Metric-Learning-Based Deep Hashing Network for Content-Based Retrieval of Remote Sensing Images

Subhankar Roy, *Student Member, IEEE*, Enver Sangineto, Begüm Demir, *Senior Member, IEEE*, and Nicu Sebe, *Senior Member, IEEE*

***Abstract*— Hashing methods have recently been shown to be very effective in the retrieval of remote sensing (RS) images due to their computational efficiency and fast search speed. Common hashing methods in RS are based on hand-crafted features on top of which they learn a hash function, which provides the final binary codes. However, these features are not optimized for the final task (i.e., retrieval using binary codes). On the other hand, modern deep neural networks (DNNs) have shown an impressive success in learning optimized features for a specific task in an end-to-end fashion. Unfortunately, typical RS data sets are composed of only a small number of *labeled* samples, which make the training (or fine-tuning) of big DNNs problematic and prone to overfitting. To address this problem, in this letter, we introduce a metric-learning-based hashing network, which: 1) implicitly uses a big, pretrained DNN as an intermediate representation step without the need of retraining or fine-tuning; 2) learns a semantic-based metric space where the features are optimized for the target retrieval task; and 3) computes compact binary hash codes for fast search. Experiments carried out on two RS benchmarks highlight that the proposed network significantly improves the retrieval performance under the same retrieval time when compared to the state-of-the-art hashing methods in RS.**

***Index Terms*— Content-based image retrieval (CBIR), deep hashing, metric learning, remote sensing (RS).**

I. Introduction

THE advancement of satellite technology has resulted in an explosive growth of remote sensing (RS) image archives. Content-based image retrieval (CBIR) methods in RS aim to retrieve those archive images, which are the most similar with respect to a given query image. Traditional nearest neighbor (NN) search algorithms (which exhaustively compare the query image with all the images in the archive) are time-consuming and, thus, prohibitive for large-scale RS image retrieval problems [1]. To alleviate this issue, hashing-based search techniques have been recently proposed in RS. A hashing-based method involves learning a hash function, $\mathbf{b} = h(\mathbf{d})$, which maps a high-dimensional image descriptor $\mathbf{d} \in \mathbb{R}^D$ into a low-dimensional binary code $\mathbf{b} \in \{0, 1\}^K$, where $D \gg K$. Besides improving the computational efficiency, hashing methods also reduce the storage costs significantly. For instance, locality-sensitive hashing (LSH) [2] learns n different hash functions $[h_1, h_2, \ldots, h_n]$ by choosing n random projecting vectors obtained from a multivariate Gaussian distribution. In the context of CBIR in RS, two hashing-based methods have been introduced in [3]–[5]. Demir and Bruzzone [3] learn two hash functions in the kernel space using hand-crafted features (SIFT) and a bag-of-visual-words representation. Reato *et al.* [4] extend these hashing methods by describing each image with multihash codes and using the spectral histograms of the image regions. However, the effectiveness of the binary codes relies on both the approach used for the hash-function and the adopted image descriptors. Specifically, common hand-crafted features are not optimized for the retrieval task and have a limited capability to accurately represent the high-level semantic content of RS images.

The limitations of hand-crafted features are nowadays addressed by using deep convolutional neural networks (CNNs), which learn a feature space directly optimized for the final task (e.g., classification). For instance, Li *et al.* [5] introduce a deep Hashing neural network (DHNN) to address CBIR in RS. DHNN jointly learns semantically accurate deep features and binary hash codes by means of a cross-entropy loss function. However, entropy-based losses are particularly effective in classification problems, but less effective in defining a *metric space* which clusters together similar images, a concept which is crucial in a CBIR framework. Specifically, the absence of a margin threshold between positive and negative samples leads to a poor generalization. As a result, to achieve a high retrieval performance, DHNN requires long hash codes and a large amount of *annotated* training images, which are difficult to collect in common RS archives.

To address these issues, we propose to use an intermediate representation, obtained with a big CNN (Inception Net [6]) pretrained on ImageNet and *not* fine-tuned. In our approach, Inception Net is not retrained; thus, we avoid overfitting risks when using small labeled data sets, as those commonly available in RS. Inception Net is pretrained on ImageNet, a data set of more than one million (labeled), non-RS images. The visual knowledge acquired by the Inception Net on this data set is exploited to extract an intermediate representation of our RS images. Using this representation as input, we *train* a second network whose final features are jointly optimized (using a combination of different losses) for both the retrieval task and

Manuscript received April 2, 2019; revised October 1, 2019 and January 23, 2020; accepted February 13, 2020. This work was supported by the European Research Council (ERC) through the ERC-2017-STG BigEarth Project under Grant 759764. *(Corresponding author: Begüm Demir.)*

Subhankar Roy, Enver Sangineto, and Nicu Sebe are with the Department of Information Engineering and Computer Science, University of Trento, 38123 Trento, Italy.

Begüm Demir is with the Faculty of Electrical Engineering and Computer Science, Technische Universität Berlin Fakultät IV Elektrotechnik und Informatik, 10587 Berlin, Germany (e-mail: demir@tu-berlin.de).

Color versions of one or more of the figures in this letter are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/LGRS.2020.2974629

the hash-code production. In summary, the contributions of our **M**etr**i**c-**L**earning-based Deep H**a**shing **N**etwork (MiLaN) are as follows.

1) We use as input an intermediate feature representation to alleviate overfitting problems on small RS annotated data sets.
2) We train a deep neural network (DNN) on top of these intermediate features, whose final feature representation is jointly optimized for both a CBIR task and a hash-code production task. This letter extends our work presented in [7] introducing a detailed ablation study of MiLaN, a latent-space visualization, and additional experiments, including testing on another RS data set.

II. Metric-Learning-Based Deep Hashing Network

We assume that a small set of training images is available: $\mathcal{I} = \{x_1, x_2, \ldots, x_P\}$, where each x_i is associated with a corresponding class label $y_i \in \mathcal{Y} = \{y_1, y_2, \ldots, y_P\}$. Our goal is to learn a hash function encoding an image into a binary code: $h : \mathcal{I} \rightarrow \{0, 1\}^K$, where K is the number of bits in the hash code. Importantly, h should capture the semantics of the images in a similarity space. This is achieved using a specific loss function, which clusters together semantically similar samples. Specifically, we use a *triplet loss* [8], which learns a metric space such that the Euclidean distance between two points in this space faithfully corresponds to the visual similarity between the corresponding pair of images in the pixel space. Moreover, we use two additional losses: 1) a representation penalty loss, which pushes the activations of the last layer of the network to be binary and 2) a bit-balancing loss, which encourages the network to produce hash codes having (on expectation) an equivalent number of 0 and 1s. The latter is important to guarantee that all the final bit codes are effectively useful for the binary-based retrieval task.

As mentioned in Section I, the success of DNNs in vision tasks relies on the use of big CNNs, which extract visual information from raw-pixel data. However, these big networks are prone to overfit when trained using the small labeled data set, which is the common case in RS frameworks. In order to solve this problem, we propose to use a big CNN (Inception Net [6]) pretrained on ImageNet, without retraining. Specifically, the Inception Net is used to extract an intermediate representation of our RS images. Then, a smaller network (f) is trained using this representation as input in order to learn our final hash function. In more detail, each image in $\mathcal{I}$ is fed to the Inception Net [6], and we use the last layer ($Pool_3$) right before the softmax layer as our intermediate feature representation. This layer is composed of 2048 neurons, whose activations represent the result of an average pooling on the convolutional feature maps of the layer before (Fig. 1).

Let $\mathcal{G} = \{\mathbf{g}_1, \mathbf{g}_2, \ldots, \mathbf{g}_P\}$, $\mathbf{g}_i \in \mathbb{R}^{2048}$ be the set of the intermediate features extracted from $\mathcal{I}$. The elements of $\mathcal{G}$ are fed to the network f, whose weights are *randomly initialized*. The goal of f is to learn a mapping from the intermediate features to a metric space, which is semantically significant for the specific RS retrieval task: $f : \mathbb{R}^{2048} \rightarrow \mathbb{R}^K$. The real-valued activations of the last layer of f are finally quantized using a simple thresholding to obtain the binary hash codes.

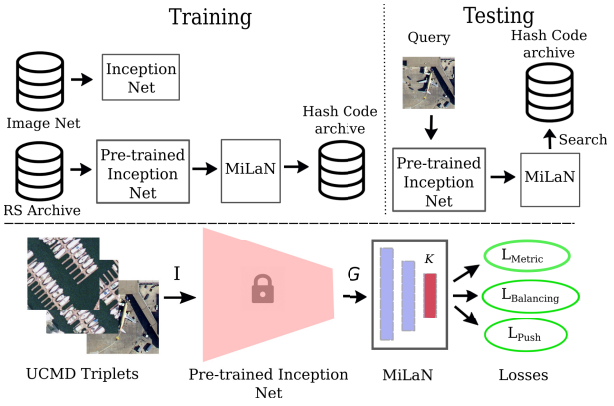


Fig. 1. (Top) Overall pipeline. (Bottom) Inception Net is used to extract an intermediate image representation, which is then fed to our MiLaN to obtain binary codes of length K.

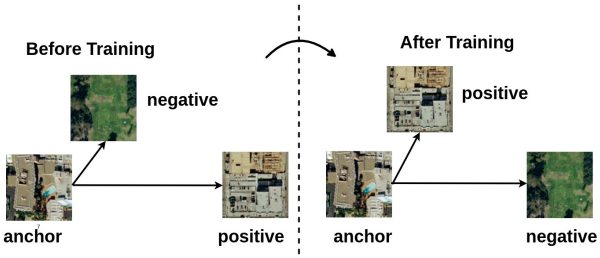


Fig. 2. Intuition behind the triplet loss: after training, a positive sample is "moved" closer to the anchor sample than the negative samples of the other classes.

The main loss used to train f is the triplet loss, which is based on the intuition that images with the same label (hence, sharing the same visual semantics) should lie closer to each other in the learned metric space more than those images having different labels (see Fig. 2). In more detail, a triplet ($\mathcal{T}$) of samples is sampled from $\mathcal{G}$: $\mathcal{T} = \{(\mathbf{g}_i^a, \mathbf{g}_i^p, \mathbf{g}_i^n)\}$, where $\mathbf{g}_i^a$ (called *anchor*) is sampled randomly from $\mathcal{G}$ together with its associated class label y_i; $\mathbf{g}_i^p$ (called the *positive* sample) is drawn among the samples having the same class labels (y_i) of the anchor; and $\mathbf{g}_i^n$ (called the *negative* sample) is again chosen randomly from $\mathcal{G}$ such that it belongs to a class y_j, where $y_j \neq y_i$. For training the network with stochastic gradient descent (SGD), a mini-batch of M triplets is sampled, and the following *triplet loss* is minimized:

$$\mathcal{L}_{\text{Metric}} = \sum_{i=1}^{M} \max\big(0, \big\| f\left(\mathbf{g}_i^a\right) - f\left(\mathbf{g}_i^p\right)\big\|_2^2 - \big\| f\left(\mathbf{g}_i^a\right) - f\left(\mathbf{g}_i^n\right)\big\|_2^2 + \alpha\big) \quad (1)$$

where α is the minimum *margin* threshold, which is forced between the positive and negative distances.

The network f is composed of 3 fully connected layers with 1024, 512, and K neurons each, being K the number of desired hash bits. We use LeakyReLU in the two hidden layers to allow negative gradients to flow during the backward pass and a sigmoid activation in the final layer to restrict the output activations in [0, 1].

In order to push the final real activations toward the extremities of the sigmoid range, and inspired by Yang *et al.* [9], we use a second loss, whose goal is to maximize the sum of

the squared errors between the output-layer activations and the value 0.5

$$\mathcal{L}_{\text{Push}} = -\frac{1}{K}\sum_{i=1}^{M} \|f(\mathbf{g}_i) - 0.5\mathbf{1}\|^2 \tag{2}$$

where $\mathbf{1}$ is a vector of 1s having dimension K.

Also inspired by Yang *et al.* [9], we adopt a third loss, which encourages each output neuron to fire with a 50% chance. This means that the binary code representations of the images will have (on average) a balanced number of 0 and 1s; hence, all the bits of the code are equally used. The balancing loss is

$$\mathcal{L}_{\text{Balancing}} = \sum_{i=1}^{M} (\text{mean}(f(\mathbf{g}_i)) - 0.5)^2 \tag{3}$$

where $mean(f(\mathbf{g}_i))$ is the mean of the output activations.

The final objective function is a weighted combination of the three losses

$$\mathcal{L} = \mathcal{L}_{\text{Metric}} + \lambda_1 \mathcal{L}_{\text{Push}} + \lambda_2 \mathcal{L}_{\text{Balancing}} \tag{4}$$

where λ_1 and λ_2 weight the relative loss importance.

When training of f is done, the final hashing function h is obtained by quantizing the values in $\mathbb{R}^K$. For a given archive image x, let $\mathbf{g}$ be its intermediate feature obtained using Inception Net and $\mathbf{v} = f(\mathbf{g})$. The final binary code $\mathbf{b} = h(f(\mathbf{g}))$ is given by

$$\mathbf{b}_n = (\text{sign}(\mathbf{v}_n - 0.5) + 1)/2, \quad 1 \le n \le K. \tag{5}$$

Finally, in order to retrieve an image x_j, which is semantically similar to a query image x_q, we use the Hamming distance between $h(x_q)$ and $h(x_j)$. Note that, with a slight abuse of notation, in the rest of this letter, we write $h(x)$ to mean $h(f(\mathbf{g}))$, where $\mathbf{g}$ is the intermediate feature corresponding to x. During retrieval, the Hamming distance is computed between the query $h(x_q)$ and each image in the archive, and the obtained distances are sorted in the ascending order of the magnitude. The top-k instances with the lowest distances are retrieved.

III. Data Set Description and Design of the Experiments

In our experiments, we use two RS benchmarks. The first archive is the widely used University of California Merced [10] (UCMD) archive that contains 2100 aerial images from 21 different land-cover categories, where each category includes 100 images. The pixel size of each image in the archive is 256×256, and the spatial resolution is 30 cm. The second benchmark archive is the aerial image data set [11] (AID), which is much larger with respect to the UCMD archive. The images are extracted from the Google Earth, and each image in the archive is a section of 600×600 pixels. AID contains 10 000 images from 30 different categories, and the number of images in each category varies between 220 and 420. The spatial resolution of the images varies between 50 cm and 8 m.

MiLaN[1] was trained using a mini-batch of triplets having a cardinality $M = 30$. For a much larger archive, we suggest to include only those triplets in the mini-batch, which contain semihard negatives (i.e., only those negative samples that violate the threshold margin α) that allow for a faster convergence of the network. The value of the threshold margin α was set to 0.2, which is determined by cross-validation, using a validation set composed of 20 images per class. The hyperparameters $\lambda_1 = 0.001$ and $\lambda_2 = 1$ have also been chosen using the same validation set. We used the Adam Optimizer with a small learning rate $\eta = 10^{-4}$. The other two Adam hyperparameters β_1 and β_2 were set to 0.5 and 0.9, respectively.

We compare MiLaN with the state-of-the-art supervised hashing methods, kernel-based supervised LSH (KSLSH) [3] and DHNN [5]. However, due to the lack of a publicly available code for DHNN, we can only report its results for the UCMD archive. A Gaussian kernel is used for KSLSH. We also use, as a baseline, an NN search with our intermediate features. Specifically, "NN-Inception feat. (Euclidean)" refers to an NN search using the Inception Net features and the Euclidean distance. The comparison with MiLaN shows the advantage of learning a metric space on top of these features. Moreover, "NN-Inception feat. (Hamming)" refers to an NN search *using binary hash codes* (of length $K = 2048$). These hash codes are computed using the same thresholding rule as in (5). The results of each method are provided in terms of: 1) mean average precision (mAP) and 2) computation time. Specifically, mAP at k is based on the top-k retrieved images. We consider two different scenarios. In the first, no data augmentation is used, and 60% of all the images of each category are used for training and cross-validation, while the rest is used for testing. In the second scenario, we use data augmentation obtained by means of basic geometric transformations.

IV. Experimental Results

A. Results: the UCMD Data Set

Table I shows that the MiLaN mAP is significantly higher than "NN-Inception feat. (Euclidean)" (for all the values of K), which shows the advantage of learning a metric space. Moreover, the retrieval time is also drastically reduced by the fact that our network outputs binary hash codes and an efficient Hamming distance can be computed over the archive elements. On the other hand, "NN-Inception feat. (Hamming)" is faster (as expected), but the plain binarization of the CNN features leads to a *drastic* loss of information, which explains the results inferior to "NN-Inception feat. (Euclidean)." This shows the importance of *learning* how to compute effective binary codes, which, in our case, is the result of applying (2) and (3). The last row of Table I reports the results obtained using the final MiLaN features *before* quantization (5). As expected, retrieval in an $\mathbb{R}^K$ space is more effective than that in $\{0, 1\}^K$. However, hash codes ("Our MiLaN") significantly improve the searching time with a negligible accuracy drop. This shows that MiLaN is able to learn features, which can be easily binarized while mostly preserving the metric space.

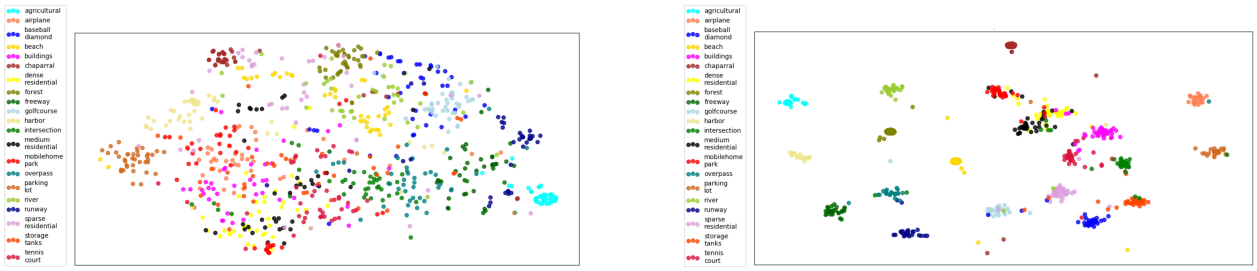
In order to show the contribution of the hash-code production losses, we report below the results obtained, respectively, with $K = 16, 24$, and 32, when: 1) $\lambda_1 = \lambda_2 = 1$: 0.231, 0.524, and 0.408 and 2) $\lambda_1 = \lambda_2 = 0.001$: 0.823, 0.851, and 0.907.

[1]Our code is available at: https://github.com/MLEnthusiast/MHCLN

TABLE I

mAP AT 20 AND AVERAGE RETRIEVAL TIME IN THE UCMD ARCHIVE

Methods	mAP	Time (in ms)	# Hash Bits K					
			K=16		K=24		K=32	
			mAP	Time (in ms)	mAP	Time (in ms)	mAP	Time (in ms)
NN-Inception feat. (Euclidean)	0.724	45.1	-	-	-	-	-	-
NN-Inception feat. (Hamming)	0.359	10.1	-	-	-	-	-	-
KSLSH [3]	-	-	0.557	25.3	0.594	25.5	0.630	25.6
Our **MiLaN**	-	-	**0.875**	25.3	**0.890**	25.5	**0.904**	25.6
Our **MiLaN** (Eucledian)	-	-	**0.903**	35.3	**0.894**	35.8	**0.916**	36.0



(a) (b)

Fig. 3. t-SNE 2-D scatterplots comparing the 2-D projection of the K-dimensional binary hash codes of the test images in the UCMD archive. Each color in the plot represents a land-cover category. This is best viewed in color with maximum zoom. (a) t-SNE scatterplot: KSLSH [3]. (b) t-SNE scatterplot: MiLaN.

TABLE II

mAP WITH DATA AUGMENTATION FOR THE UCMD ARCHIVE

	DPSH [12]	DHNN [5]	Our **MiLaN**
K = 32	0.748	0.939	**0.977**
K = 64	0.817	0.971	**0.991**

MiLaN, with $K = 24$, also significantly outperforms (+29.6% mAP) KSLSH with a similar retrieval time. Qualitative results are shown in Fig. 4, where, using as query an image of an "airplane," MiLaN semantically retrieves more similar images than KSLSH does.

We perform experiments on the UCMD archive also under the second scenario, where we consider data augmentation as suggested by Li *et al.* [5]. Accordingly, we replicate the experimental setup of DHNN [5] by augmenting the original data set with images rotated by 90°, 180°, and 270°, thus yielding 8400 images. Then, out of the 8400 images, 7400 are randomly chosen for training the network, and the remaining 1000 images are used for evaluation. It is worth noting that the 7400 training images are also used as a search set as suggested by Li *et al.* [5]. Table II shows the comparison of DHNN [5] with our MiLaN using mAP. These results show that MiLaN requires a smaller K to reach a higher mAP. For instance, when $K = 32$, the mAP of MiLaN is 3.8% higher than that of DHNN. Moreover, the mAP of MiLaN with $K = 32$ is again 0.6% higher than that of DHNN with twice the number of hash bits, i.e., $K = 64$. We highlight that the use of such an experimental setup, as proposed by Li *et al.* [5], may lead to the identical, but rotated variants of the same image being present in both the training and the test sets, thus leading to over-saturated results. In fact, the network can memorize the test samples (rotated variants), which are present in the training set. However, we adopt this training and evaluation protocol in order to fairly compare with [5].

To visualize the K-dimensional binary codes corresponding to the test images, we perform a t-distributed stochastic neighbor embedding (t-SNE), which projects these representations

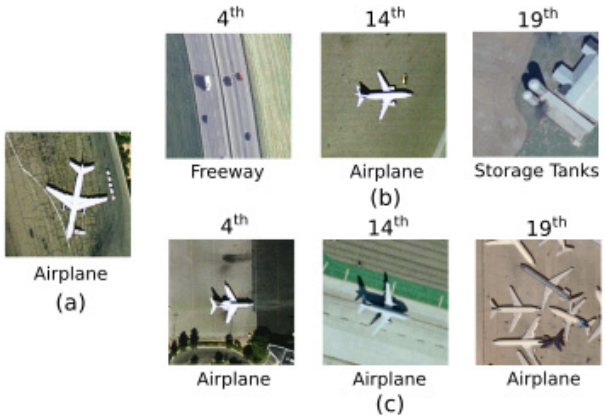


Fig. 4. (a) Query image. (b) Images retrieved by KSLSH [3]. (c) Images retrieved by our MiLaN.

in a 2-D space. Fig. 3(a) shows the results in the case of KSLSH. Note that, although the samples of the "agricultural" category stand out clearly from the rest of the samples belonging to the other categories, the remainder of the samples are mostly cluttered in one big heap having a minimal interclass separation. This shows that KSLSH is not discriminative enough. In contrast, Fig. 3(b) shows that MiLaN leads to samples exhibiting much more compact clusters. For instance, the categories "freeway" and "overpass", which exhibit similar visual patterns, lie close to each other and yet far enough to have their own well-separated clusters.

We also analyze the impact of the length of the hash codes on the retrieval performance. Fig. 5(a) shows that both for KSLSH and for MiLaN, the mAP at 20 increases with an increase in the number of the hash bits. However, the average precision curve for MiLaN with respect to all the K values stays considerably above the KSLSH results. This indicates that the proposed network efficiently maps semantic information into its discriminative hash codes with varying lengths, thus significantly outperforming the corresponding KSLSH results. Finally, we train MiLaN with different train:test splits

TABLE III
mAP AT 20 AND AVERAGE RETRIEVAL TIME FOR THE AID ARCHIVE

Methods	mAP	Time (in ms)	# Hash Bits K					
			K=16		K=24		K=32	
			mAP	Time (in ms)	mAP	Time (in ms)	mAP	Time (in ms)
NN-Inception feat. (Euclidean)	0.719	145.1	-	-	-	-	-	-
NN-Inception feat. (Hamming)	0.402	60.2	-	-	-	-	-	-
KSLSH [3]	-	-	0.426	115.3	0.467	116.1	0.495	117.5
Our **MiLaN**	-	-	**0.876**	117.5	**0.891**	116	**0.926**	114.5

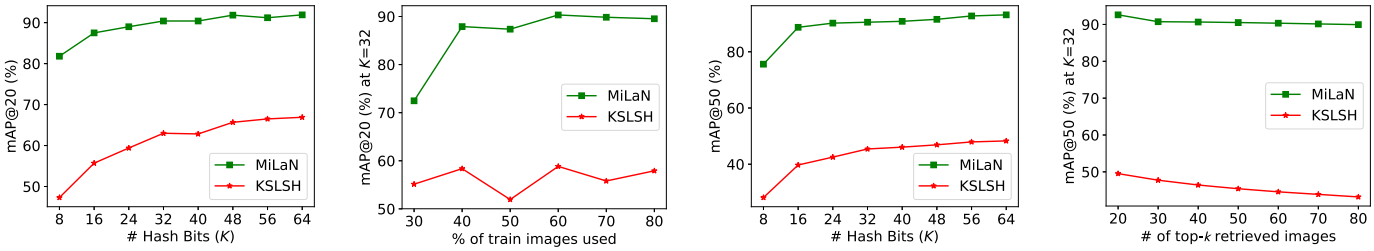


Fig. 5. Comparative analysis of KSLSH and MiLaN. (a) and (b) mAP at 20 curves of KSLSH and MiLaN on UCMD for (a) different numbers of hash bits and (b) different train: test ratios at $K = 32$. (c) and (d) mAP at 50 curves of KSLSH and MiLaN on AID for (c) different numbers of hash bits and (d) different numbers of top-k retrieved images at $K = 32$.

and report mAP at 20 at $K = 32$ in Fig. 5(b). For instance, Fig. 5(b) shows that the mAP obtained by MiLaN using only 30% of the training images is much higher than the mAP obtained by KSLSH with 80% of the total training images.

B. Results: the AID Data Set

In the experiments on the AID archive, we split the images of each category with a 60:40 train:test split ratio. The results of this archive demonstrate similar trends as those in UCMD. In summary, MiLaN achieves a higher mAP at 20 with respect to all the other methods (Table III). For instance, with $K = 16$, the mAP at 20 of MiLaN is 45% higher than that of KSLSH. A similar trend is kept between these two methods for higher values of K. Moreover, the AID data set results confirm all the observations pointed out in Section IV-A concerning the comparison among MiLaN, "NN-Inception feat. (Euclidean)," and "NN-Inception feat. (Hamming)," again confirming the advantage of simultaneously learning a metric space and the hash codes on top of the adopted CNN features.

Finally, we report an ablation study conducted on the AID archive. Fig. 5(c) shows that with increasing values of K, the map at 50 values of MiLaN increases monotonically and outperforms KSLSH for all values of K. Similarly, in Fig. 5(d), we observe that when the number of the top-k retrieved images increases, the map at 50 value slightly decreases for MiLaN but quite significantly for KSLSH.

V. CONCLUSION

In this letter, we have introduced a metric-learning-based deep hashing method for fast and accurate RS CBIR. The proposed approach is based on an intermediate representation of the RS images obtained using an external, not-retrained CNN (in our case, we used Inception Net, pretrained on ImageNet, but other networks can be used as well). This representation is important to avoid overfitting risks in small-size-labeled RS data sets. Using this representation as input instead of raw-pixel data, we train our hashing network using three different losses: a triplet loss which is in charge of learning a metric space and a balancing and a representation loss which effectively learn to produce binary hash codes. In this way, we jointly solve two problems: we avoid overfitting risks using the intermediate features and we learn how to produce similarity-based hash codes. Our empirical analysis shows the advantage of this combined strategy, which is more accurate and time-efficient than state-of-the-art methods.

REFERENCES

[1] T. D. G. Shakhnarovich and P. Indyk, *Nearest-Neighbor Methods in Learning and Vision: Theory and Practice*. Cambridge, MA, USA: MIT Press, 2006.
[2] M. Slaney and M. Casey, "Locality-sensitive hashing for finding nearest neighbors [lecture notes]," *IEEE Signal Process. Mag.*, vol. 25, no. 2, pp. 128–131, Mar. 2008.
[3] 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, Feb. 2016.
[4] 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. 2019.
[5] 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. 2018.
[6] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna, "Rethinking the inception architecture for computer vision," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2016, pp. 2818–2826.
[7] S. Roy, E. Sangineto, B. Demir, and N. Sebe, "Deep metric and hash-code learning for content-based retrieval of remote sensing images," in *Proc. IEEE Int. Geosci. Remote Sens. Symp. (IGARSS)*, Jul. 2018, pp. 4539–4542.
[8] F. Schroff, D. Kalenichenko, and J. Philbin, "FaceNet: A unified embedding for face recognition and clustering," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2015, pp. 815–823.
[9] H.-F. Yang, K. Lin, and C.-S. Chen, "Supervised learning of semantics-preserving hash via deep convolutional neural networks," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 40, no. 2, pp. 437–451, Feb. 2018.
[10] O. E. Dai, B. Demir, B. Sankur, and L. Bruzzone, "A novel system for content-based retrieval of single and multi-label high-dimensional remote sensing images," *IEEE J. Sel. Topics Appl. Earth Observ. Remote Sens.*, vol. 11, no. 7, pp. 2473–2490, Jul. 2018.
[11] 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.
[12] W.-J. Li, S. Wang, and W.-C. Kang, "Feature learning based deep supervised hashing with pairwise labels," 2015, *arXiv:1511.03855*. [Online]. Available: http://arxiv.org/abs/1511.03855