enhancements of meta-hashing are 1.04% (DMML) and 3.59% (prototypical network). An interesting observation is that DMML can achieve competitive results sometimes. The reason is that DMML is a metric learning method in which the retrieval results are obtained by the learned distances between samples rather than calculating the Hamming distances between hash codes. Nevertheless, its efficiency is relatively low compared with our models since the Hamming distance could be computed by the efficient bit-wise XOR operation. The details of retrieval speed will be discussed in Section IV-F.

TABLE III

MAP VALUES (%) OF DIFFERENT METHODS COUNTED BY THE TOP 20 RETRIEVAL RESULTS ACROSS THE DIFFERENT LENGTH OF HASH CODES AND DIFFERENT DATASETS

| Data sets | Codes length | Ours | | Meta-learning | | Supervised | | Semi-supervised | | Fine-tune |
|---|---|---|---|---|---|---|---|---|---|---|
| | | Dynamic-meta-hashing | Meta-hashing | DMML | Prototypical Network | FAH | DSH | SSDH | SDAH | MHCLN |
| UCMerced-5 | 24-bits | 85.01 | **85.12** | 84.29 | 80.44 | 76.66 | 70.85 | 79.08 | 74.35 | 65.55 |
| | 32-bits | **87.11** | 85.97 | 86.24 | 82.08 | 79.03 | 70.66 | 81.31 | 73.67 | 66.96 |
| | 40-bits | **87.67** | 86.34 | 84.84 | 80.69 | 80.39 | 73.71 | 81.96 | 76.12 | 66.18 |
| | 48-bits | **86.73** | 85.92 | 85.30 | 79.29 | 81.52 | 70.73 | 80.85 | 76.78 | 67.61 |
| UCMerced-8 | 24-bits | **90.07** | 89.64 | 89.00 | 85.66 | 83.61 | 81.94 | 87.40 | 83.80 | 72.33 |
| | 32-bits | **90.67** | 89.67 | 89.02 | 86.52 | 84.51 | 85.30 | 86.56 | 82.53 | 74.49 |
| | 40-bits | **92.21** | 91.09 | 89.75 | 86.26 | 85.61 | 86.23 | 88.63 | 84.54 | 74.56 |
| | 48-bits | 91.40 | **92.01** | 89.41 | 85.90 | 87.19 | 82.68 | 90.03 | 85.20 | 74.76 |
| UCMerced-10 | 24-bits | **93.74** | 92.80 | 91.41 | 88.25 | 87.25 | 89.44 | 91.73 | 83.83 | 72.61 |
| | 32-bits | **93.16** | 93.08 | 91.50 | 88.80 | 87.02 | 89.54 | 91.11 | 85.21 | 72.51 |
| | 40-bits | 93.41 | **93.46** | 91.88 | 88.73 | 87.20 | 90.62 | 90.11 | 87.55 | 73.35 |
| | 48-bits | **94.24** | 93.66 | 92.09 | 87.13 | 88.19 | 89.71 | 90.19 | 87.50 | 76.76 |
| AID-5 | 24-bits | **75.52** | 74.12 | 74.50 | 71.93 | 67.62 | 69.97 | 70.95 | 59.87 | 51.16 |
| | 32-bits | **76.18** | 75.06 | 74.02 | 71.47 | 69.96 | 71.43 | 72.00 | 64.93 | 51.18 |
| | 40-bits | **77.64** | 76.75 | 75.46 | 71.57 | 70.13 | 68.98 | 70.30 | 66.30 | 53.01 |
| | 48-bits | **78.69** | 77.38 | 76.76 | 73.72 | 72.06 | 66.64 | 71.43 | 69.66 | 53.20 |
| AID-8 | 24-bits | **81.80** | 80.58 | 80.15 | 77.41 | 76.44 | 75.22 | 78.28 | 71.60 | 51.53 |
| | 32-bits | **82.38** | 80.75 | 80.69 | 77.45 | 76.91 | 75.27 | 77.29 | 73.29 | 54.85 |
| | 40-bits | **82.43** | 82.25 | 81.49 | 78.41 | 78.55 | 75.37 | 76.48 | 74.67 | 56.82 |
| | 48-bits | **82.85** | 82.75 | 82.09 | 78.82 | 78.89 | 72.14 | 79.48 | 76.16 | 57.94 |
| AID-10 | 24-bits | 82.04 | **83.23** | 82.17 | 79.12 | 78.45 | 76.55 | 78.67 | 73.23 | 55.69 |
| | 32-bits | **83.77** | 81.98 | 82.35 | 79.00 | 79.36 | 76.58 | 78.85 | 77.23 | 57.16 |
| | 40-bits | **84.53** | 83.60 | 82.31 | 79.23 | 79.03 | 76.61 | 79.43 | 78.49 | 58.17 |
| | 48-bits | **84.00** | 83.82 | 83.01 | 79.52 | 79.44 | 75.88 | 77.35 | 79.19 | 60.21 |
| NWPU-5 | 24-bits | **67.43** | 66.56 | 66.24 | 65.03 | 59.74 | 58.02 | 60.77 | 57.05 | 41.53 |
| | 32-bits | **68.16** | 66.46 | 67.07 | 65.82 | 63.19 | 60.76 | 60.79 | 61.40 | 47.06 |
| | 40-bits | **70.69** | 70.39 | 67.95 | 66.30 | 62.80 | 58.26 | 62.35 | 61.61 | 48.88 |
| | 48-bits | **70.38** | 69.54 | 68.38 | 65.53 | 63.54 | 59.57 | 61.21 | 64.26 | 50.48 |
| NWPU-8 | 24-bits | **72.37** | 69.17 | 70.55 | 68.24 | 65.05 | 62.32 | 67.71 | 64.95 | 44.25 |
| | 32-bits | **73.08** | 69.75 | 71.01 | 68.68 | 68.24 | 63.58 | 67.72 | 67.84 | 48.52 |
| | 40-bits | **73.83** | 69.85 | 72.39 | 69.40 | 69.67 | 63.47 | 66.97 | 67.25 | 49.77 |
| | 48-bits | **74.20** | 70.44 | 71.46 | 70.09 | 70.79 | 65.22 | 65.09 | 70.66 | 52.01 |
| NWPU-10 | 24-bits | **74.78** | 73.18 | 72.67 | 70.28 | 69.49 | 71.96 | 71.66 | 68.31 | 46.65 |
| | 32-bits | **76.42** | 72.74 | 75.14 | 72.02 | 70.78 | 70.31 | 70.63 | 69.37 | 50.31 |
| | 40-bits | **77.41** | 73.33 | 75.28 | 71.74 | 73.85 | 71.50 | 70.82 | 70.86 | 51.74 |
| | 48-bits | **77.68** | 74.14 | 73.35 | 73.79 | 73.70 | 71.24 | 68.45 | 70.67 | 52.85 |

3) *Dynamic-Meta-Hashing and Meta-Hashing Comparison:* From observing the MAP values of dynamic-meta-hashing and meta-hashing across different hash bits and datasets, we find the dynamic-meta-hashing outperforms the meta-hashing in most cases, especially when the datasets are complex. Taking the results of AID-8 as examples, compared with the meta-hashing method, the improvements of dynamic-meta-hashing are 1.22% (24-bits), 1.63% (32-bits), 0.18% (40-bits), and 0.10% (48-bits), respectively. When the datasets are more complex, the performance gain of dynamic-meta-hashing tends to be large. For NWPU-8, the enhancements of dynamic-meta-hashing are 3.2% (24-bits), 3.33% (32-bits), 3.98% (40-bits), and 3.76% (48-bits) compared with the meta-hashing method. The contents discussed above illustrate the following points. First, adjusting the values of $N$ and $K$ dynamically during the training procedure can help the model to explore complex contents within the HRRS images. Second, by changing the values of $N$ and $K$ randomly, the generalization of the model can be improved further.

Apart from the contents discussed above, there are some notable points we want to further explain. Sometimes, the behavior of meta-hashing is weaker than that of dynamic-meta-hashing. For instance, the meta-hashing method can outperform the dynamic method by 0.11% (UCMerced-5, 24-bits), 0.61% (UCMerced-8, 48-bits), 0.05% (UCMerced-10, 40-bits), and 1.19% (AID-10, 24-bits). This is due to the parameter settings. In detail, during the training process, we set values of the number of training samples per class and the hyper-parameters $m$ and $\alpha$ equally for two models. However, their contributions would be varied for different datasets. Nevertheless, dynamic-meta-hashing still outperforms meta-hashing in most cases, which confirms our dynamic boosting scheme is effective. The encouraging results discussed above prove the superiority of our models.

*2) Visual Results:* To further study the performance of our models and compared methods, we draw the precision and recall curves of all methods based on different code lengths and datasets in Fig. 7. From observing these curves, we can easily find that the areas under the precision and recall curves of our dynamic-meta-hashing and meta-hashing are larger than

that of other methods. This demonstrates that our models outperform the compared method directly. There is another point we want to touch on, that is, the precision values of DMML and the prototypical network is dropped fast with their recall values increase. The reasons can be concluded as two aspects. The first one is that DMML and the prototypical network do not take the relationships between the samples within support sets into account which may lead the training samples cannot be embedded compactly. The other one is there are no additional terms in DMML and prototypical network's objective functions to penalize the distances between negative samples. This would decrease the discrimination of clusters belonging to different classes. In contrast, for some other hash learning methods including dynamic-meta-hashing, meta-hashing, DSH, and SSDH, they adopt a large margin (i.e., larger than 24 in dynamic-meta-hashing and meta-hashing, 24 in DSH and 3 in SSDH) to penalize the distances between negative samples. Consequently, their precision values are stable with the recall values increase.

To confirm the above observations, we also study the structure of different retrieval codes. Due to the space limitations, we depict the visual structure of the retrieval codes obtained by all compared methods in UCMerced-10 (24-bits), AID-10 (24-bits), and NWPU-10 (24-bits) datasets. For showing their structure clearly, we adopt the t-distributed stochastic neighbor embedding (t-sne) algorithm [66] to reduce the dimension of the retrieval codes into 2-D space and exhibit them in Fig. 8. From observing these pictures, we can easily find that the clusters of the hash learning methods (i.e., dynamic-meta-hashing, meta-hashing, DSH, and SSDH) which adopt the large margin to restrict the distances between negative samples are more compact. While, for the DMML and prototypical network, their clusters are a little loose as there are no penalty terms to control the distances between negative samples. As shown in all the pictures, we can find the clusters of our methods are more distinct and clean than others. These positive results demonstrate the discrimination of retrieval codes obtained by our methods is high which is beneficial to the HRRS image retrieval task.

Besides the precision and recall curves and feature structure, we also provide some visual retrieval examples of different methods using 24-bits codes. Three HRRS images are selected from UCMerced-10, AID-10, and NWPU-10 randomly, and their top ten retrieval results are exhibited. From the observation of visual retrieval examples, the superiorities of our models are confirmed again. The details can be found in the Supplementary Material.

### D. Effectiveness of SAP-Conv Block

In Section III-C, we develop the SAP-Conv block and embed it into ResNet18 to construct our hashing net. SAP-Conv aims at capturing the multi-scale features from HRRS images, so that the rich objects can be explored. To study SAP-Conv's contributions to our model, we conduct the following ablation experiments. First, we construct two networks for generating the hash codes, they are as follows.

*Net1:* Resnet18+Hashing-Layer+Classifier-Layer.

*Net2:* Resnet18+SAP-Conv+Hashing-Layer+Classifier-Layer.

Then, the proposed meta-hashing and dynamic meta-hashing algorithms are used to train them, respectively. Net1 is the proposed hashing net without the SAP-Conv block, and Net2 is our hashing net. Their retrieval results of different datasets based on diverse hash codes are exhibited in Fig. 9. From observing them, we can easily find that Net2 outperforms Net1 in all cases, which confirms the usefulness of the proposed SAP-Conv block.

### E. Parameters Analysis

In this section, we study the influence of two free parameters (i.e., $\alpha$ and the margin $m$) used in our methods. For clarity, we select three datasets (i.e., UCMerced-8, AID-8, and NWPU-8) and set the code length to be 32 bits to analyze their influence. First, we fix $m = 32.0$ and vary the value of $\alpha$ in $[0.0, 1.0]$ for studying the influence of $\alpha$. Then we keep $\alpha$ to be 1 and change the value of $m$ in $[2.0, 32.0]$ to analyze its contributions. The results are summarized in Figs. 10 and 11, respectively.

*1) Parameter $\alpha$:* From observing Fig. 10, we can find the following points.

1) For the UCMerced-8 dataset, the behavior of meta-hashing is getting strong when the values of $alpha$ increase. For instance, when $\alpha = 0.0$ the MAP value of meta-hashing is 87.62%. While when $\alpha = 0.20$ the MAP value of meta-hashing is 90.32%. There is a distinct performance gap between them, which means the larger $\alpha$ the better performance for meta-hashing. When $\alpha \in [0.20, 1.0]$, meta-hashing performs steadily. Meanwhile, due to the dynamic scheme, the performance of dynamic-meta-hashing fluctuates slightly. Its MAP values always stay around 91% when $\alpha \in [0.0, 1.0]$. Also, the tendency of dynamic-meta-hashing behavior is upward with the increasing $\alpha$.
2) For the AID-8 dataset, similar to the results of UCMerced-8, when the values of $alpha$ become larger the performance of dynamic-meta-hashing and meta-hashing gets better. For meta-hashing, its MAP values grow from 77.34% to 81.23% when $alpha$ increases from 0.0 to 0.4. When $alpha \in [0.4, 1.0]$, its MAP values equal 81% approximately and the peak value (81.97%) can be reached at $alpha = 0.8$. For dynamic-meta-hashing, the MAP values increase from 74.02% to 80.94% when $alpha$ increases from 0.0 to 0.2. When $\alpha \in [0.2, 1.0]$, dynamic-meta-hashing performs steadily and performance peak value (82.38%) appears at $alpha = 1.0$.
3) For the NWPU-8 dataset, the MAP values of dynamic-meta-hashing and meta-hashing are both raising with increasing $alpha$. In detail, the MAP values of dynamic-meta-hashing and meta-hashing are 64.46% and 50.99% when $\alpha = 0.0$. Their MAP values can be reached 73.08% and 69.75% at $\alpha = 1.0$, respectively.

*2) Parameter $m$:* From observing Fig. 11, the following points can be summarized.

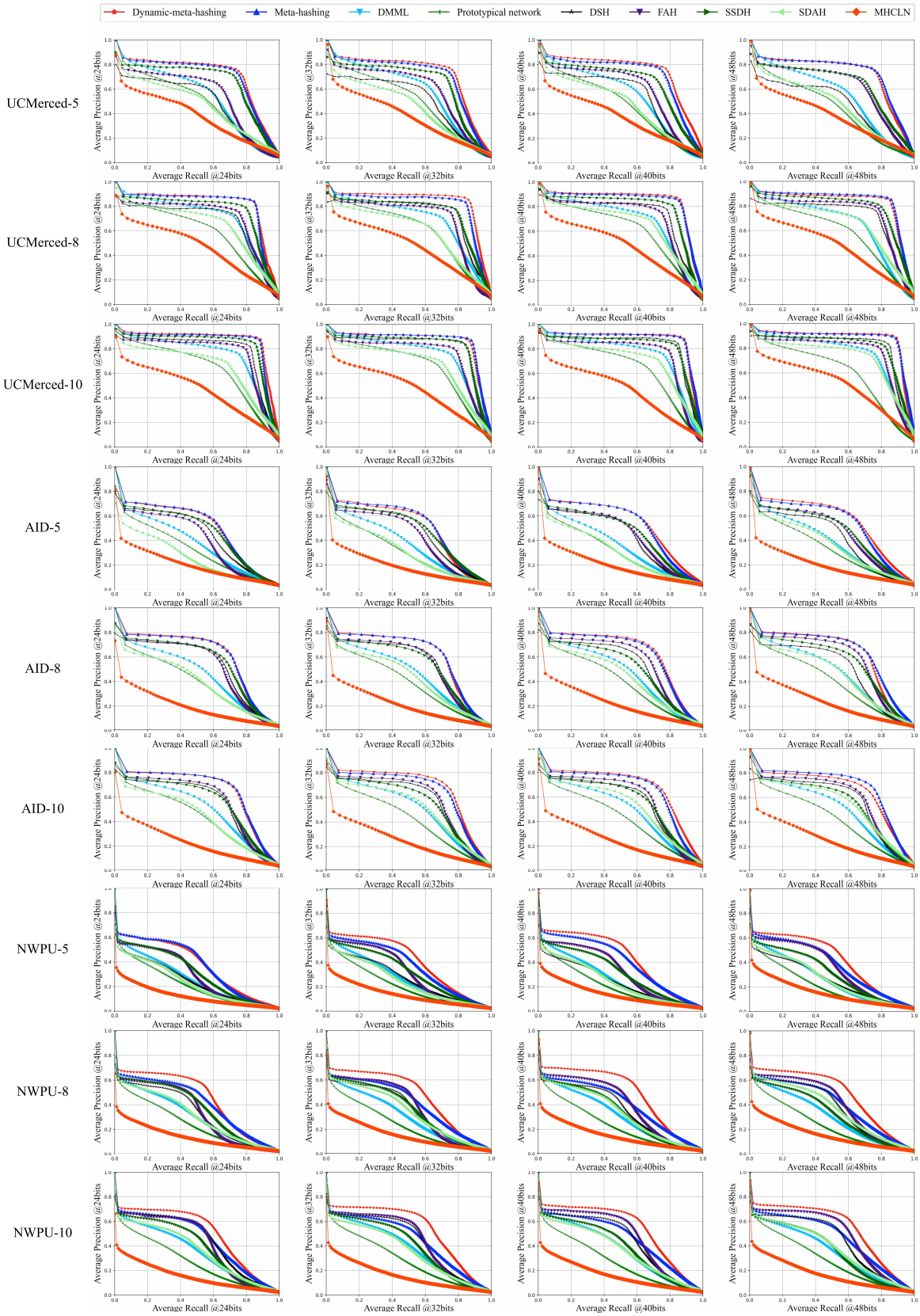


Fig. 7. Precision and recall curves of all the methods across diverse bits and datasets.

1) For the UCMerced-8 dataset, the MAP values of dynamic-meta-hashing are ranged in $[88.81, 91.82]$ when $m \in [2.0, 32.0]$ and the peak value (91.82%) is obtained at $m = 16.0$. For meta-hashing, $m$ impacts its performance distinctly. For instance, its MAP value is 80.24% when $m = 2.0$ and its peak value (90.52%)


Fig. 8. Two-dimensional scatterplots of different retrieval codes obtained by t-sne over all datasets. The relationships between number ID and categories is same with the Figs. 3–5.

appears at $m = 16.0$. Also, when $m \in [16.0, 32.0]$, the MAP values of meta-hashing are stable which stay around 90%.

2) For the AID-8 dataset, we also find the dynamic-meta-hashing could obtain a relatively stable MAP values (about 82%) when $m \in [2.0, 32.0]$ and the peak value (82.38%) is reached at $m = 32.0$. For the meta-hashing model, increasing the value of $m$ could bring the performance enhancement. In detail, the MAP value of meta-hashing is 77.90% when $m = 2.0$ and 81.40% (peak value) when $m = 24.0$. When $m \in [20.0, 32.0]$, the MAP values of meta-hashing stay around 81%.
3) For the NWPU-8 dataset, similar to the observations of the other two datasets, the MAP values of dynamic-meta-hashing are stable (around 73%) when $m \in [2.0, 32.0]$ and the peak value (74.76%) is achieved at $m = 20.0$. While the behavior of meta-hashing is getting stronger with the increasing the values of $m$. In detail, the its MAP value is 51.14% when $m = 2.0$, and it will be as high as 73.39% when $m = 24.0$. When $m \in [24, 32]$, the MAP values of meta-hashing are stable (around 70%).

Based on the contents discussed above, we can obtain the following two conclusions. First, the performance of dynamic-meta-hashing and meta-hashing could be improved when $\alpha$ increases, especially for the complex and large-scale HRRS image dataset (i.e., NWPU-8). Taking all the results into account, we suggest $\alpha \in [0.8, 1.0]$ is an appropriate configuration for dynamic-meta-hashing and meta-hashing methods. Second, for the margin $m$, dynamic-meta-hashing could achieve well performance when $m \in [2.0, 32.0]$ and meta-hashing obtain the competitive performance when $m \in [24.0, 32.0]$. In particular, all the experiments about $m$ are conducted by the length of hash codes is fixed 32. Therefore,

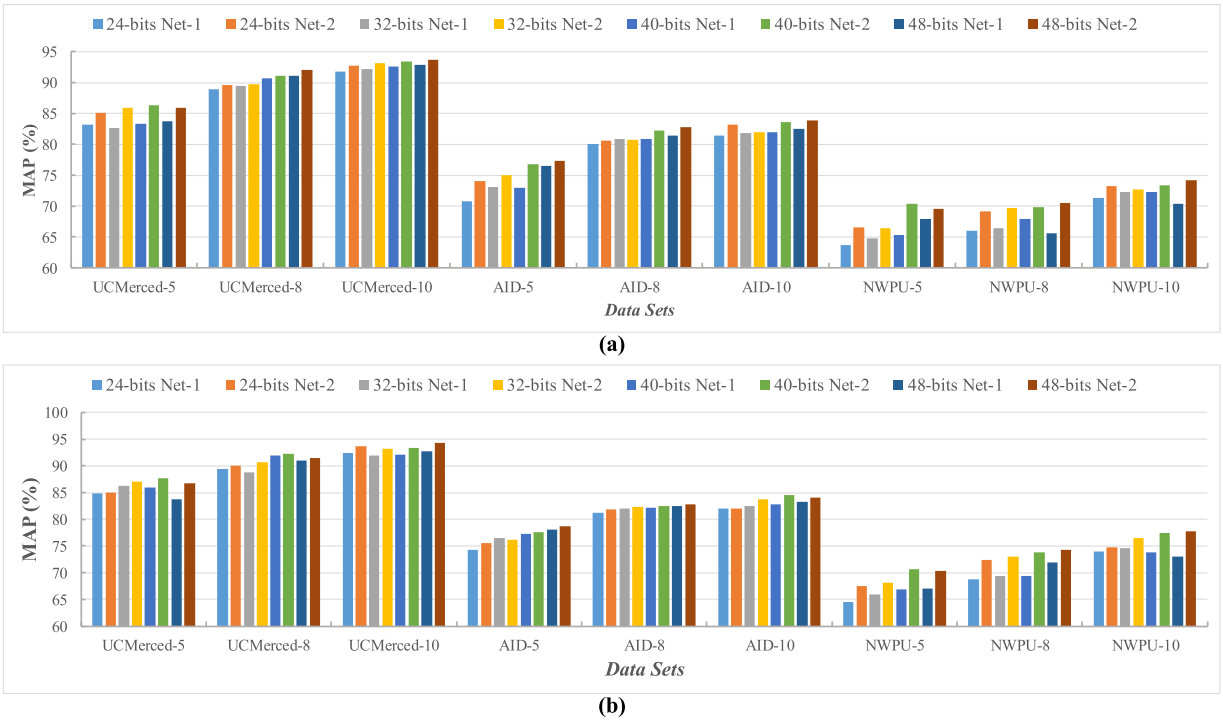


Fig. 9. Retrieval performance of Net1 (hashing net without the SAP-Conv block) and Net2 (hashing net) with diverse hash codes counted on different datasets. Two nets are trained by (a) meta-hashing algorithm and (b) dynamic-meta-hashing algorithm.

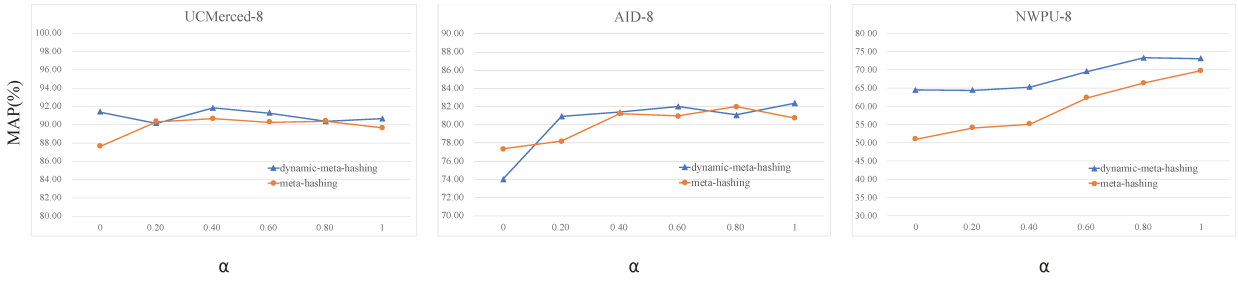


Fig. 10. Influence of parameter $\alpha$ on the UCMerced/AID/NWPU-8 datasets.

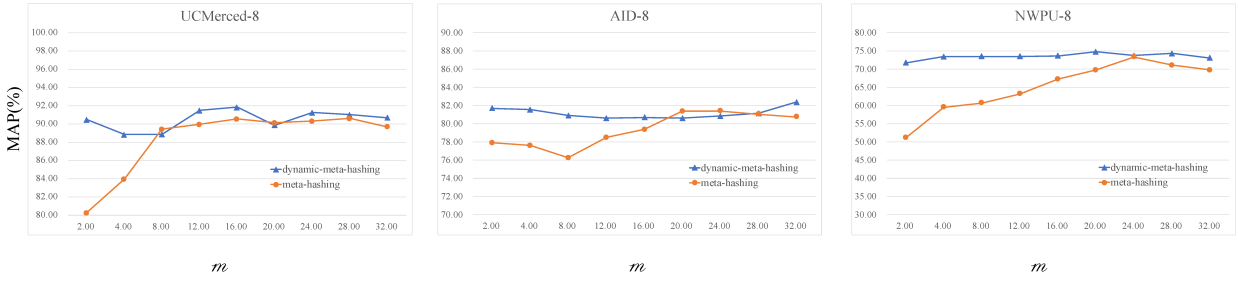


Fig. 11. Influence of parameter $m$ on the UCMerced/AID/NWPU-8 datasets.

to apply our methods easily in the HRRS image retrieval task, we suggest the value of $m$ can be equal to the code length.

### F. Retrieval Speed Discussion

As we mentioned in Section III, the retrieval speed based on hash codes is faster than the common features (i.e., continuous and dense visual representation). To validate it, we compare the time costs of retrieval based on hash codes and common features. Other factors, such as data reading and code extracting, are not taken into account since their influence on different methods is the same as each other. Before conducting the time-cost experiments, we assume that the following two preliminary works have been done. First, all of the compared methods in this article have been optimized. Second,

TABLE IV
RETRIEVAL SPEED BASED ON HASH CODES AND FEATURES (UNIT: MILLISECOND)

| Target Archive Size | | 10000 | 15000 | 20000 | 25000 | 30000 |
|---|---|---|---|---|---|---|
| Meta-hashing | 24-bits | 0.78 | 1.11 | 1.46 | 1.88 | 2.26 |
| | 32-bits | 0.91 | 1.35 | 1.73 | 2.38 | 2.80 |
| | 40-bits | 1.11 | 1.69 | 2.52 | 2.88 | 3.32 |
| | 48-bits | 1.32 | 1.92 | 2.67 | 3.23 | 3.69 |
| DMML | 24-bits | 2.26 | 3.42 | 4.63 | 5.84 | 6.78 |
| | 32-bits | 2.59 | 3.90 | 5.27 | 7.21 | 8.54 |
| | 40-bits | 2.92 | 4.51 | 6.17 | 8.27 | 9.70 |
| | 48-bits | 3.31 | 5.16 | 6.98 | 9.01 | 11.55 |

we have used the optimized models to generate the hash codes/common features for HRRS images within the database. To compare the retrieval speed extensively, we construct the target achieves with different sizes (i.e., 10 000, 15 000, 20 000, 25 000, 30 000) using the NWPU-8 dataset and randomly select 100 samples as the queries to compute the average retrieval time. Note that, we just compare the time cost of similarity calculation between meta-hashing (which retrieves based on hash codes) and DMML (which retrieves based on common features), since the time consummation of other methods is similar to each other. Particularly, for meta-hashing and DMML, we use Hamming distance and Euclidean distance to measure the similarities, respectively. Also, we conduct these experiments ten times and take the average values as the final results which are summarized in Table IV. From comparing the retrieval time between meta-hashing and DMML across different code lengths, we can easily find hash codes could well improve the retrieval speed distinctly, especially when the size of the target archive is large. The reason is that the Hamming distance can be computed by the efficient bit-wise XOR operation. The positive results demonstrate the hash learning methods could enhance the retrieval efficiency for large-scale HRRS image retrieval tasks.

## V. CONCLUSION

In this article, taking the characteristics of HRRS images into account, we develop a new supervised hash learning method (named meta-hashing) for HRRS image retrieval under the paradigm of FSL, in which the problem of hash code learning is completed in a meta-way. Therefore, not only the issue of few labeled training samples in the RS community can be addressed but also the generalization of our hashing model can be enhanced. First, considering the properties of diverse types of HRRS images, we design SAP-Conv block for fully capturing the multi-scale features within HRRS images and embed this block into our hashing net. Second, to learn an effective hash model by a few labeled training samples, we develop the meta-hashing method which accomplishes the hash learning based on the way of task learning so that the generalization of the learned model can be ensured. Meta-hashing can preserve the similarities between the samples in each task by optimizing three kinds of distances. These distances not only take the relationships between the support set and query set into account but also consider the relationships between the samples within the support set. The positive results demonstrate our hash learning method can get competitive retrieval performance under the scenario of a few training samples. Besides, to further improve the performance of meta-hashing, we construct the dynamic-meta-hashing method. It means the numbers of support and query images in each task are changeable during the training. The extensively experimental results counted on the widely used HRRS images datasets demonstrate the dynamic-meta-hashing method could enhance the retrieval performance to a certain extent.

Although the proposed meta-hashing achieves encouraging retrieval performance under the FSL scenario, its limitations cannot be neglected. For instance, the pseudo center generation strategy [see (12)] emphasizes the contributions of query samples and ignores support samples. Thus, the distribution of support samples is not fully considered, leading to outliers influencing this strategy. How to address this issue is one of our future works. Also, the hashing net in this article was constructed under the paradigm of supervised learning. To further relieve the model from the demands of labeled data, many other techniques can be adopted in the future, such as self-supervised learning and multi-task learning.

## REFERENCES

[1] J. Li, X. Huang, and J. Gong, "Deep neural network for remote-sensing image interpretation: Status and perspectives," *Nat. Sci. Rev.*, vol. 6, no. 6, pp. 1082–1086, May 2019.
[2] X. Tang, Q. Ma, X. Zhang, F. Liu, J. Ma, and L. Jiao, "Attention consistent network for remote sensing scene classification," *IEEE J. Sel. Topics Appl. Earth Observ. Remote Sens.*, vol. 14, pp. 2030–2045, 2021.
[3] X. Tang, X. Zhang, F. Liu, and L. Jiao, "Unsupervised deep feature learning for remote sensing image retrieval," *Remote Sens.*, vol. 10, no. 8, p. 1243, Aug. 2018.
[4] X. Tang, L. Jiao, W. J. Emery, F. Liu, and D. Zhang, "Two-stage reranking for remote sensing image retrieval," *IEEE Trans. Geosci. Remote Sens.*, vol. 55, no. 10, pp. 5798–5817, Oct. 2017.
[5] X. Tang *et al.*, "Hyperspectral image classification based on 3-D octave convolution with spatial–spectral attention network," *IEEE Trans. Geosci. Remote Sens.*, vol. 59, no. 3, pp. 2430–2447, Mar. 2020.
[6] J. Feng *et al.*, "Deep reinforcement learning for semisupervised hyperspectral band selection," *IEEE Trans. Geosci. Remote Sens.*, vol. 60, pp. 1–19, 2021.
[7] W. Zhou, S. Newsam, C. Li, and Z. Shao, "PatternNet: A benchmark dataset for performance evaluation of remote sensing image retrieval," *ISPRS-J. Photogramm. Remote Sens.*, vol. 145, pp. 197–209, Nov. 2018.
[8] Y. Gu, Y. Wang, and Y. Li, "A survey on deep learning-driven remote sensing image scene understanding: Scene classification, scene retrieval and scene-guided object detection," *Appl. Sci.*, vol. 9, no. 10, p. 2110, May 2019.
[9] W. Zhou, H. Li, and Q. Tian, "Recent advance in content-based image retrieval: A literature survey," 2017, *arXiv:1706.06064*.
[10] L. Jiao, X. Tang, B. Hou, and S. Wang, "SAR images retrieval based on semantic classification and region-based similarity measure for earth observation," *IEEE J. Sel. Topics Appl. Earth Observ. Remote Sens.*, vol. 8, no. 8, pp. 3876–3891, Sep. 2015.
[11] X. Tang, L. Jiao, and W. J. Emery, "SAR image content retrieval based on fuzzy similarity and relevance feedback," *IEEE J. Sel. Topics Appl. Earth Observ. Remote Sens.*, vol. 10, no. 5, pp. 1824–1842, May 2017.
[12] R. Waseem and Z. Wang, "Deep convolutional neural networks for image classification: A comprehensive review," *Neural Comput.*, vol. 29, no. 9, pp. 2352–2449, Aug. 2017.
[13] X. X. Zhu *et al.*, "Deep learning in remote sensing: A comprehensive review and list of resources," *IEEE Geosci. Remote Sens. Mag.*, vol. 5, no. 4, pp. 8–36, Dec. 2017.
[14] Y. Li, Z. Zhu, J.-G. Yu, and Y. Zhang, "Learning deep cross-modal embedding networks for zero-shot remote sensing image scene classification," *IEEE Trans. Geosci. Remote Sens.*, vol. 59, no. 12, pp. 10590–10603, Dec. 2021.

[15] J. Wang, H. T. Shen, J. Song, and J. Ji, "Hashing for similarity search: A survey," 2014, *arXiv:1408.2927*.

[16] J. Wang, T. Zhang, J. Song, N. Sebe, and H. T. Shen, "A survey on learning to hash," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 40, no. 4, pp. 769–790, Apr. 2017.

[17] J. Wang, W. Liu, S. Kumar, and S.-F. Chang, "Learning to hash for indexing big data—A survey," *Proc. IEEE*, vol. 104, no. 1, pp. 34–57, Jan. 2015.

[18] X. Luo, D. Wu, C. Chen, M. Deng, J. Huang, and X.-S. Hua, "A survey on deep hashing methods," 2020, *arXiv:2003.03369*.

[19] Y. Li, J. Ma, and Y. Zhang, "Image retrieval from remote sensing big data: A survey," *Inf. Fusion*, vol. 67, pp. 94–115, Mar. 2021.

[20] Y. Li, Y. Zhang, X. Huang, H. Zhu, and J. Ma, "Large-scale remote sensing image retrieval by deep hashing neural networks," *IEEE Trans. Geosci. Remote Sens.*, vol. 56, no. 2, pp. 950–965, Feb. 2017.

[21] X. Tang, C. Liu, J. Ma, X. Zhang, F. Liu, and L. Jiao, "Large-scale remote sensing image retrieval based on semi-supervised adversarial hashing," *Remote Sens.*, vol. 11, no. 17, p. 2055, 2019.

[22] C. Liu, J. Ma, X. Tang, F. Liu, X. Zhang, and L. Jiao, "Deep hash learning for remote sensing image retrieval," *IEEE Trans. Geosci. Remote Sens.*, vol. 59, no. 4, pp. 3420–3443, Apr. 2021.

[23] P. Li *et al.*, "Hashing nets for hashing: A quantized deep learning to hash framework for remote sensing image retrieval," *IEEE Trans. Geosci. Remote Sens.*, vol. 58, no. 10, pp. 7331–7345, Oct. 2020.

[24] H. Li *et al.*, "RSI-CB: A large scale remote sensing image classification benchmark via crowdsource data," 2017, *arXiv:1705.10450*.

[25] Y. Li, Y. Zhang, X. Huang, and J. Ma, "Learning source-invariant deep hashing convolutional neural networks for cross-source remote sensing image retrieval," *IEEE Trans. Geosci. Remote Sens.*, vol. 56, no. 11, pp. 6521–6536, Nov. 2018.

[26] Y. Wang, Q. Yao, J. T. Kwok, and L. M. Ni, "Generalizing from a few examples: A survey on few-shot learning," *ACM Comput. Surv.*, vol. 53, no. 3, pp. 1–34, 2020.

[27] M. Andrychowicz *et al.*, "Learning to learn by gradient descent by gradient descent," in *Proc. Adv. Neural Inf. Process. Syst.*, 2016, pp. 3981–3989.

[28] T. Hospedales, A. Antoniou, P. Micaelli, and A. Storkey, "Meta-learning in neural networks: A survey," 2020, *arXiv:2004.05439*.

[29] R. Caruana, "Multitask learning," *Mach. Learn.*, vol. 28, no. 1, pp. 41–75, 1998.

[30] Y. Zhang and Q. Yang, "A survey on multi-task learning," 2017, *arXiv:1707.08114*.

[31] C.-Y. Wu, R. Manmatha, A. J. Smola, and P. Krahenbuhl, "Sampling matters in deep embedding learning," in *Proc. IEEE Int. Conf. Comput. Vis. (ICCV)*, Oct. 2017, pp. 2840–2848.

[32] P. Cui, X. Wang, J. Pei, and W. Zhu, "A survey on network embedding," *IEEE Trans. Knowl. Data Eng.*, vol. 31, no. 5, pp. 833–852, May 2018.

[33] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2016, pp. 770–778.

[34] B. Demir and L. Bruzzone, "Hashing-based scalable remote sensing image search and retrieval in large archives," *IEEE Trans. Geosci. Remote Sens.*, vol. 54, no. 2, pp. 892–904, Sep. 2015.

[35] B. Kulis and K. Grauman, "Kernelized locality-sensitive hashing," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 34, no. 6, pp. 1092–1104, Jun. 2011.

[36] W. Liu, J. Wang, R. Ji, Y.-G. Jiang, and S.-F. Chang, "Supervised hashing with kernels," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jun. 2012, pp. 2074–2081.

[37] P. Li and P. Ren, "Partial randomness hashing for large-scale remote sensing image retrieval," *IEEE Geosci. Remote Sens. Lett.*, vol. 14, no. 3, pp. 464–468, Mar. 2017.

[38] T. Reato, B. Demir, and L. Bruzzone, "An unsupervised multicode hashing method for accurate and scalable remote sensing image retrieval," *IEEE Geosci. Remote Sens. Lett.*, vol. 16, no. 2, pp. 276–280, Feb. 2018.

[39] R. Fernandez-Beltran, B. Demir, F. Pla, and A. Plaza, "Unsupervised remote sensing image retrieval using probabilistic latent semantic hashing," *IEEE Geosci. Remote Sens. Lett.*, vol. 18, no. 2, pp. 256–260, Feb. 2021.

[40] J. Snell, K. Swersky, and R. Zemel, "Prototypical networks for few-shot learning," in *Proc. Adv. Neural Inf. Process. Syst.*, 2017, pp. 4077–4087.

[41] K. Nguyen and S. Todorovic, "Feature weighting and boosting for few-shot segmentation," in *Proc. IEEE/CVF Int. Conf. Comput. Vis. (ICCV)*, Oct. 2019, pp. 622–631.

[42] J.-M. Perez-Rua, X. Zhu, T. M. Hospedales, and T. Xiang, "Incremental few-shot object detection," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2020, pp. 13846–13855.

[43] X. Zhou, D. Wang, and P. Krähenbühl, "Objects as points," 2019, *arXiv:1904.07850*.

[44] S. Motiian, Q. Jones, S. Iranmanesh, and G. Doretto, "Few-shot adversarial domain adaptation," in *Proc. Adv. Neural Inf. Process. Syst.*, 2017, pp. 6670–6680.

[45] F. Zhao, J. Zhao, S. Yan, and J. Feng, "Dynamic conditional networks for few-shot learning," in *Proc. Eur. Conf. Comput. Vis. (ECCV)*, 2018, pp. 19–35.

[46] R. Zhang, T. Che, Z. Ghahramani, Y. Bengio, and Y. Song, "MetaGAN: An adversarial approach to few-shot learning," in *Proc. Adv. Neural Inf. Process. Syst.*, 2018, pp. 2365–2374.

[47] B. Liu, X. Yu, A. Yu, P. Zhang, G. Wan, and R. Wang, "Deep few-shot learning for hyperspectral image classification," *IEEE Trans. Geosci. Remote Sens.*, vol. 57, no. 4, pp. 2290–2304, Apr. 2018.

[48] X. Ma, S. Ji, J. Wang, J. Geng, and H. Wang, "Hyperspectral image classification based on two-phase relation learning network," *IEEE Trans. Geosci. Remote Sens.*, vol. 57, no. 12, pp. 10398–10409, Dec. 2019.

[49] M. Rostami, S. Kolouri, E. Eaton, and K. Kim, "Deep transfer learning for few-shot SAR image classification," *Remote Sens.*, vol. 11, no. 11, p. 1374, Jun. 2019.

[50] W. Li *et al.*, "Classification of high-spatial-resolution remote sensing scenes method using transfer learning and deep convolutional neural network," *IEEE J. Sel. Topics Appl. Earth Observ. Remote Sens.*, vol. 13, pp. 1986–1995, 2020.

[51] Y. Qu, R. K. Baghbaderani, and H. Qi, "Few-shot hyperspectral image classification through multitask transfer learning," in *Proc. 10th Workshop Hyperspectral Imag. Signal Process., Evol. Remote Sens. (WHISPERS)*, Sep. 2019, pp. 1–5.

[52] R. Hadsell, S. Chopra, and Y. LeCun, "Dimensionality reduction by learning an invariant mapping," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, vol. 2, Jun. 2006, pp. 1735–1742.

[53] F. Yu and V. Koltun, "Multi-scale context aggregation by dilated convolutions," 2015, *arXiv:1511.07122*.

[54] D. Hien. (Apr. 2017). *A Guide to Receptive Field Arithmetic for Convolutional Neural Networks*. [Online]. Available: http://www.medium.com

[55] Y. Yang and S. Newsam, "Bag-of-visual-words and spatial extensions for land-use classification," in *Proc. 18th Int. Conf. Adv. Geographic Inf. Syst. (SIGSPATIAL)*, 2010, pp. 270–279.

[56] G.-S. Xia *et al.*, "AID: A benchmark data set for performance evaluation of aerial scene classification," *IEEE Trans. Geosci. Remote Sens.*, vol. 55, no. 7, pp. 3965–3981, Jul. 2017.

[57] G. Cheng, J. Han, and X. Lu, "Remote sensing image scene classification: Benchmark and state of the art," *Proc. IEEE*, vol. 105, no. 10, pp. 1865–1883, Oct. 2017.

[58] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, "ImageNet: A large-scale hierarchical image database," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jun. 2009, pp. 248–255.

[59] A. Paszke *et al.*, "PyTorch: An imperative style, high-performance deep learning library," in *Proc. Adv. Neural Inf. Process. Syst.*, 2019, pp. 8024–8035.

[60] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," 2014, *arXiv:1412.6980*.

[61] H. Liu, R. Wang, S. Shan, and X. Chen, "Deep supervised hashing for fast image retrieval," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2016, pp. 2064–2072.

[62] J. Zhang and Y. Peng, "SSDH: Semi-supervised deep hashing for large scale image retrieval," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 29, no. 1, pp. 212–225, Jan. 2017.

[63] S. Roy, E. Sangineto, B. Demir, and N. Sebe, "Metric-learning-based deep hashing network for content-based retrieval of remote sensing images," *IEEE Geosci. Remote Sens. Lett.*, vol. 18, no. 2, pp. 226–230, Feb. 2021.

[64] G. Chen, T. Zhang, J. Lu, and J. Zhou, "Deep meta metric learning," in *Proc. IEEE/CVF Int. Conf. Comput. Vis. (ICCV)*, Oct. 2019, pp. 9547–9556.

[65] T. Mei, Y. Rui, S. Li, and Q. Tian, "Multimedia search reranking: A literature survey," *ACM Comput. Surveys*, vol. 46, no. 3, pp. 1–38, Jan. 2014.

[66] L. van der Maaten and G. Hinton, "Visualizing data using t-SNE," *J. Mach. Learn. Res.*, vol. 9, pp. 2579–2605, Nov. 2008.

**Xu Tang** (Senior Member, IEEE) received the B.Sc., M.Sc., and Ph.D. degrees in electronic circuit and system from Xidian University, Xi'an, China, in 2007, 2010, and 2017, respectively, and the joint Ph.D. degree from the University of Colorado at Boulder, Boulder, CO, USA, in 2016, along with Prof. W. J. Emery.

He is currently an Associate Professor with the Key Laboratory of Intelligent Perception and Image Understanding, Ministry of Education, Xidian University. His research interests include remote sensing image content-based retrieval and reranking, hyperspectral image processing, remote sensing scene classification, and object detection.

**Yuqun Yang** (Student Member, IEEE) received the B.Sc. degree in information and computing science from the Xi'an University of Technology, Xi'an, China, in 2019. He is currently pursuing the Ph.D. degree with the School of Artificial Intelligence, Xidian University, Xi'an.

His research interests include machine learning, object detection, and image classification.

**Jingjing Ma** (Member, IEEE) received the B.S. and Ph.D. degrees from Xidian University, Xi'an, China, in 2004 and 2012, respectively.

She is currently an Associate Professor with the Key Laboratory of Intelligent Perception and Image Understanding, Ministry of Education, Xidian University. Her research interests include computational intelligence and image understanding.

**Yiu-Ming Cheung** (Fellow, IEEE) received the Ph.D. degree from the Department of Computer Science and Engineering, The Chinese University of Hong Kong, Hong Kong, in 2000.

He is currently a Full Professor with the Department of Computer Science, Hong Kong Baptist University, Hong Kong. His research interests include machine learning, data science, pattern recognition, visual computing, and optimization.

Dr. Cheung is also a fellow of the International Engineering and Technology Institute (IET) and the British Computer Society (BCS). He serves as an Associate Editor for several prestigious journals, including IEEE TRANSACTIONS ON CYBERNETICS, IEEE TRANSACTIONS ON EMERGING TOPICS IN COMPUTATIONAL INTELLIGENCE, IEEE TRANSACTIONS ON COGNITIVE AND DEVELOPMENTAL SYSTEMS, IEEE TRANSACTIONS ON NEURAL NETWORKS AND LEARNING SYSTEMS, and *Pattern Recognition*.

**Chao Liu** received the B.Eng. degree in electronic science and technology from Xidian University, Xi'an, China, in 2013, where he is currently pursuing the master's degree with the Institute of Artificial Intelligence.

His research interests include machine learning, meta-learning, and image retrieval.

**Fang Liu** (Member, IEEE) was born in China in 1990. She received the B.S. degree in information and computing science from Henan University, Kaifeng, China, in 2012, and the Ph.D. degree in intelligent information processing from Xidian University, Xi'an, China, in 2018.

She is currently an Associate Professor with the Nanjing University of Science and Technology, Nanjing, China. Her research interests include deep learning, object detection, polarimetric SAR image classification, and change detection.

**Xiangrong Zhang** (Senior Member, IEEE) received the B.S. and M.S. degrees from the School of Computer Science, Xidian University, Xi'an, China, in 1999 and 2003, respectively, and the Ph.D. degree from the School of Electronic Engineering, Xidian University, in 2006.

From January 2015 to March 2016, she was a Visiting Scientist with the Computer Science and Artificial Intelligence Laboratory, Massachusetts Institute of Technology, Cambridge, MA, USA. She is currently a Professor with the Key Laboratory of Intelligent Perception and Image Understanding, Ministry of Education, Xidian University. Her research interests include pattern recognition, machine learning, and remote sensing image analysis and understanding.

**Licheng Jiao** (Fellow, IEEE) received the B.S. degree in high voltage from Shanghai Jiao Tong University, Shanghai, China, in 1982, and the M.S. and Ph.D. degrees in electronic engineering from Xi'an Jiaotong University, Xi'an, China, in 1984 and 1990, respectively.

From 1984 to 1986, he was an Assistant Professor with the Civil Aviation Institute of China, Tianjin, China. From 1990 to 1991, he was a Post-Doctoral Fellow with the Key Laboratory for Radar Signal Processing, Xidian University, Xi'an, where he is currently the Director of the Key Laboratory of Intelligent Perception and Image Understanding, Ministry of Education of China. He has authored or coauthored more than 200 scientific articles. His research interests include signal and image processing, nonlinear circuits and systems theory, wavelet theory, natural computation, and intelligent information processing.

Dr. Jiao is a member of the IEEE Xi'an Section Executive Committee and an Executive Committee Member of the Chinese Association of Artificial Intelligence. He is the Chairperson of the Awards and Recognition Committee.