Scene Classification via Triplet Networks

Yishu Liu and Chao Huang

***Abstract*—Scene classification is a fundamental task for automatic remote sensing image understanding. In recent years, convolutional neural networks have become a hot research topic in the remote sensing community, and have made great achievements in scene classification. Deep convolutional networks are primarily trained in a supervised way, requiring huge volumes of labeled training samples. However, clearly labeled remote sensing data are usually limited. To address this issue, in this paper, we propose a novel scene classification method via triplet networks, which use weakly labeled images as network inputs. Besides, we initiate a theoretical study on the three existing loss functions for triplet networks, analyzing their different underlying mechanisms for dealing with "hard" and/or "easy" triplets during training. Furthermore, four new loss functions are constructed, aiming at laying more stress on "hard" triplets to improve classification accuracy. Extensive experiments have been conducted, and the experimental results show that triplet networks coupled with our proposed losses achieve a state-of-the-art performance in scene classification tasks.**

***Index Terms*—Deep learning, difference loss, ratio loss, scene classification, triplet networks.**

I. Introduction

SCENE classification that aims to label a remote sensing image according to a set of semantic categories is a fundamental task for automatic remote sensing image understanding, and finds many applications in land resource management, urban planning, computer cartography, and so on [1], [2]. At the same time, scene classification is a challenging problem thanks to the highly complex geometrical structures and spatial patterns inherent in remote sensing images. Some identical land-cover types or object classes are frequently shared among different scene categories, increasing both intraclass variability and interclass similarity; and objects always appear at different scales and orientations. The task becomes incrementally more difficult and urgent, with the rapid progress in remote sensing technology and the dramatical reduction of acquisition cost, which make a huge number of high-resolution remote sensing images available nowadays [3], [4]. Generally speaking, the key to the success of any model for scene classification is to find a good representation of images, that is, what features we use is crucial.

Manuscript received May 1, 2017; revised July 12, 2017 and August 21, 2017; accepted September 29, 2017. Date of publication October 26, 2017; date of current version January 12, 2018. This work was supported in part by NSFC under Grant 61673184 and 11501377. *(Corresponding author: Yishu Liu.)*

Y. Liu is with the School of Geography, South China Normal University, Guangzhou 510631, China (e-mail: yishuliu_gz@hotmail.com).

C. Huang is with the College of Mathematics and Statistics, Shenzhen University, Shenzhen 518060, China (e-mail: hchao@szu.edu.cn).

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

Digital Object Identifier 10.1109/JSTARS.2017.2761800

A. State of the Art and Motivation

Different from pixel/object-level image classification that interprets images in a bottom-up manner, scene classification is apt to directly model a scene image by developing its holistic representations [5]–[16]. In terms of the degree of abstraction, features used for scene classification can broadly be divided into three levels [5]: low level, midlevel, and high level (i.e., deep features).

1) Low- and Midlevel Features: Low-level features are extracted from such visual attributes as texture, structure, and spectral information, etc. The well-known scale invariant feature transform (SIFT) [17] belongs to this group. Other widely used techniques include Gist [18], color histogram [19], local binary pattern (LBP) [20], texture [21], and so on. Besides, in order to further improve classification accuracy, some works [16], [22] incorporated several kinds of low-level features to form a multiple-feature representation.

Generally speaking, low-level features perform well on the images with uniform structures and spatial arrangements, but it is difficult for them to characterize the scenes with high spatial variations and complex geometrical structures.

Midlevel features are always generated from low-level ones. A general pipeline is to first extract low-level features, e.g., SIFT, LBP, and color histograms of local image patches, and then, encode them to create a holistic midlevel representation for a scene image. One of the most popular midlevel approaches is the bag-of-visual-words (BoVW) model [9], which uses SIFT as local descriptors. To improve the discriminative power, many researchers have proposed numerous variants of BoVW. For instance, some [23], [24] integrated various low-level descriptors, besides SIFT, into the BoVW framework; some proposed different encoding schemes such as the improved fisher vector (IFK) [25] and vectors of locally aggregated tensors (VLAT) [26]; some added the spatial distribution of visual words to the BoVW model, employing typical approaches such as spatial pyramid match kernel (SPM) [6], spatial cooccurrence kernel [9], spatial pyramid cooccurrence kernel (SPCK) [27], and the pyramid-of-spatial-relation (PSR) model [11]; and others took the semantic relationship among the visual words into account [8], [28], using latent Dirichlet allocation or the probabilistic latent semantic analysis as mathematical models.

Furthermore, unsupervised approaches, such as sparse coding [11], spectral clustering [12], and saliency-guided techniques [10], have also been investigated to produce midlevel features for scene images.

2) High-Level Features (Deep Features): The aforementioned low-level features and midlevel ones are both hand crafted, lacking the flexibility in discovering highly intricate structures in high-dimensional remote sensing data. This is in direct contrast with deep learning [29], the crucial characteristic of which is that features are not designed by human engineers, instead, they are directly learned from data using a general-purpose learning procedure [30]. Deep learning techniques can adaptively discover multiple levels of abstractions from the images and combine them from low level to high level to form a hierarchical representation, with higher level features representing more abstract aspects of the data. In this paper, we call deep features *high-level features*.

In recent years, deep learning has become a dominant research topic in numerous fields and frequently achieved a better performance than traditional low- and midlevel features. As a typical and the most widely used deep learning model, convolutional neural network (CNN) [29], [31] has made great achievements in the remote sensing community. In this paper, we focus on CNN, other deep learning models and their applications in remote sensing image processing/understanding can be found in [32].

The scene classification methods based on high-level CNN features mainly fall into following four groups.

1) Directly using the features extracted from the pretrained CNNs: The CNNs pretrained on natural images have been successfully utilized for remote sensing scene classification. For instance, Penatti *et al.* [33] employed the two popular architectures, OverFeat and CaffeNet, to extract global features; Castelluccio *et al.* [34] exploited and evaluated two deep CNNs, i.e., CaffeNet and GoogLeNet. They all reported impressive performance on the publicly available UC-Merced (UCM) dataset [9]. Moreover, recently two new challenging aerial image datasets, AID [5] and NWPU-RESISC45 (N-R) [35], have been created, and pretrained CNNs showed significant advantages over low- and midlevel features on these two databases.
2) Reprocessing the features extracted from the pretrained CNNs: Instead of using the pretrained CNN as an off-the-shelf and final feature extractor, some researchers reprocessed the deep features. Hu *et al.* [36] extracted multiscale dense CNN activations from the last convolutional layer, and subsequently, coded them by means of feature encoding methods like BoVW [9], vector of locally aggregated descriptors [37], and IFK [25] to generate the final image representation. Marmanis *et al.* [38] proposed a two-stage framework: An initial set of representations were extracted from the pretrained OverFeat, and then, the derived representations along with their class labels were transferred into a supervised CNN classifier, effectively training the system.
3) Fine tuning the pretrained CNNs: Castelluccio *et al.* [34] fine tuned the pretrained CaffeNet and GoogLeNet on the remote sensing scene datasets, achieving a promising classification performance. Besides, the study [35] fine tuned three pretrained CNNs using aerial images, and the experimental results demonstrated that fine tuning can obtain a much higher accuracy than directly extracting features from fully connected layers.
4) Training CNNs from scratch: Some researchers trained CNNs from scratch using remote sensing datasets. In [34], the trained-from-scratch networks achieved a better performance on the Brazilian Coffee Scenes dataset than both the "GoogLeNet + fine-tuning" scheme and the "using GoogLeNet-features directly" strategy. However, Nogueira *et al.* [39] reported that training CNNs from scratch showed a drop in accuracy compared with using the off-the-shelf networks directly, which maybe was due to the limited training data. Moreover, using multiple partial views of the given samples as inputs, Luus *et al.* [40] built a CNN for land-use classification. Besides, Zhang *et al.* [41] trained a new network called gradient boosting random convolutional network.

3) Motivation: The aforementioned methods 1–3 have two shortcomings. First, the CNN architectures used by them are bulky and fixed, and cannot be tailored to specific tasks. Second, the final feature vectors are very high dimensional—usually the researchers use the outputs of the fully connected layers, the length of which is 4096 in the popular architectures such as OverFeat, CaffeNet, OxfordNet, AlexNet, and PlacesNet, and 1024 in GoogLeNet; in the case of reprocessing CNN activations, the resulting feature vector is even much longer, for example, the length in [36] is more than 50K. Undoubtedly, a concise representation is more desirable for the subsequent classification.

On the contrary, the fourth method builds relatively shallow and small networks since remote sensing data with labels are usually scarce. However, the generalization ability of a small and shallow network is often lower than that of large networks. In fact, scarcity of labeled data in remote sensing community is so prevalent that it presents a dilemma when we train our own CNNs from scratch. As far as we know, SAT-4&SAT-6 [42] is the only remote sensing scene dataset that has sufficient labeled images to train deep neural networks from the ground up; however, there are only four scene classes in the SAT-4 subset and six in SAT-6, and the image size is as small as 28×28, so this database is not suitable for verifying complicated multiclass scene classification methods.

Above all, the training/pretraining/fine-tuning process of these four methods are purely supervised learning, which requires that all data be clearly labeled.

To address these issues, we propose using weakly supervised triplet networks for remote sensing scene classification. A triplet network can be trained from scratch employing weakly labeled data, to some degree relieving the pressure of labeling a great number of images. And in a way, it makes a tradeoff among the availability of labeled data, the flexibility of CNN architecture and the size of a network. Most important of all, triplet networks coupled with our proposed losses achieve a state-of-the-art performance in scene classification.

B. Related Work

1) Triplet Networks in the Computer Vision Community: Besides single-branch CNNs [29], [31], Siamese Networks [43], [44] with two branches and triplet networks [45]–[49] with three branches have also been studied in the computer vision community. Wang *et al.* [45] proposed a deep ranking model using

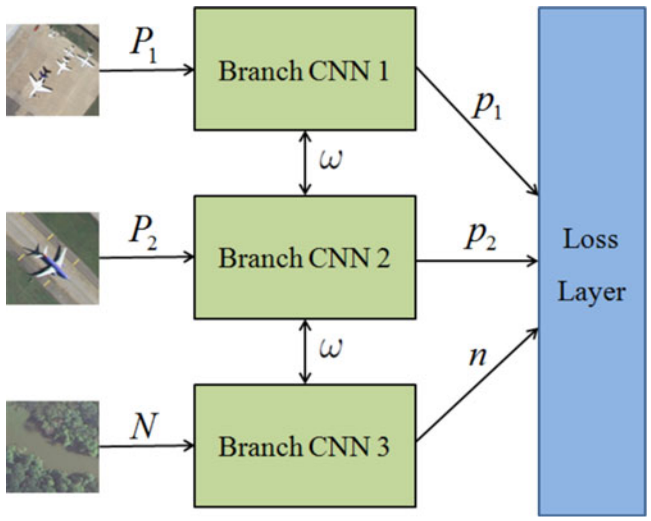

Fig. 1. Architecture of a triplet network. A triplet network has three branches sharing the same weights ω. Two positive images (P_1 and P_2) and one negative image (N) are input to three branches, with the corresponding outputs being p_1, p_2, and n, respectively. We require that $\|p_1 - p_2\|_2$ take a small value and $\min(\|p_1 - n\|_2, \|p_2 - n\|_2)$ take a large value by reasonably designing the loss function.

triplet networks to learn fine-grained image similarity; Ding *et al.* [46] applied triplet networks to person reidentification; Hoffer *et al.* [47], [48] and Balntas [49] conducted patch matching via triplet networks. They all reported a performance better than or comparable to that of deep single-branch CNNs.

A triplet network has three branch CNNs that share exactly the same architecture and the same set of weights, as shown in Fig. 1. Three images are simultaneously fed to a triplet network, each for one branch. Two of the images, P_1 and P_2 (P means *positive*), are from the same class; and the third one, N (N means *negative*), is from a different class. Hereafter, such a set of three images is called *triplet* for simplicity.

Regard each branch CNN as a vector-valued function $\mathcal{O}$, which maps an image to a real vector, then

$$\mathcal{O}(P_i) = p_i, \qquad i = 1, 2 \tag{I.1}$$

$$\mathcal{O}(N) = n \tag{I.2}$$

where p_1, p_2, and n are subsequently used to compute the loss. We refer to $\|p_1 - p_2\|_2$ as *positive distance*, and $\|p_1 - n\|_2$ and $\|p_2 - n\|_2$ as *negative distances*. Unlike the traditional deep learning methods aiming to minimize the classification error, triplet features are learned in such a way that the disparity between positive and negative distances will be as big as possible for each triplet, simultaneously forcing positive distances to take small values and negative distances to take large values. In fact, this is the principle underlying how a loss function is constructed.

Training a triplet network requires only knowing whether two images are from the same class, rather than knowing what the classes are (thus, it can be regarded as a weakly supervised learning method). And determining whether two images are similar is much easier than determining what class(es) they belong to—the former only needs weakly labeled data, while the latter demands definite labels. Therefore, it is easier to collect training samples for triplet networks than for traditional deep CNNs. Moreover, the triplet-based training imposes more constraints on distance learning, thereby helping to alleviate the overfitting problem. This is why we choose triplet networks as our deep feature learning model.

2) Existing Loss Functions for Triplet Networks: As far as we know, there have mainly been three types of loss functions for triplet networks.

Let the network parameters

$$\omega = \{\omega_j | j = 1, 2, \ldots, J\} \tag{I.3}$$

where ω_j denotes the parameters of the *j*th layer of any branch CNN (remember that three branches share exactly the same parameters), which have been reshaped into a column vector, and J denotes the number of total layers of each branch. Note that here ω involves weights and biases.

For a given training triplet $\{P_1, P_2, N\}$, the loss in [45] is defined as

$$\begin{aligned}&\mathcal{L}_1(\omega; P_1, P_2, N)\\ &= \max(0, T + \|\mathcal{O}(P_1) - \mathcal{O}(P_2)\|_2 - \|\mathcal{O}(P_1) - \mathcal{O}(N)\|_2)\\ &= \max(0, T + \|p_1 - p_2\|_2 - \|p_1 - n\|_2)\end{aligned} \tag{I.4}$$

where $T \geq 0$ is a margin threshold. We hope that $\|p_1 - n\|_2 > T + \|p_1 - p_2\|_2$, if that is not the case, the network will adjust its weights to achieve this result; if the marginal distance difference is satisfied, the loss is 0, and thus, the weights will not be updated.

In [47], the loss is defined as

$$\begin{aligned}\mathcal{L}_2(\omega; P_1, P_2, N) = &\left(\frac{e^{\|p_1 - p_2\|_2}}{e^{\|p_1 - p_2\|_2} + e^{\|p_1 - n\|_2}}\right)^2\\ &+ \left(1 - \frac{e^{\|p_1 - n\|_2}}{e^{\|p_1 - p_2\|_2} + e^{\|p_1 - n\|_2}}\right)^2.\end{aligned} \tag{I.5}$$

In addition, there is another loss in [48]

$$\mathcal{L}_3(\omega; P_1, P_2, N) = -\ln \frac{e^{-\|p_1 - p_2\|_2}}{e^{-\|p_1 - p_2\|_2} + e^{-\|p_1 - n\|_2}}. \tag{I.6}$$

There is no margin coupled with (I.5) or (I.6).

All these losses ignore the negative distance $\|p_2 - n\|_2$, taking only $\|p_1 - n\|_2$ into account. However, Balntas [49] used (I.4) and (I.5) for patch matching, and found that replacing $\|p_1 - n\|_2$ with $\min(\|p_1 - n\|_2, \|p_2 - n\|_2)$ leads to improved results, because $\min(\|p_1 - n\|_2, \|p_2 - n\|_2)$ can mine "hard" pairs inside the triplets, ensuring they are used for back propagation. We also adopt this scheme in this paper.

Let

$$d_+ = \|p_1 - p_2\|_2 \tag{I.7}$$

and

$$d_- = \min(\|p_1 - n\|_2, \|p_2 - n\|_2) \tag{I.8}$$

then (I.4)–(I.6) can be converted into the following forms:

$$\text{Loss-1}: \ \mathcal{L}_1(\omega; P_1, P_2, N) = \max(0, T + d_+ - d_-) \qquad T \geq 0 \tag{I.9}$$

$$\text{Loss-2}: \ \mathcal{L}_2(\omega; P_1, P_2, N) = \left(\frac{e^{d_+}}{e^{d_+} + e^{d_-}}\right)^2 + \left(1 - \frac{e^{d_-}}{e^{d_+} + e^{d_-}}\right)^2 \tag{I.10}$$

$$\text{Loss-3}: \ \mathcal{L}_3(\omega; P_1, P_2, N) = -\ln \frac{e^{-d_+}}{e^{-d_+} + e^{-d_-}}. \tag{I.11}$$

3) Our New Idea: Emphasizing "Hard" Triplets via Loss Functions: Some of the training triplets are "easy," that is, for them the disparity between positive and negative distances has been significant enough. Naturally we hope that less attention will be paid to "easy" triplets and "hard" triplets can contribute more to the average gradient in the optimization process.

To address this issue, techniques like hard mining [44] that is frequently coupled with Siamese networks and a contrastive loss [43] can be used. A Siamese network have two identical branches sharing the same weights. Suppose that the image pair input to a Siamese network is $\{Q_1, Q_2\}$, and

$$\mathcal{O}(Q_i) = q_i, \qquad i = 1, 2 \tag{I.12}$$

then the contrastive loss is defined as

$$\mathcal{L}_{\text{contr}}(\omega; Q_1, Q_2, \iota) = \begin{cases} \|q_1 - q_2\|_2, & \text{if } \iota = 1 \\ \max(0, T - \|q_1 - q_2\|_2), & \text{if } \iota = -1 \end{cases} \tag{I.13}$$

where ι can only take on two values, 1 and -1, with 1 indicating Q_1 and Q_2 belong to the same class and -1 indicating different classes; and $T \geq 0$ is a threshold. Intuitively the contrastive loss penalizes "1" pairs that have large distance and "-1" pairs that have small distance.

Hard mining strategy [44] identifies "hard" pairs via their distances, and a subset of these examples are refed to the network for parameter update in each iteration. Nevertheless, while this process leads to more discriminative convolutional features, it also comes at a very high computational cost, since in each epoch, a subset of the training data need to be backpropagated again through the network. Specifically, the best-performing architecture from [44] required 67% of the computational cost to be spent in mining hard pairs [49]. So Balntas [49] did not use this strategy to mine "hard" triplets; instead, he swapped two positive images if necessary, enabling the backpropagation to concentrate on "harder" pairs within the triplets. This approach does not need extra convolutional overhead and has been used by us as mentioned before.

However, we can go further, and let "hard" training examples play a dominant role in updating parameters via appropriate loss functions. We find that loss functions themselves indicate how much emphasis will be laid on "hard" triplets, and this has never been noticed by other researchers.

In this paper, we make a theoretical analysis of the existing loss functions, providing insight about how they behave differently in treating "hard/easy" triplets. Besides, we propose several new loss functions, which outperform or can compete with the existing ones.

C. Contributions

Overall, the contributions of this paper are listed as follows:

1) an initial attempt to apply triplet networks to remote sensing scene classification is made;
2) a theoretical analysis of the existing loss functions for triplet networks are conducted from the viewpoint of how they emphasize "hard" triplets;
3) several novel loss functions that outperform or can compete with the existing ones are proposed;
4) extensive experiments on scene classification are conducted on three publicly available datasets, i.e., SAT-4&SAT-6, UCM, and N-R, and state-of-the-art results are achieved.

D. Organization

The rest of this paper is organized as follows: in Section II, a theoretical analysis of the existing loss functions that are based on distance differences is made; in Section III, we propose several new loss functions based on distance ratios and investigate how they deal with "hard" and/or "easy" training triplets; Section IV presents details of our experiments and results; and finally, Section V summarizes this paper.

II. Difference Losses

Losses functions 1–3, i.e., (I.9)–(I.11), are all based on the difference between positive and negative distances, as can be seen later, so we call them *difference losses*. In this section, an analysis of these three losses is performed first, and then, an improved difference loss is proposed.

A. Theoretical Analysis of the Existing Losses

For any $x \in \mathbb{R}$, let

$$f(x) = \frac{1}{1 + e^{-x}}. \tag{II.14}$$

In fact, $f(x)$ is the widely used sigmoid function. Obviously, its derivative function

$$f'(x) = f(x)[1 - f(x)]. \tag{II.15}$$

And let

$$\delta = d_+ - d_-. \tag{II.16}$$

Note that, to simplify the notation, we use δ instead of $\delta(\omega; P1, P2, N)$; the same applies to d_+ and d_-.

Obviously

$$\text{Loss-1}: \ \mathcal{L}_1(\omega; P_1, P_2, N) = \max(0, \delta + T) \tag{II.17}$$

$$\triangleq \ell_1(\delta). \tag{II.18}$$

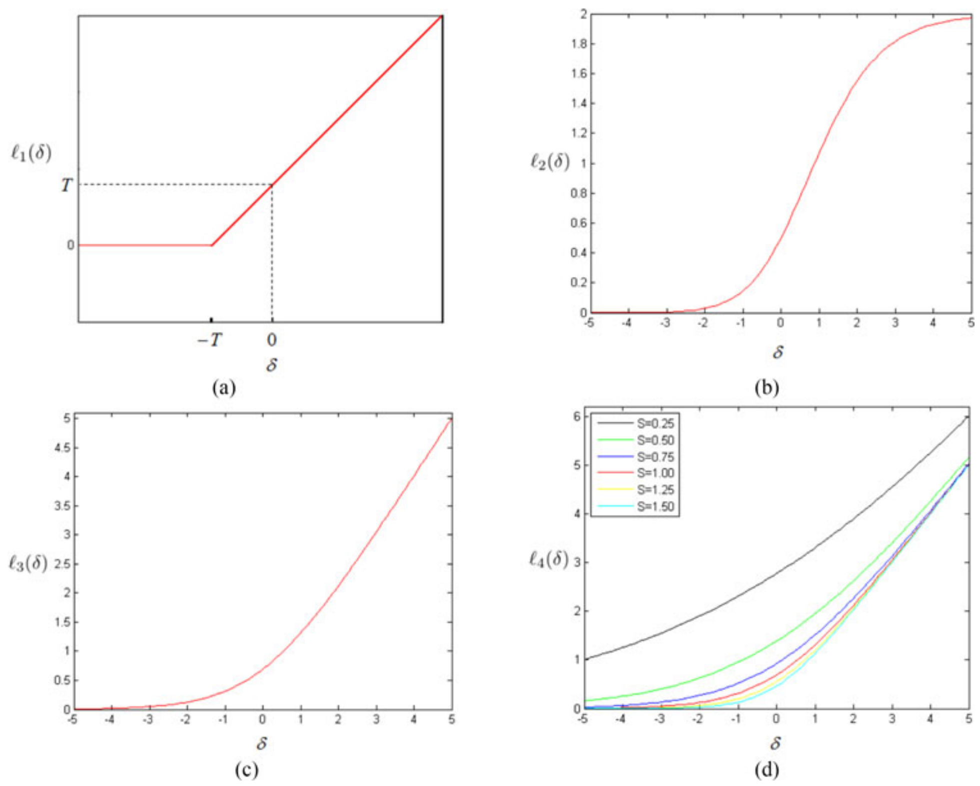

Fig. 2. Graphs of difference loss functions. These four functions are monotonically increasing, indicating that "harder" triplets contribute more to the loss because a large δ corresponds to a "harder" triplet. (a) Loss-1. (b) Loss-2. (c) Loss-3. (d) Loss-4.

Noting (II.14), we can simplify (I.10) to

$$\text{Loss-2}: \ \mathcal{L}_2(\omega; P_1, P_2, N) = 2f^2(\delta) \tag{II.19}$$

$$\triangleq \ell_2(\delta) \tag{II.20}$$

and (I.11) to

$$\text{Loss-3}: \ \mathcal{L}_3(\omega; P_1, P_2, N) = \ln(1 + e^{\delta}) \tag{II.21}$$

$$\triangleq \ell_3(\delta). \tag{II.22}$$

Regarding these losses as the functions of δ, we can draw their graphs [see Fig. 2(a)–(c)].

Bearing in mind that the greater value δ takes, the "harder" the corresponding triplet is, now we analyze how these three loss functions treat "hard" training triplets as follows.

Differentiating $\ell_1(\delta)$ leads to

$$\ell_1{}'(\delta) = \begin{cases} 0, & \delta < -T \\ \frac{1}{2}, & \delta = -T \\ 1, & \delta > -T. \end{cases} \tag{II.23}$$

Differentiating $\ell_2(\delta)$ and $\ell_3(\delta)$ with respect to δ, and noting (II.15), we have

$$\ell_2{}'(\delta) = 4f^2(\delta)[1 - f(\delta)] \tag{II.24}$$

and

$$\ell_3{}'(\delta) = f(\delta). \tag{II.25}$$

The graphs of $\ell_1{}'(\delta)$, $\ell_2{}'(\delta)$, and $\ell_3{}'(\delta)$ are shown in Fig. 3(a)–(c).

Applying the chain rule, we can compute the partial derivative of $\mathcal{L}_i(\omega; P_1, P_2, N)$ with respect to ω_j as follows:

$$\frac{\partial \mathcal{L}_i}{\partial \omega_j} = \ell_i{}'(\delta)\frac{\partial \delta}{\partial \omega_j}, \quad i \in \{1, 2, 3\}. \tag{II.26}$$

So far, no specific triplet has been involved, so we drop "$(\omega; P_1, P_2, N)$" and use "δ" concisely. When stochastic gradient descent (SGD) [50] works as the optimization algorithm, a mini-batch of triplets are involved in calculating the average gradient during backpropagation. In this case, we use "$\delta^{(m)}$" instead to indicate that it is coupled with the mth triplet. The same applies to "$\mathcal{L}_i^{(m)}$." Note that both $\delta^{(m)}$ and $\mathcal{L}_i^{(m)}$ are functions of ω. Correspondingly, (II.26) can be rewritten as

$$\frac{\partial \mathcal{L}_i^{(m)}}{\partial \omega_j} = \ell_i{}'(\delta^{(m)})\frac{\partial \delta^{(m)}}{\partial \omega_j}, \quad i \in \{1, 2, 3\}. \tag{II.27}$$

Then, one iteration of gradient descent updates the parameters as follows:

$$\omega_j \leftarrow \omega_j - \alpha\left(\frac{1}{M}\sum_{m=1}^{M} \ell_i{}'(\delta^{(m)})\frac{\partial \delta^{(m)}}{\partial \omega_j}\right), \quad i \in \{1, 2, 3\} \tag{II.28}$$

where M is the number of triplets in each minibatch, α denotes the learning rate, and $\frac{1}{M}\sum_{m=1}^{M} \ell_i{}'(\delta^{(m)})\frac{\partial \delta^{(m)}}{\partial \omega_j}$, i.e., $\frac{1}{M}\sum_{m=1}^{M} \frac{\partial \mathcal{L}_i^{(m)}}{\partial \omega_j}$, is the average gradient for a minibatch of triplets. One thing to note is that, for the sake of simplicity,

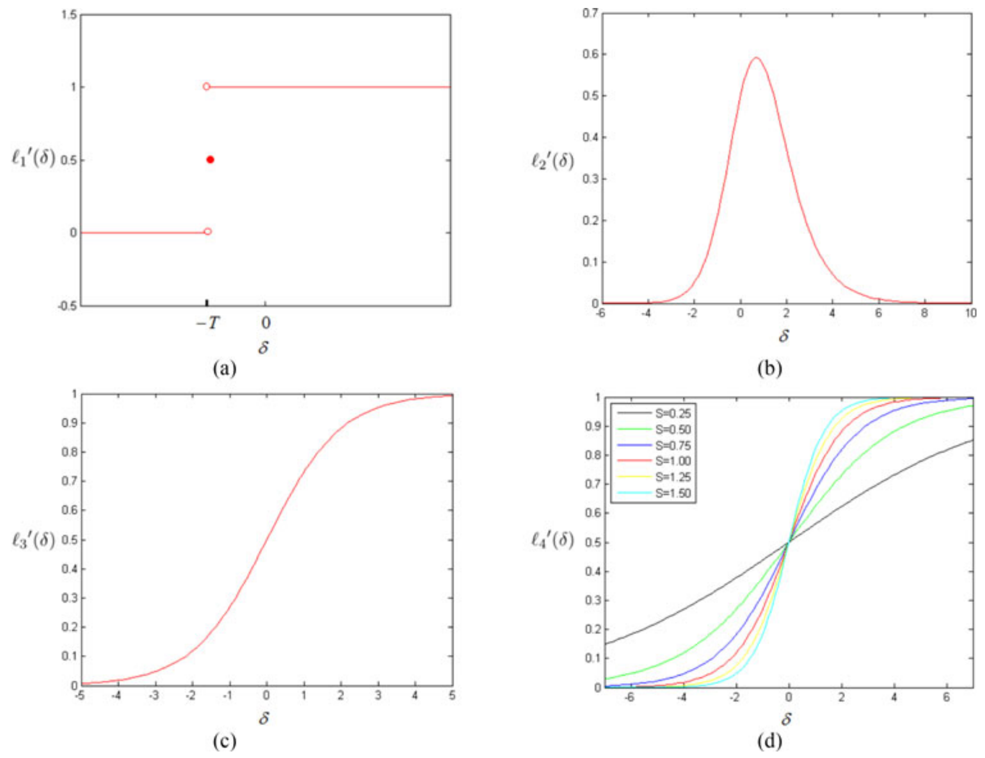

Fig. 3. Derivatives of difference losses. These graphs indicate how "hard" and/or "easy" triplets are dealt with in the backpropagation. The first, third, and fourth derivative functions are monotonically increasing, indicating that "harder" triplets contribute more to the average gradient. However, the second derivative function is monotonically decreasing when $\delta \in [\ln 2, +\infty)$, indicating that the "harder" a triplet is, the less attention is paid to it. This is not reasonable. (a) Loss-1. (b) Loss-2. (c) Loss-3. (d) Loss-4.

here we drop the regularization term (i.e., the weight decay term).

Obviously, the triplets that have greater gradient $\ell_i'(\delta^{(m)})\frac{\partial \delta^{(m)}}{\partial \omega_j}$ contribute more to the average gradient, and thus, have a more significant effect on parameter updates, making the network parameters satisfy their needs preferentially.

For any $i \in \{1, 2, 3\}$, $\frac{\partial \delta}{\partial \omega_j}$ (or $\frac{\partial \delta^{(m)}}{\partial \omega_j}$) is computed in exactly the same way. The difference lies in $\ell_i'(\delta)$ (or $\ell_i'(\delta^{(m)})$), that is to say, the formulas (II.23)–(II.25) indicate the different ways in which "hard" and/or "easy" triplets are dealt with.

Bearing in mind that the greater value δ takes, the "harder" the corresponding triplet is, now we analyze how these three loss functions treat "hard" training triplets.

1) From Fig. 3(a), it can be seen that Loss-1 ignores the triplets that have a δ smaller than $-T$, that is to say, the "easy" triplets do not participate in updating the parameters. On the other hand, the "hard" triplets, i.e., those that have a δ greater than $-T$, are equally dealt with.
2) From (II.24), we know that $\ell_2'(\delta)$ reaches its maximum at $\delta = \ln 2$. As shown in Fig. 3(b), $\ell_2'(\delta) \to 0$ when $\delta \to +\infty$ or $\delta \to -\infty$; this means that "extremely easy" and "extremely hard" triplets are both discarded. In particular, when $\delta \in [\ln 2, +\infty)$, $\ell_2'(\delta)$ is monotonically decreasing, indicating that the "harder" a triplet is, the less attention is paid to it. Obviously this is not reasonable, and in this regard, Loss-2 is not so good as Loss-1, as has been empirically proved in [49], where Loss-1 outperformed Loss-2 on the whole. And it has also been validated by us (see Section IV).
3) In contrast, $\ell_3'(\delta)$ is monotonically increasing in $(-\infty, +\infty)$, therefore, Loss-3 lays more emphasis upon the "harder" triplets. This seems quite justifiable.

B. New Difference Loss

Here, we propose a novel loss function

$$\text{Loss-4}: \quad \mathcal{L}_4(\omega; P_1, P_2, N) = \frac{1}{S}\ln(1 + e^{S\delta}), \quad S > 0 \tag{II.29}$$

$$\triangleq \ell_4(\delta). \tag{II.30}$$

Compared with Loss-3, Loss-4 has a parameter S that is an adjustable positive real number and whose optimal value can be determined through empirical study (see Section IV-B). This loss function's graph is drawn in Fig. 2(d).

It is easy to show that

$$\ell_4'(\delta) = f(S\delta). \tag{II.31}$$

Fig. 3(d) shows the graph of $\ell_4'(\delta)$, from which we can see that S is used to control the steepness of the curve, a greater S corresponding to a sharper change in the value of derivative function. In this way, we endow Loss-4 with more flexibility and adaptability, making it more resilient to different datasets.

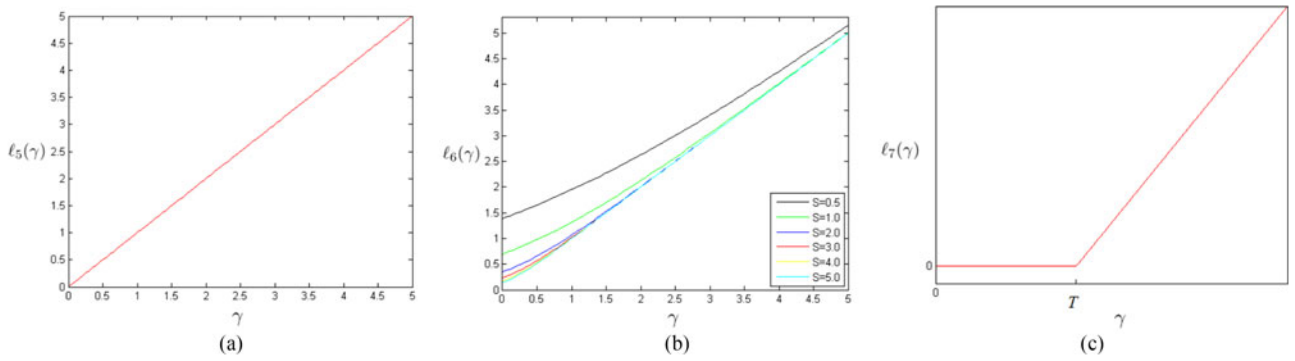

Fig. 4. Graphs of ratio loss functions. These three functions are monotonically increasing, indicating that "harder" triplets contribute more to the loss because a large γ corresponds to a "harder" triplet. (a) Loss-5. (b) Loss-6. (c) Loss-7.

III. Ratio Losses

The difference losses are all based on δ, the difference between a positive distance d_+ and a negative one d_-. Alternatively, the ratio of d_+ to d_- can also be used to construct a loss. We call this kind of loss functions *ratio losses*.

In order to facilitate the partial derivative calculation, we use the square of ratio, instead. Hereafter, ratio always means squared ratio.

Denoting by γ the ratio, i.e.,

$$\gamma = \left(\frac{d_+}{d_-}\right)^2 \tag{III.32}$$

we now propose the following three ratio losses

$$\begin{aligned}
\text{Loss-5 :}\quad & \mathcal{L}_5(\omega; P_1, P_2, N) \\
& = \gamma && \text{(III.33)} \\
& \triangleq \ell_5(\gamma) && \text{(III.34)} \\
\text{Loss-6 :}\quad & \mathcal{L}_6(\omega; P_1, P_2, N) \\
& = \frac{1}{S}\ln(1 + e^{S\gamma}), \quad S > 0 && \text{(III.35)} \\
& \triangleq \ell_6(\gamma) && \text{(III.36)} \\
\text{Loss-7 :}\quad & \mathcal{L}_7(\omega; P_1, P_2, N) \\
& = \max(0, \gamma - T), \quad T \geq 0 && \text{(III.37)} \\
& \triangleq \ell_7(\gamma). && \text{(III.38)}
\end{aligned}$$

Fig. 4 demonstrates the graphs of these three functions.

It is easy to show that

$$\ell_5{}'(\gamma) = 1 \tag{III.39}$$

$$\ell_6{}'(\gamma) = f(S\gamma) \tag{III.40}$$

and

$$\ell_7{}'(\gamma) = \begin{cases} 0, & \gamma < T \\ \frac{1}{2}, & \gamma = T \\ 1, & \gamma > T. \end{cases} \tag{III.41}$$

Fig. 5 displays their graphs.

The ratio loss functions' partial derivatives are given in a similar way to (II.26) and (II.27) as follows:

$$\frac{\partial \mathcal{L}_i}{\partial \omega_j} = \ell_i{}'(\gamma)\frac{\partial \gamma}{\partial \omega_j}, \quad i \in \{5, 6, 7\} \tag{III.42}$$

$$\frac{\partial \mathcal{L}_i^{(m)}}{\partial \omega_j} = \ell_i{}'(\gamma^{(m)})\frac{\partial \gamma^{(m)}}{\partial \omega_j}, \quad i \in \{5, 6, 7\}. \tag{III.43}$$

And the parameter updating rule is also given in a like manner as follows:

$$\omega_j \leftarrow \omega_j - \alpha\left(\frac{1}{M}\sum_{m=1}^{M} \ell_i{}'(\gamma^{(m)})\frac{\partial \gamma^{(m)}}{\partial \omega_j}\right), \quad i \in \{5, 6, 7\}. \tag{III.44}$$

Therefore, $\ell_i{}'(\gamma)$ plays the same role as $\ell_i{}'(\delta)$ in determining how much importance is attached to "hard" triplets.

The aforementioned loss functions, including difference losses and ratio losses, are all increasing functions of δ or γ. However, minimizing them is not equivalent to minimizing δ or γ because each loss function has its own mechanism to deal with "hard" and/or "easy" training data, as has been explained previously.

IV. Numerical Experiments

In this section, we first present an experimental setup, including an introduction to the databases, training parameter settings and branch-CNN architectures, etc., and then, detailed experimental results are demonstrated, discussed, and analyzed.

A. *Experimental Setup*

1) Datasets: SAT-4&SAT-6: This dataset [42] includes two subsets: SAT-4 and SAT-6. SAT-4 consists of 500 000 image patches, including four classes: barren land, tree, grassland, and a class that is composed of all other land-cover types [see Fig. 6(a)]. SAT-6 is a database of 405 000 image patches covering six classes: building, barren land, tree, grassland, road, and water [see Fig. 6(b)]. These images have four bands—red, green, blue, and near infrared, have a resolution of 1 m per pixel, and are of 28×28 pixels. Both SAT-4 and SAT-6 provide sufficient data for training deep networks from scratch.

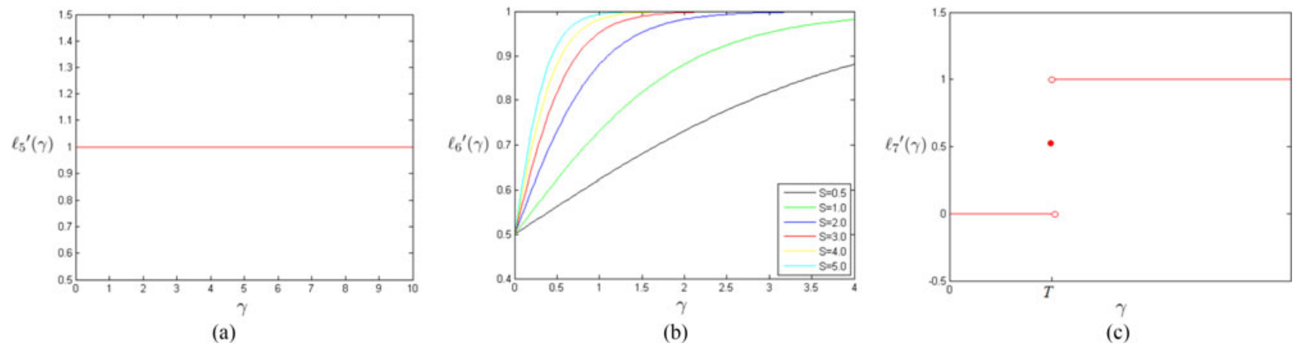

Fig. 5. Derivatives of ratio losses. These graphs indicate how "hard" and/or "easy" triplets are dealt with in the backpropagation. $\ell_5{}'(\gamma)$ is a constant function, indicating that "harder" and "easy" triplets contribute equally to the average gradient; $\ell_6{}'(\gamma)$ is a strictly increasing function, indicating that "harder" triplets contribute more to the average gradient; and $\ell_7{}'(\gamma)$ is a step function, indicating that "easy" triplets do not participate in updating the network weights and "hard" triplets contribute equally to the average gradient. (a) Loss-5. (b) Loss-6. (c) Loss-7.

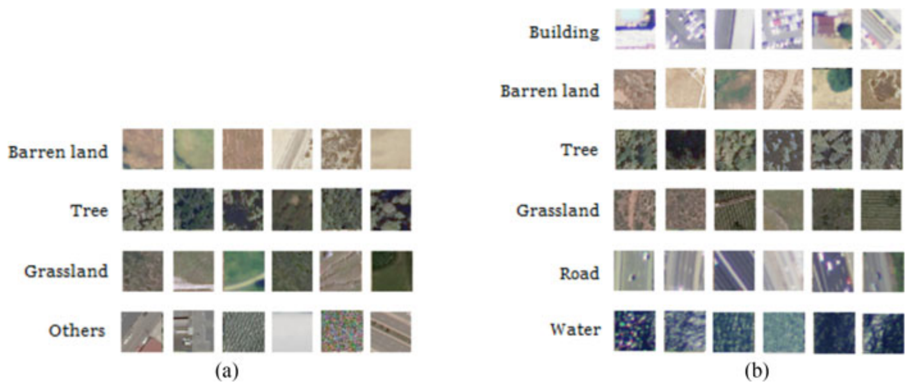

Fig. 6. Example images from the SAT-4&SAT-6 dataset. SAT-4 has four scene categories, and SAT-6 has six ones. (a) SAT-4. (b) SAT-6.

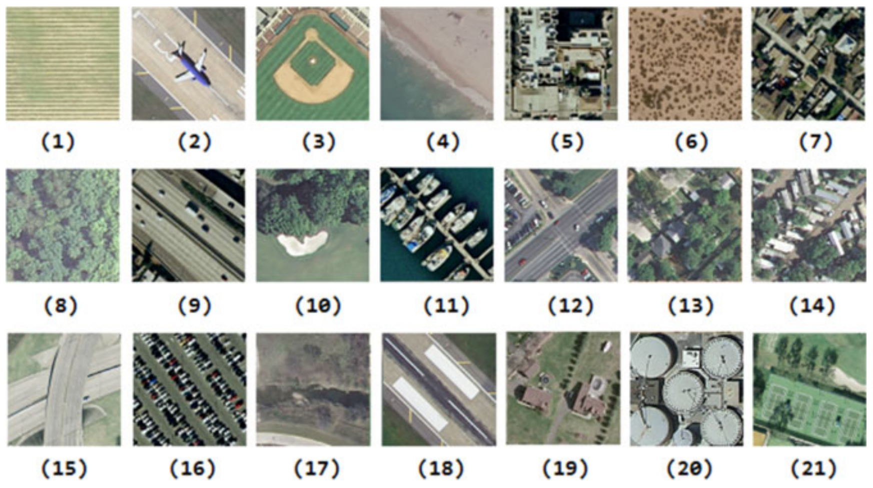

Fig. 7. Example images from the UCM dataset: (1) Agricultural; (2) Airplane; (3) Base ball diamond; (4) Beach; (5) Building; (6) Chaparral; (7) Dense residential; (8) Forest; (9) Freeway; (10) Golf course; (11) Harbor; (12) Intersection; (13) Medium residential; (14) Mobile home park; (15) Overpass; (16) Parking lot; (17) River; (18) Runway; (19) Sparse residential; (20) Storage tank; (21) Tennis court.

UCM: In the UCM dataset [9] there are 21 scene categories, each containing 100 images. The images are with the size of 256×256 pixels and 1-ft spatial resolution. Fig. 7 shows some example images from UCM, with one sample per class. Since this database contains only a small quantity of images, we apply data augmentation to it.

N-R: N-R [35] is a newly released dataset that has 45 challenging scene classes and 31 500 images, with 700 images per class (see Fig. 8 for some example images). The image size is 256×256, and the spatial resolution varies from about 30 to 0.2 m per pixel. The images in each scene category possess very large variations in translation, viewpoint, object pose, object

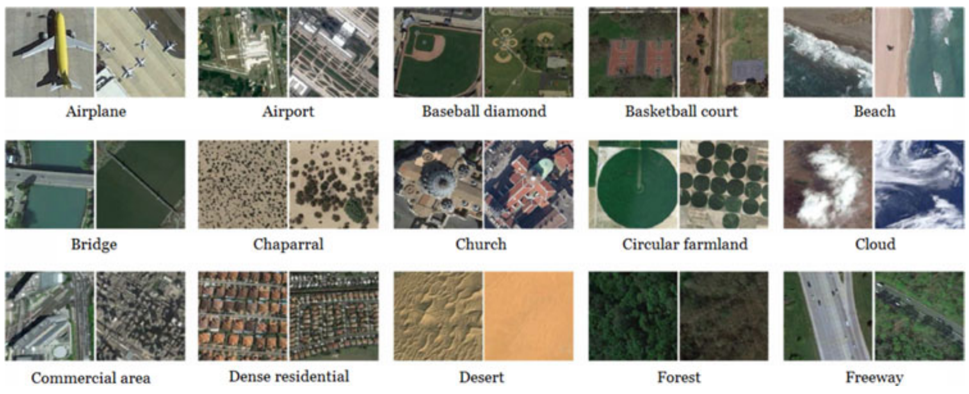

Fig. 8. Example images from the N-R dataset (figures taken from [35]). N-R has 45 scene categories, only 15 of which are shown here.

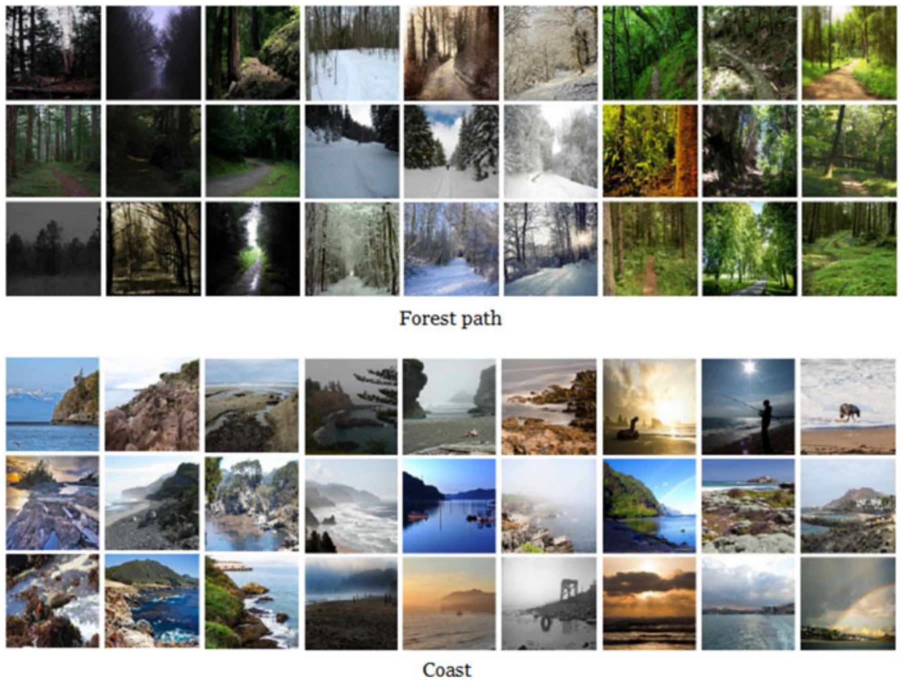

Fig. 9. Example images from the Places-205 dataset (figures taken from [52]). Places-205 is scene centric and has 205 classes, only two of which are shown here.

appearance, spatial resolution, illumination, background, and occlusion, etc. And some of the classes have very high semantic overlapping. Because Neither UCM nor N-R is big enough to be used to train deep neural networks from scratch, we adopt the scheme of "pretraining over natural image datasets + fine tuning over UCM/N-R."

Natural image datasets SubImageNet and SubPlaces: We use the subsets of the natural image datasets ImageNet [51] and Places-205 [52] to pretrain our triplet networks, followed by fine tuning over UCM/N-R. ImageNet is a 1000-class object-centric database, many well-known CNN models such as AlexNet [31] and OxfordNet [53] were trained over it. We randomly select 410 classes from ImageNet, with randomly selected 1000 images per class. In this way, the subset SubImageNet is generated. Since what we focus on is scene classification, pretraining over Places-205, a scene-centric dataset containing 205 classes, is also investigated. We randomly select 2000 images per class to create the subset SubPlaces. Only a few examples of Places-205 are shown in Fig. 9 due to limited space.

2) Data Augmentation: Recent works [36] show that in both training and testing stages, data augmentation, which is carried out by sampling and flipping multiple subimage windows, is an effective way to enhance classification performance. Since SAT-4, SAT-6, SubImageNet and SubPlaces all include a great many images, we only expand the UCM and N-R datasets. We adopt the same augmentation strategy as that in [40]: four subimage windows are acquired at the image corners at 75% input coverage, and another five are obtained at the corners and center at 50% input coverage. These nine subimages as well as the original image are flipped horizontally or vertically with a probability of 0.5. It is worth noting that the final feature vector for each test image is constructed by averaging the activation vectors over the 20 (sub-)image windows.

TABLE I
DETAILS OF DATASET DIVISION

Dataset	# of Classes	# of Samples per Class	# of Total Samples	# of Samples for (Pre-)Training or Fine Tuning	# of Samples for Validation	# of Samples for Testing
SAT-4	4	Inequal	500 K	350 K	50 K	100 K
SAT-6	6	Inequal	405 K	283.5 K	40.5 K	81 K
SubImageNet	410	1000	410 K	369 K	41 K	-
SubPlaces	205	2000	410 K	369 K	41 K	-
UCM	21	100 (O)[1] 2000 (A)[2]	2.1 K(O) 42 K(A)	29.4 K (A)	4.2 K (A)	0.42 K(O)[3] 8.4 K(A)[3]
N-R	45	700 (O)[1] 14000 (A)[2]	31.5 K(O) 630 K(A)	126 K (A)	63 K (A)	22.05 K(O)[3] 441 K(A)[3]

[1] *O* means "original."
[2] *A* means "augmented."
[3]The final feature vector for each test image is constructed by averaging the activation vectors of the last layer of the branch CNN over the 20 (sub-)image windows.

TABLE II
DETAILS OF TRIPLETS

Dataset	# of Triplets for (Pre-)Training or Fine Tuning	# of Triplets for Validation
SAT-4	2.0 M	0.1 M
SAT-6	2.0 M	0.1 M
SubImageNet	2.0 M	0.1 M
SubPlaces	2.0 M	0.1 M
UCM	0.3 M	0.04 M
N-R	1.0 M	0.1 M

3) Dataset Division: The SAT-4, SAT-6, and UCM datasets are all randomly divided into subsets for training/fine tuning (70%), validation (10%), and testing (20%). And over N-R, the ratios are 20%, 10%, and 70%, respectively. Furthermore, both SubImageNet and SubPlaces are randomly split into subsets for pretraining (90%) and validation (10%). Table I presents the details of data partition. In the (pre-)training and fine-fining stages, we use the validation sets to observe how the network performs, and then, to decide whether the learning rate needs to be changed. And in the testing stage, we use them for training the classifier.

The numbers of the triplets generated from each dataset's training set and validation set are shown in Table II. And each triplet is formed in this way: A pair of images from the same class ($P1$ and $P2$) are randomly chosen, and subsequently, another image, N, is randomly chosen from among the other classes.

In order to obtain reliable results for independent test data, we conduct ten repetitions of each experiment, each repetition having different dataset division. And the final performance is reported as the mean of the overall accuracies over the ten rounds.

4) Branch CNN Architecture: We construct two kinds of branch CNN architecture, one for SAT-4 and SAT-6, and the other for UCM and N-R. Different branch architectures are used because of different image sizes (28×28 for SAT-4 and SAT-6 and 256×256 for UCM and N-R), different numbers of input channels (four for SAT-4 and SAT-6, and three for UCM and N-R), and different levels of classification difficulty (SAT-4 and SAT-6 are much less challenging than UCM and N-R, thus shallower and smaller networks are enough to guarantee a high classification performance).

The two kinds of branch architecture are illustrated in Fig. 10, in which the parameter configuration in convolutional layers is denoted as "(receptive field size): The number of feature maps." Obviously, all the (sub-)images in UCM, N-R, SubImageNet, and SubPlaces should be resized to $192 \times 192 \times 3$ beforehand. And it should be stressed that batch normalization, following convolution, is always performed to accelerate and regularize the learning.

5) Parameters of SGD: We (pre-)train/fine-tune the triplet networks using SGD [50], with a batch size of 128, a momentum of 0.9, and a weight decay of 0.0005. As for the learning rate, initially we use 0.01 for SAT-4/SAT-6, 0.1 for SubImageNet/SubPlaces, and 0.005 for UCM/N-R; afterwards, so long as the average value of the objective function over the validation set stops decreasing, we divide the learning rate by 5.

6) Hardware and Software Environment: All the experiments are run on an AMAX workstation with two Intel® Xeon® E5-2640Wv4 CPUs with ten cores, two NVIDIA Titan X GPUs, and a 128-GB memory. As for development environment, we use MatConvNet [54] that is based on MATLAB.

7) Classifiers: After training/fine tuning has been finished, any two of the three branch CNNs are removed. An image's feature vector can be obtained via a forward pass through the retained branch, and then, it is fed to the classifier.

In order to fully investigate the potential of triplet features and exclude the influences that different classifiers have on classification, we use three classifiers, i.e., *K*-nearest neighbors (KNN), back-propagation neural networks (BP-NN) [55], and support vector machine (SVM) [56], and compare their results.

Note that at this stage labels are needed. However, a small number of labeled samples will suffice, compared to the enormous quantity of clearly labeled images needed for training traditional deep CNNs. As mentioned previously, we use the validation sets to train the classifiers because their sizes are appropriate (certainly, any other set that is disjoint from the test sets can be used instead).

Taking SVM as an example, here we explain the classifier training/testing procedure. We employ the LIBSVM implementation [56], and as suggested in [57], Radial Basis Function

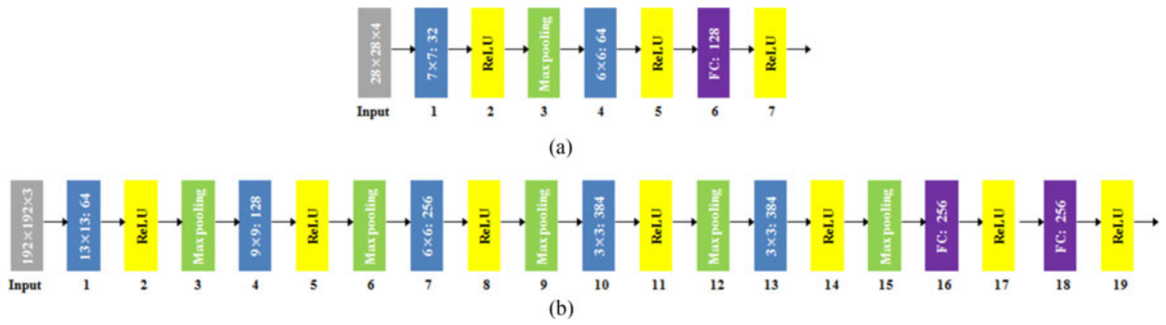

Fig. 10. Branch CNN Architecture. The parameter configuration in convolutional layers (blue) is denoted as “(receptive field size): the number of feature maps,” *FC* (purple) means “fully connected layer,” and max pooling (green) is of 2 × 2. (a) For SAT-4 and SAT-6. (b) For UCM and N-R.

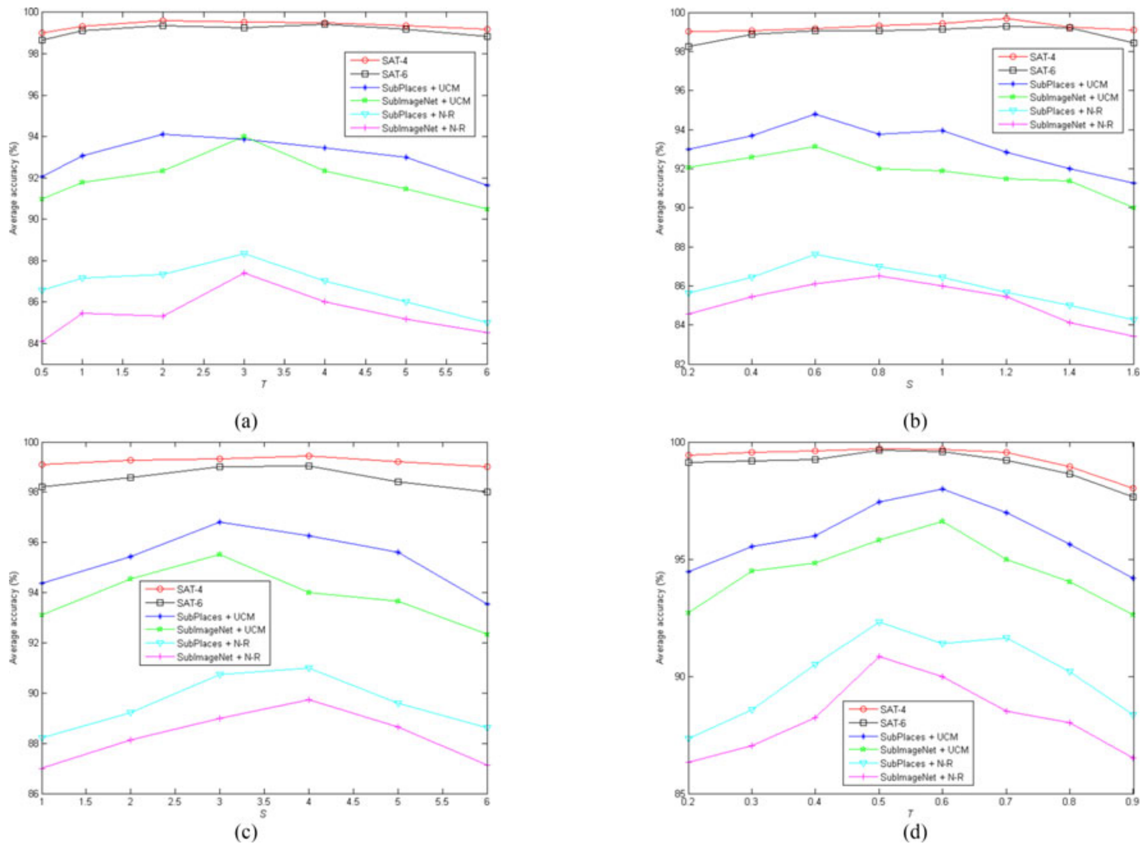

Fig. 11. Values of parameters versus Average accuracy. The parameters’ optimal values vary with the datasets, so to use dataset-specific parameters will result in a better performance. (a) Loss-1. (b) Loss-4. (c) Loss-6. (d) Loss-7.

(RBF) is used as the kernel function. The training/testing steps are as follows:

1) transform data to the format of LIBSVM;
2) conduct simple scaling on the data;
3) randomly divide the validation set into five subsets, and use 5-fold cross-validation and grid-search to seek the optimal values of two classifier parameters (the classifier parameters include one penalty parameter and one kernel parameter);
4) use the optimal classifier parameters to train an SVM over the whole validation set;
5) use the trained classifier to classify the test set.

A more detailed explanation about each step can be found in [57].

Both K in KNN and the number of hidden layer neurons in BP-NN (we use only one hidden layer) are also found via 5-fold cross-validation, they vary with the dataset and loss function.

B. Determining the Optimal Parameters of Loss Functions

We investigate how the parameters coupled with Loss-1, Loss-4, Loss-6, and Loss-7 affect the classification performance. Ten rounds of experiments are carried out over each remote sensing dataset, and the mean of the overall accuracies is shown in Fig. 11 (for the sake of space, only the results produced by the SVM are given). It can be seen that the optimal values of parameters vary with the dataset. In particular, Fig. 11(a) shows that

TABLE III
ACCURACY COMPARISON AMONG LOSSES

Dataset	Classifier	Loss Function						
		Loss-1	Loss-2	Loss-3	Loss-4	Loss-5	Loss-6	Loss-7
SAT-4	KNN	99.11 ± 0.05	98.50 ± 0.06	99.03 ± 0.06	99.19 ± 0.05	98.89 ± 0.04	98.95 ± 0.05	**99.22 ± 0.04**
	SVM	99.60 ± 0.05	99.03 ± 0.06	99.52 ± 0.05	99.70 ± 0.05	99.33 ± 0.05	99.44 ± 0.04	**99.72 ± 0.04**
	BP-NN	**99.83 ± 0.04**	99.12 ± 0.06	99.52 ± 0.05	99.80 ± 0.06	99.30 ± 0.05	99.38 ± 0.06	**99.83 ± 0.05**
SAT-6	KNN	98.88 ± 0.07	97.89 ± 0.06	98.50 ± 0.07	98.63 ± 0.07	98.00 ± 0.06	98.48 ± 0.06	**99.18 ± 0.07**
	SVM	99.43 ± 0.07	98.97 ± 0.08	99.12 ± 0.06	99.30 ± 0.07	99.02 ± 0.07	99.05 ± 0.07	**99.65 ± 0.06**
	BP-NN	99.36 ± 0.08	98.88 ± 0.08	99.22 ± 0.09	99.31 ± 0.08	99.00 ± 0.06	99.00 ± 0.08	**99.69 ± 0.06**
SubImageNet + UCM	KNN	92.89 ± 0.58	90.49 ± 0.65	91.25 ± 0.58	92.08 ± 0.75	93.57 ± 0.60	94.58 ± 0.57	**95.42 ± 0.58**
	SVM	94.00 ± 0.51	91.58 ± 0.59	92.11 ± 0.60	93.12 ± 0.72	94.78 ± 0.63	95.52 ± 0.60	**96.62 ± 0.59**
	BP-NN	95.12 ± 0.63	91.50 ± 0.55	92.00 ± 0.65	93.89 ± 0.68	94.65 ± 0.70	96.00 ± 0.60	**96.68 ± 0.63**
SubPlaces + UCM	KNN	93.15 ± 0.57	91.45 ± 0.70	92.44 ± 0.58	93.56 ± 0.57	94.68 ± 0.67	95.19 ± 0.66	**96.48 ± 0.59**
	SVM	94.10 ± 0.58	92.57 ± 0.61	93.40 ± 0.52	94.78 ± 0.60	95.64 ± 0.71	96.82 ± 0.56	**97.99 ± 0.53**
	BP-NN	95.14 ± 0.63	93.00 ± 0.65	93.77 ± 0.60	94.65 ± 0.63	96.00 ± 0.58	96.75 ± 0.62	**97.99 ± 0.60**
SubImageNet + N-R	KNN	86.46 ± 0.23	83.68 ± 0.19	85.01 ± 0.21	85.44 ± 0.19	87.39 ± 0.20	88.55 ± 0.19	**89.64 ± 0.20**
	SVM	87.41 ± 0.22	84.79 ± 0.18	86.11 ± 0.19	86.50 ± 0.23	88.48 ± 0.15	89.73 ± 0.18	**90.85 ± 0.17**
	BP-NN	87.00 ± 0.21	85.00 ± 0.20	86.77 ± 0.20	87.02 ± 0.20	88.40 ± 0.18	90.12 ± 0.21	**91.25 ± 0.18**
SubPlaces + N-R	KNN	87.24 ± 0.19	84.91 ± 0.20	85.25 ± 0.22	86.71 ± 0.19	88.04 ± 0.19	89.58 ± 0.21	**91.16 ± 0.20**
	SVM	88.33 ± 0.18	85.89 ± 0.22	86.34 ± 0.20	87.61 ± 0.16	89.24 ± 0.21	91.00 ± 0.19	**92.33 ± 0.20**
	BP-NN	88.53 ± 0.20	86.41 ± 0.20	87.12 ± 0.21	87.59 ± 0.21	89.88 ± 0.20	91.23 ± 0.21	**92.40 ± 0.19**

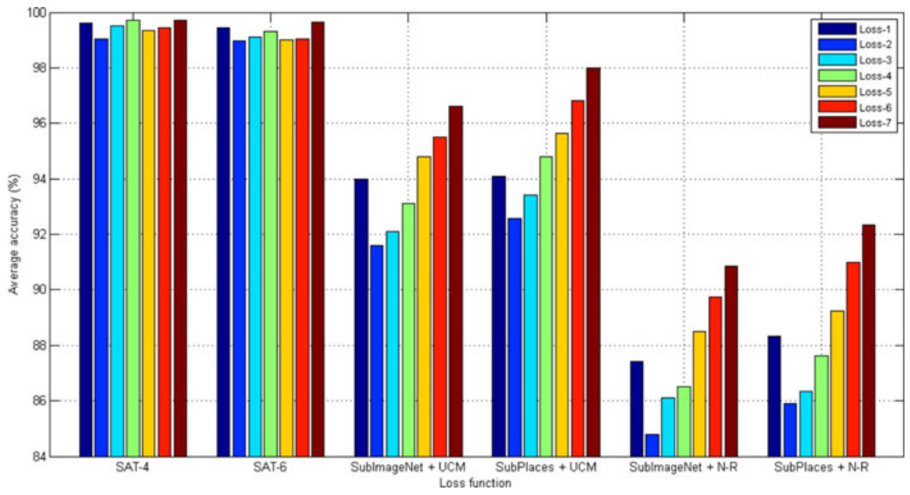

Fig. 12. Performance comparison among losses. Over all the datasets, Loss-7 performs best and Loss-2 performs worst.

the best values of T for SAT-4, SAT-6, "SubImageNet + UCM," "SubPlaces + UCM," "SubImageNet + N-R," and "SubPlaces + N-R" are 2, 4, 3, 2, 3, and 3, respectively. However, most of the studies [45], [46] that used Loss-1 as a loss function set T to 1 all the time, which is not very reasonable according to our experimental observations.

Hereinafter, the numerical results associated with these four losses invariably correspond to the optimal values of T or S.

C. Performance Comparison Among Losses

Triplet networks "equipped with" different loss functions perform differently. We summarize the means and standard deviations of overall accuracy in Table III and Fig. 12 (for the sake of space, only SVM's numerical results are illustrated in Fig. 12). Based on these results, our observations are listed as follows.

1) Loss-7 outperforms all the other losses over all the datasets, having the highest accuracies.
2) Loss-2 performs worst, with the lowest accuracy over all the datasets. This is not unexpected, because it does not attach enough importance to very "hard" triplets, as indicated by (II.24).
3) Over "SubImageNet + UCM," "SubPlaces + UCM," "SubImageNet + N-R," and "SubPlaces + N-R" difference losses are inferior to ratio losses; more specifically, the weakest ratio loss outperforms the strongest difference loss. These observations indicate that ratio losses are a better choice than difference losses over challenging datasets.
4) Over SAT-4 and SAT-6, difference losses and ratio losses have a similar performance. These two data subsets are less challenging, so even a "weak" loss function can

achieve a high accuracy over them. In other words, over an "easy" dataset, it does not make much difference to choose ratio losses or difference losses.

5) Of the three ratio losses, Loss-5 is the weakest, with the lowest accuracies over all the datasets, as can be explained by its lack of a mechanism for laying more stress on "hard" training triplets.
6) Of the four difference losses, Loss-3 is the second weakest, superior to Loss-2, but inferior to Loss-1 and Loss-4. This is mainly due to the fact that Loss-3 emphasizes "hard" triplets in a fixed way. In contrast, both Loss-1 and Loss-4 have adjustable parameters, and hence, possess more flexibility. As a matter of fact, Loss-3 is only a special case of Loss-4, so naturally it is weaker than Loss-4.
7) Regardless of which loss function is used, pretraining over SubPlaces always leads to a better performance than over SubImageNet. This is to be expected, since SubImageNet is object centric, whereas SubPlaces is scene centric, and hence, more appropriate for scene classification. However, this is quite different from what was claimed in [36]—the finding therein was that used as an off-the-shelf feature extractor, PlacesNet, a popular deep CNN model trained over Places-205, always performs worse than other models trained over ImageNet. A possible explanation for the different findings is that out of 1000 classes of ImageNet, only 410 classes are selected to pretrain our triplet networks, thus the decreased diversity of data results in insufficient expressive power of the networks.
8) The four remote sensing datasets can be ranked in ascending order of the standard deviation of overall accuracy as follows: SAT-4, SAT-6, N-R, and UCM, with the number of testing samples being exactly in the opposite order. Over UCM, there are only dozens of images per class for testing, therefore, overall accuracy will be quite different if a few more images are misclassified, leading to a greater numerical variation.
9) In general, BP-NN works best, slightly outperforming the SVM. Although KNN performs the worst, the difference in accuracy between it and BP-NN/SVM is small (less than 2.00% basically). KNN is a very simple classifier, so it cannot contribute much to a promising performance. All this indicates that our classification results are basically independent of classifiers and instead depend on the features learned by triplet networks to a large extent.

D. Training Time Comparison Among Losses

The average training (or pretraining + fine tuning) time for each dataset (pair) is given in Table IV.

Over SAT-4 and SAT-6, the average training time is less than one hour for any loss function. Actually, it takes only 15–30 epochs for the triplet networks to converge over these two data subsets. And surprisingly, even after the first epoch, i.e., after about 2 min of training over SAT-4 (SAT-6), the learned features associated with Loss-7 result in a classification accuracy as high as 99.3% (99.2%).

TABLE IV
AVERAGE TRAINING TIME

Dataset	Average Training Time (h)						
	Loss-1	Loss-2	Loss-3	Loss-4	Loss-5	Loss-6	Loss-7
SAT-4	0.64	0.68	0.67	0.66	0.70	0.65	**0.59**
SAT-6	**0.62**	0.68	0.70	0.69	0.66	0.67	0.64
SubImageNet + UCM	57.12	58.23	63.15	61.32	62.54	58.26	**55.68**
SubPlaces + UCM	**63.34**	70.33	69.56	73.48	68.92	70.71	66.38
SubImageNet + N-R	80.27	82.66	88.49	83.32	87.18	83.26	**77.33**
SubPlaces + N-R	81.78	85.21	86.62	84.19	88.35	82.46	**78.45**

Since the second triplet network is much bigger, training it needs about 2.5∼3.5 days.

In general, pretraining over SubPlaces takes more time than over SubImageNet, maybe the former involves higher variety and more diversity of data.

Finally, the most important finding is that of the seven loss functions, Loss-1 and Loss-7 always lead to the fastest convergence.

E. Confusion Analysis

We make a confusion analysis, and Fig. 13 presents the confusion matrices, where an entry in the *i*th row and *j*th column denotes the rate of test samples from the *i*th class that are misclassified as the *j*th class. Any confusion matrix here corresponds to the experiment whose overall accuracy is the median value of the ten experimental results.

Over SAT-4, the class with the highest accuracy is tree, and the relatively notable class confusion is between barren land and grassland. Over SAT-6, water has an accuracy of 100%, and confusion occurs mainly between road and building, barren land and grassland, tree and grassland. And over UCM, the category with the least accurate prediction is medium residential, followed by storage tanks. As for N-R, the major confusion exists between palace and church, railway and railway station, dense residential and medium residential, resulting from their similar global structure or spatial layout; and the classes with higher accuracy, such as chaparral, cloud, desert, sea ice, and snowberg, etc., always have a regular textural and spatial structure.

F. Comparison With State-of-the-Art Methods

In this section, our numerical results are compared with the state-of-the-art accuracy obtained over the same datasets. In order to reduce the impact of classifiers, and hence, guarantee a fairer comparison, we do not employ the classification results yielded by BP-NN that works best among the three classifiers. Instead, those of the SVM are used for comparison.

Since the widely used contrastive loss (I.13) for Siamese networks is exactly the counterpart of our proposed triplet losses, we also make a comparison between it and our losses. However, to the best of our knowledge, contrastive loss has never been validated over any of the four remote sensing datasets. So, we build Siamese networks coupled with the contrastive loss, conduct experiments under the same experimental conditions for

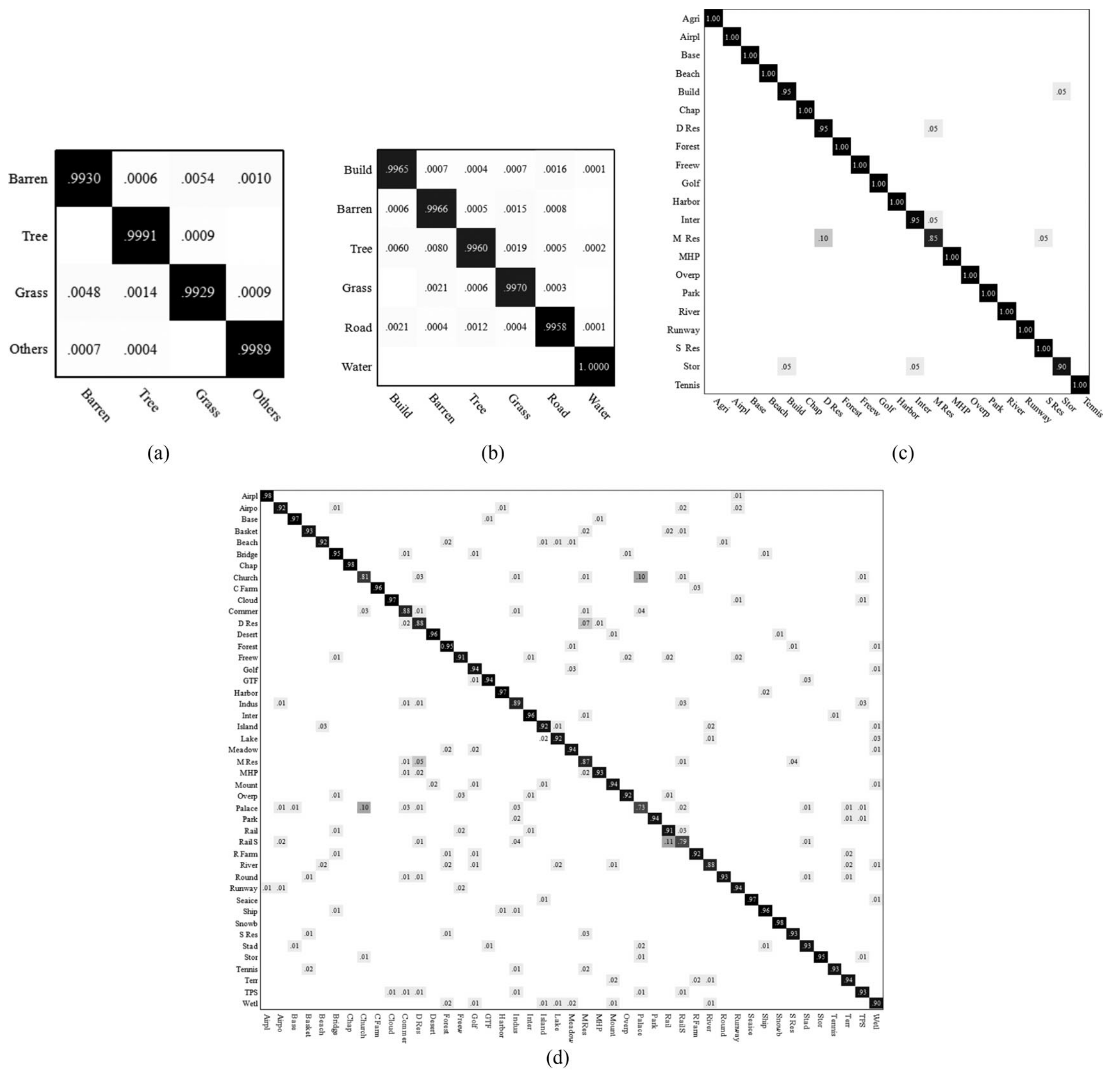

Fig. 13. Confusion Matrices. An entry in the *i*th row and *j*th column denotes the rate of test samples from the *i*th class that are misclassified as the *j*th class. (a) SAT-4. (b) SAT-6. (c) UCM. (d) N-R.

triplet networks as described in Section IV-A, and use the same hard mining strategy as that in [44].

SAT-4&SAT-6 is a new database, so far the publicly reported results about it have been mainly achieved via deep learning models. A numerical comparison is given in Table V, from which it can be seen that the triplet networks obtains the highest accuracies over both SAT-4 and SAT-6. Especially, our proposed weakly supervised learning method outperforms the completely supervised agile CNN [58] that was also trained from scratch.

For the sake of a comprehensive comparison, we summarize many methods validated on the UCM dataset in recent years, see Table VI. We can see that our proposed method outperforms all the traditional approaches based-on midlevel features, generally by a large margin. And it is also better than most of the approaches based on deep learning, with the exception of the "OverFeat + CaffeNet" [33] and "combining features from scenarios I and II" [36] schemes—our accuracy is lower than theirs by 1.44% and 0.50%, respectively. In spite of this, our method is superior to these two in the following respects:

TABLE V
PERFORMANCE COMPARISON AMONG STATE-OF-THE-ART METHODS OVER SAT-4&SAT-6

Method	Year	Overall Accuracy (%)	
		SAT-4	SAT-6
Stacked denoising autoencoder [42]	2015	79.98	78.43
Deep belief networks [42]	2015	81.78	76.47
CNN [42]	2015	86.83	79.1
DeepSat [42]	2015	97.95	93.92
Contrastive loss [44] (revalidated by us)	2015	98.67 ± 0.07	98.48 ± 0.07
Multilayer perceptron [58]	2017	94.76	97.46
Traditional CNN [58]	2017	98.43	98.34
Agile CNN (linear normalization) [58]	2017	99.43 ± 0.12	99.50 ± 0.08
Agile CNN (z-score normalization) [58]	2017	99.65 ± 0.04	99.54 ± 0.07
Triplet networks [this work]	2017	**99.72 ± 0.04**	**99.65 ± 0.06**

1) We only use "stand-alone" features, while the two studies [33], [36] utilized the combined features—one [33] combined features from two CNNs, i.e., OverFeat and CaffeNet, and the other [36] combined features from the

TABLE VI
PERFORMANCE COMPARISON AMONG STATE-OF-THE-ART METHODS OVER UCM

Method	Year	Overall Accuracy (%)
Midlevel Features		
SPM [6]	2006	74
SCK [9]	2010	72.52
SPCK [27]	2011	77.38
Dense SIFT [8]	2014	81.67+1.23
A concentric circle-structured BoVW model [13]	2014	86.64 ± 0.81
Dirichlet-based histogram feature transform [14]	2014	92.8 ± 0.9
Vectors of locally aggregated tensors [26]	2014	94.3
Saliency-guided unsupervised feature learning[10]	2015	82.72 ± 1.18
MinTree + KD-Tree [59]	2015	83.1 ± 1.2
PSR [11]	2015	89.1
Sparse PCA + Saliency computing [60]	2016	86.33 ± 0.61
MSIFT [16]	2016	90.97 + 1.81
Quaternion orthogonal matching pursuit [61]	2016	92.29 ± 0.71
Multicolumn SDAs + Fisher vector [62]	2016	92.7
Dirichlet-derived multiple topic model [63]	2016	92.92 ± 1.23
A local-global feature BoVW scene classifier [64]	2016	96.88 ± 1.32
A multipath sparse coding architecture [15]	2017	91.95 ± 0.72
Deep Features		
Using OverFeat [33]	2015	90.91 ± 1.19
Using CaffeNet [33]	2015	93.42 ± 1.00
Multiview deep learning [40]	2015	93.48 ± 0.82
Contrastive loss [44] (revalidated by us)	2015	95.93 ± 0.63
Scenario I: FC layers of CNN [36]	2015	96.88 ± 0.72
Scenario II: Convolutional layers of CNN [36]	2015	96.90 ± 0.77
Fine tuning GoogLeNet [34]	2015	97.10
Combining features from scenarios I and II [36]	2015	**98.49**
OverFeat + CaffeNet [33]	2015	**99.43**
Sparse representation + Deep belief network [65]	2016	81.92
OverFeat + Trainable CNN [38]	2016	92.4
Gradient Boosting Random CNN [41]	2016	94.53
Random-scale stretched CNN [66]	2016	95.10
Triplet networks [this paper]	2017	**97.99 ± 0.53**

fully connected layers of VGG-S and the convolutional layers of VGG-VD16. Since in many pattern recognition problems feature fusion usually leads to a much higher accuracy, we think that our triplet network features would perform better if combined with other features, but this is far from the scope of this work. Actually, it should be noted that all the "stand-alone" methods proposed in [33] and [36] obtained a worse result than ours (see the results of "using OverFeat," "using CaffeNet," "Scenario I: FC layers of CNN," and "Scenario II: Convolutional layers of CNN").

2) Our final feature vectors are 256-dimensional, providing a concise and contract representation; by comparison, the final feature vectors in [33], [36] are 8192-dimensional, making the subsequent classification tasks more difficult.
3) Our method learns features directly and no postprocessing steps are required, in contrast with the complicated and laborious process of encoding the extracted features in [36]; and we believe that our model will further improve the classification performance if it is coupled with effective and efficient feature encoding schemes.

TABLE VII
PERFORMANCE COMPARISON AMONG STATE-OF-THE-ART METHODS OVER N-R

Method	Year	Overall Accuracy (%)
Low-Level Features		
GIST [35]	2017	17.88 ± 0.22
LBP [35]	2017	21.74 ± 0.18
Color histograms [35]	2017	27.52 ± 0.14
Midlevel Features		
BoVW + SPM [35]	2017	32.96 ± 0.47
Locality constrained linear coding [35]	2017	40.03 ± 0.34
BoVW [35]	2017	44.97 ± 0.28
Deep Features		
Contrastive loss [44] (revalidated by us)	2015	87.15 ± 0.17
Using GoogLeNet [35]	2017	78.48 ± 0.26
Using VGGNet-16 [35]	2017	79.79 ± 0.15
Using AlexNet [35]	2017	79.85 ± 0.13
Fine-tuning AlexNet [35]	2017	85.16 ± 0.18
Fine-tuning GoogLeNet [35]	2017	86.02 ± 0.18
Fine-tuning VGGNet-16 [35]	2017	90.36 ± 0.18
Triplet networks [this paper]	2017	**92.33 ± 0.20**

Released in March 2017, N-R is a very fresh dataset, and the numerical results publicly available about it are only the performance baselines reported in [35]. These baseline data are summarized in Table VII, from which we can see that the triplet network features not only greatly outperform the low- and midlevel features, but also surpass all the deep features. Besides a higher accuracy, our proposed approach has the following advantages over the deep learning methods used in [35]:

1) Our model is tiny, while the study [35] utilized cumbersome and heavy AlexNet, VGGNet-16, and GoogLeNet—our triplet network has about 5M parameters, while VGGNet-16, for example, has about 138M parameters.
2) Our final feature vector (256-dimensional) is much shorter than those of AlexNet (4096-dimensional), VGGNet-16 (4096-dimensional), and GoogLeNet (1024-dimensional).
3) Our networks are trained from scratch, thus more task specific and more flexible; while the study [35] used fixed and pretrained CNNs.

In addition, from the aforementioned numerical results, it can be seen that in terms of classification performance, triplet networks coupled with our proposed losses are superior to Siamese networks coupled with a contrastive loss over all the four datasets. This further demonstrates that our method is feasible and effective.

Finally, it should be pointed out that a straightforward comparison between classification accuracy is not very objective due to different experimental conditions, though it is the most commonly used method. Strictly speaking, in order to reach statistically valid conclusions, we should have conducted statistical tests [67]. This will be our future work.

G. Discussion About Weakly Labeled Data

Although our experiments are carried out over clearly labeled datasets for the sake of fair comparison with other studies, labels are not used during (pre-)training or fine tuning. Actually what we need know is if two images are from the same class rather than what that class is, that is to say, we only require the information whether two images are similar. Since people generally have a better sense of what similarity is, the "weak labels" can be obtained in a crowdsourced manner [68], ameliorating the problem of the scarcity of clearly labeled remote sensing images.

Furthermore, our experimental results show that the scheme of "pretraining over natural image datasets + fine tuning over remote sensing image datasets" can also lead to a satisfactory performance. As a matter of fact, the "weak labels" of everyday images used for pretraining can be collected via volunteered geographic information. Nowadays an increasing number of users have uploaded geotagged photographs on social media websites, such as Flickr, Instagram, Picasa, and Panoramio. Owing to advancements in camera and mobile technology, social media photos contain a vast amount of information, ranging from nonspatial information, such as detailed semantic tags, titles, and descriptions, to spatial and temporal information, such as the locations and times in which the photos are taken [69]. This provides a relatively easy means to collect data trainable on triplet networks; for example, pictures taken at the same location or sharing similar semantic tags may belong to the same class.

In addition, Sun *et al.* [70] proposed transferring class labels from ground-level images to overhead, remote sensing images and demonstrated an unsupervised approach for domain adaption. We think the following scheme that combines triplet networks with class label transferring is worth investigating: to pretrain triplet networks using the weakly labeled ground view images collected from social media websites, and then, to transfer the class labels to remote sensing images using an unsupervised method similar to that of [70]. In this way, labels of remote sensing images are not needed at all (at the same time, everyday images can be easily weakly labeled), therefore, scarcity of labeled remote sensing data is no longer a problem. This will be our future research focus.

In a word, it is easier to collect (pre-)training data for triplet networks than for general CNNs, so our approach is more applicable to the scenarios with a limited number of clearly labeled training examples.

该方法更适合带标签的训练样本
比较少的情形

V. Summary

Deep CNNs are prevailingly trained in a supervised way, requiring a significant quantity of training samples, however, clearly labeled remote sensing data are not always available. To address this issue, in this paper, we proposed a novel scene classification method based on triplet networks, which can be trained via weakly labeled images—what we need to know is whether two images are similar or not, instead of what class(es) they belong to.

In addition, we make a theoretical analysis of the existing loss functions for triplet networks from the viewpoint of how different losses deal with "hard" training samples. And then, we propose some new losses, aiming to pay more attention to the "hard" triplets.

Finally, extensive experiments have been carried out over the publicly available datasets SAT-4&SAT-6, UCM, and N-R, and the experimental results show that the triplet networks coupled with our proposed losses achieve a state-of-the-art performance, better than or comparable to supervised deep learning approaches.

Our triplet networks are trained from scratch. And compared to the widely used bulky CNN models, ours are small and light, providing a new paradigm on how remote sensing data can be effectively and efficiently dealt with via deep learning.

However, there is an issue that remains to be resolved. We have empirically demonstrated that ratio losses perform better than difference losses over challenging datasets, but it lacks sufficient theoretical support. To give a mathematical explanation will be our future research focus.

未来工作重点是用理论来
证明 radio losses 比 difference losses
表现要好

References

[1] R. K. Jaiswal, R. Saxena, and S. Mukherjee, "Application of remote sensing technology for land use/land cover change analysis," *J. Indian Soc. Remote Sens.*, vol. 27, no. 2, pp. 123–128, Jun. 1999.

[2] J. Rogan and D. Chen, "Remote sensing technology for mapping and monitoring land-cover and land-use change," *Progress Planning*, vol. 61, no. 4, pp. 301–325, 2004.

[3] T. Lillesand, R. W. Kiefer, and J. Chipman, *Remote Sensing and Image Interpretation*, 7th ed. New York, NY, USA: Wiley, Feb. 2015.

[4] G. Camps-Valls, D. Tuia, L. Gómez-Chova, S. Jiménez, and J. Malo, Eds., *Remote Sensing Image Processing*. LaPorte, CO, USA: Morgan & Claypool Publishers, Dec. 2011.

[5] G.-S. Xia *et al.*, "AID: A benchmark dataset for performance evaluation of aerial scene classification," *CoRR*, vol. abs/1608.05167, Aug. 2016. [Online]. Available: http://arxiv.org/abs/1608.05167

[6] S. Lazebnik, C. Schmid, and J. Ponce, "Beyond bags of features: Spatial pyramid matching for recognizing natural scene categories," in *Proc. IEEE Conf. Comput. Vis. Pattern Recogn.*, New York, NY, USA, Jun. 2006, pp. 2169–2178.

[7] A. Romero, C. Gatta, and G. Camps-Valls, "Unsupervised deep feature extraction for remote sensing image classification," *IEEE Trans. Geosci. Remote Sens.*, vol. 54, no. 3, pp. 1349–1362, Mar. 2016.

[8] A. M. Cheriyadat, "Unsupervised feature learning for aerial scene classification," *IEEE Trans. Geosci. Remote Sens.*, vol. 52, no. 1, pp. 439–451, Jan. 2014.

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

[10] F. Zhang, B. Du, and L. Zhang, "Saliency-guided unsupervised feature learning for scene classification," *IEEE Trans. Geosci. Remote Sens.*, vol. 53, no. 4, pp. 2175–2184, Apr. 2015.

[11] S. Chen and Y. Tian, "Pyramid of spatial relatons for scene-level land use classification," *IEEE Trans. Geosci. Remote Sens.*, vol. 53, no. 4, pp. 1947–1957, Apr. 2015.

[12] F. Hu, G. Xia, Z. Wang, and H. Sun, "Unsupervised feature learning via spectral clustering of multidimensional patches for remotely sensed scene classification," *IEEE J. Sel. Topics Appl. Earth Observ. Remote Sens.*, vol. 8, no. 5, pp. 2015–2030, May 2015.

[13] L.-J. Zhao, P. Tang, and L.-Z. Huo, "Land-use scene classification using a concentric circle-structured multiscale bag-of-visual-words model," *IEEE J. Sel. Topics Appl. Earth Observ. Remote Sens.*, vol. 7, no. 12, pp. 4620–4631, Dec. 2014.

[14] T. Kobayashi, "Dirichlet-based histogram feature transform for image classification," in *Proc. IEEE Conf. Comput. Vis. Pattern Recog.*, Columbus, OH, USA, Jun. 2014, pp. 3278–3285.

[15] J. Fan, T. Chen, and S. Lu, "Unsupervised feature learning for land-use scene recognition," *IEEE Trans. Geosci. Remote Sens.*, vol. 55, no. 4, pp. 2250–2261, Apr. 2017.

[16] A. Avramović and V. Risojević, "Block-based semantic classification of high-resolution multispectral aerial images," *Signal, Image Video Process.*, vol. 10, no. 1, pp. 75–84, Jan. 2016.

[17] D. G. Lowe, "Distinctive image features from scale-invariant keypoints," *Int. J. Comput. Vis.*, vol. 60, no. 2, pp. 91–110, Nov. 2004.

[18] A. Oliva and A. Torralba, "Modeling the shape of the scene: A holistic representation of the spatial envelope," *Int. J. Comput. Vis.*, vol. 42, no. 3, pp. 145–175, May 2001.

[19] M. J. Swain and D. H. Ballard, "Color indexing," *Int. J. Comput. Vis.*, vol. 7, no. 1, pp. 11–32, Nov. 1991.

[20] T. Ojala, M. Pietikäinen, and T. Mäenpää, "Multiresolution gray-scale and rotation invariant texture classification with local binary patterns," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 24, no. 7, pp. 971–987, Jul. 2002.

[21] V. Risojević and Z. Babić, "Aerial image classification using structural texture similarity," in *Proc. 2011 IEEE Int. Symp. Signal Process. Inf. Technol.*, Bilbao, Spain, Dec. 2011, pp. 190–195.

[22] X. Chen, T. Fang, H. Huo, and D. Li, "Measuring the effectiveness of various features for thematic information extraction from very high resolution remote sensing imagery," *IEEE Trans. Geosci. Remote Sens.*, vol. 53, no. 9, pp. 4837–4851, Sep. 2015.

[23] L. Chen, W. Yang, K. Xu, and T. Xu, "Evaluation of local features for scene classification using VHR satellite images," in *Proc. 2011 Joint Urban Remote Sens. Event*, Munich, Germany, Apr. 2011, pp. 385–388.

[24] W. Shao, W. Yang, and G. Xia, "Extreme value theory-based calibration for the fusion of multiple features in high-resolution satellite scene classification," *Int. J. Remote Sens.*, vol. 34, no. 23, pp. 8588–8602, Dec. 2013.

[25] F. Perronnin, J. Sánchez, and T. Mensink, "Improving the fisher kernel for large-scale image classification," in *Proc. 11th Eur. Conf. Comput. Vis.: Part IV*, Heraklion, Greece, Sep. 2010, pp. 143–156.

[26] R. Negrel, D. Picard, and P.-H. Gosselin, "Evaluation of second-order visual features for land-use classification," in *Proc. Int. Workshop Content-Based Multimedia Indexing*, Klagenfurt, Austria, Jun. 2014, pp. 1–5.

[27] Y. Yang and S. D. Newsam, "Spatial pyramid co-occurrence for image classification," in *Proc. IEEE Int. Conf. Comput. Vis.*, Barcelona, Spain, Nov. 2011, pp. 1465–1472.

[28] Y. Zhong, Q. Zhu, and L. Zhang, "Scene classification based on the multi-feature fusion probabilistic topic model for high spatial resolution remote sensing imagery," *IEEE Trans. Geosci. Remote Sens.*, vol. 53, no. 11, pp. 6207–6222, Nov. 2015.

[29] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. Cambridge, MA, USA: MIT Press, 2016. [Online]. Available: http://www.deeplearningbook.org

[30] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *Nature*, vol. 521, no. 7553, pp. 436–444, Sep. 2015.

[31] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "ImageNet classification with deep convolutional neural networks," in *Advances in Neural Information Processing Systems*, New York, NY, USA: Curran Associates, Inc., 2012, pp. 1097–1105.

[32] L. Zhang, L. Zhang, and B. Du, "Deep learning for remote sensing data: A technical tutorial on the state of the art," *IEEE Geosci. Remote Sens. Mag.*, vol. 4, no. 2, pp. 22–40, Jun. 2016.

[33] O. A. B. Penatti, K. Nogueira, and J. A. dos Santos, "Do deep features generalize from everyday objects to remote sensing and aerial scenes domains?" in *Proc. IEEE Conf. Comput. Vis. Pattern Recog. Workshops*, Boston, MA, USA, Jun. 2015, pp. 44–51.

[34] M. Castelluccio, G. Poggi, C. Sansone, and L. Verdoliva, "Land use classification in remote sensing images by convolutional neural networks," *CoRR*, vol. abs/1508.00092, Aug. 2015. [Online]. Available: http://arxiv.org/abs/1508.00092

[35] G. Cheng, J. Han, and X. Lu, "Remote sensing image scene classification: Benchmark and state of the art," *CoRR*, vol. abs/1703.00121, Mar. 2017. [Online]. Available: https://arxiv.org/abs/1703.00121

[36] F. Hu, G. Xia, J. Hu, and L. Zhang, "Transferring deep convolutional neural networks for the scene classification of high-resolution remote sensing imagery," *Remote Sens.*, vol. 7, no. 11, pp. 14 680–14 707, Nov. 2015.

[37] H. Jégou, F. Perronnin, M. Douze, J. Sánchez, P. Pérez, and C. Schmid, "Aggregating local image descriptors into compact codes," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 34, no. 9, pp. 1704–1716, Sep. 2012.

[38] D. Marmanis, M. Datcu, T. Esch, and U. Stilla, "Deep learning earth observation classification using ImageNet pretrained networks," *IEEE Geosci. Remote Sens. Lett.*, vol. 13, no. 1, pp. 105–109, Jan. 2016.

[39] K. Nogueira, O. A. B. Penatti, and J. A. dos Santos, "Towards better exploiting convolutional neural networks for remote sensing scene classification," *Pattern Recogn.*, vol. 61, pp. 539–556, Jan. 2017.

[40] F. P. S. Luus, B. P. Salmon, F. van den Bergh, and B. T. J. Maharaj, "Multiview deep learning for land-use classification," *IEEE Geosci. Remote Sens. Lett.*, vol. 12, no. 12, pp. 2448–2452, Dec. 2015.

[41] F. Zhang, B. Du, and L. Zhang, "Scene classification via a gradient boosting random convolutional network framework," *IEEE Trans. Geosci. Remote Sens.*, vol. 54, no. 3, pp. 1793–1802, Mar. 2016.

[42] S. Basu, S. Ganguly, S. Mukhopadhyay, R. DiBiano, M. Karki, and R. Nemani, "DeepSat: A learning framework for satellite imagery," in *Proc. 23rd SIGSPATIAL Int. Conf. Adv. Geographic Inf. Syst.*, Seattle, WA, USA, Nov. 2015, pp. 37-1–37-10.

[43] S. Chopra, R. Hadsell, and Y. LeCun, "Learning a similarity metric discriminatively, with application to face verification," in *Proc. IEEE Conf. Comput. Vis. Pattern Recog.*, vol. 1, San Diego, CA, USA, Jun. 2005, pp. 539–546.

[44] E. Simo-Serra, E. Trulls, L. Ferraz, I. Kokkinos, P. Fua, and F. Moreno-Noguer, "Discriminative learning of deep convolutional feature point descriptors," in *Proc. IEEE Conf. Comput. Vis.*, Santiago, Chile, Dec. 2015, pp. 118–126.

[45] J. Wang *et al.* "Learning fine-grained image similarity with deep ranking," in *Proc. IEEE Conf. Comput. Vis. Pattern Recog.*, Columbus, OH, USA, Jun. 2014, pp. 1386–1393.

[46] S. Ding, L. Lin, G. Wang, and H. Chao, "Deep feature learning with relative distance comparison for person re-identification," *Pattern Recog.*, vol. 48, no. 10, pp. 2993–3003, Oct. 2015.

[47] E. Hoffer and N. Ailon, "Deep metric learning using triplet network," in *Similarity-Based Pattern Recognition.* New York, NY, USA: Springer, Nov. 2015, vol. 9370, pp. 84–92.

[48] E. Hoffer, I. Hubara, and N. Ailon, "Deep unsupervised learning through spatial contrasting," *CoRR*, vol. abs/1610.00243, Oct. 2016. [Online]. Available: http://arxiv.org/abs/1610.00243

[49] V. Balntas, "Efficient learning of local image descriptors," Ph.D. dissertation, Univ. of Surrey, Guildford, U.K., 2016.

[50] L. Bottou, "Stochastic gradient descent tricks," in *Neural Networks: Tricks of the Trade*, 2nd ed. New York, NY, USA: Springer, 2012, vol. 7700, pp. 421–436.

[51] 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 Recogn.*, Miami Beach, FL, USA, Jun. 2009, pp. 248–255.

[52] B. Zhou, A. Lapedriza, J. Xiao, A. Torralba, and A. Oliva, "Learning deep features for scene recognition using places database," in *Adv. Neural Inf. Process. Syst.*, Montreal, Canada, 2014, pp. 487–495.

[53] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *CoRR*, vol. abs/1409.1556, Sep. 2014. [Online]. Available: http://arxiv.org/abs/1409.1556

[54] A. Vedaldi and K. Lenc, "MatConvNet: Convolutional neural networks for MATLAB," in *Proc. 23rd ACM Int. Conf. Multimedia*, Brisbane, Australia, Oct. 2015, pp. 689–692.

[55] C. M. Bishop, *Pattern Recognition and Machine Learning*. New York, NY, USA: Springer, 2006.

[56] C.-C. Chang and C.-J. Lin, "LIBSVM: A library for support vector machines," pp. 27-1–27-27, 2011. [Online]. Available: http://www.csie.ntu.edu.tw/~cjlin/libsvm

[57] C.-W. Hsu, C.-C. Chang, and C.-J. Lin, "A practical guide to support vector classification," May 2016, Available: http://www.csie.ntu.edu.tw/cjlin/papers/guide/guide.pdf

[58] Y. Zhong, F. Fei, Y. Liu, B. Zhao, H. Jiao, and L. Zhang, "SatCNN: Satellite image dataset classification using agile convolutional neural networks," *Remote Sens. Lett.*, vol. 8, no. 2, pp. 136–145, Feb. 2017.

[59] L. Gueguen, "Classifying compound structures in satellite images: A compressed representation for fast queries," *IEEE Trans. Geosci. Remote Sens.*, vol. 53, no. 4, pp. 1803–1818, Apr. 2015.

[60] S. Chaib, Y. Gu, H. Yao, and S. Zhao, "A VHR scene classification method integrating sparse PCA and saliency computing," in *Proc. 2016 IEEE Int. Geosci. Remote Sens. Symp.*, Beijing, China, Jul. 2016, pp. 2742–2745.

[61] V. Risojević and Z. Babić, "Unsupervised quaternion feature learning for remote sensing image classification," *IEEE J. Sel. Topics Appl. Earth Observ. Remote Sens.*, vol. 9, no. 4, pp. 1521–1531, Apr. 2016.

[62] H. W. an Baozhen Liu, W. Su, W. Zhang, and J. Sun, "Deep filter banks for land-use scene classification," *IEEE Geosci. Remote Sens. Lett.*, vol. 13, no. 12, pp. 1895–1899, Dec. 2016.

[63] B. Zhao, Y. Zhong, G. Xia, and L. Zhang, "Dirichlet-derived multiple topic scene classification model for high spatial resolution remote sensing imagery," *IEEE Trans. Geosci. Remote Sens.*, vol. 54, no. 4, pp. 2108–2123, Apr. 2016.

[64] Q. Zhu, Y. Zhong, B. Zhao, G.-S. Xia, and L. Zhang, "Bag-of-visual-words scene classifier with local and global features for high spatial resolution remote sensing imagery," *IEEE Geosci. Remote Sens. Lett.*, vol. 13, no. 6, pp. 747–751, Jun. 2016.

[65] T. Shi, C. Zhang, H. Ren, F. Li, and W. Liu, “Aerial image classification based on sparse representation and deep belief network,” in *Proc. 35th Chin. Control Conf.*, Chengdu, China, Jul. 2016, pp. 3484–3489.
[66] Y. Liu, Y. Zhong, F. Fei, and L. Zhang, “Scene semantic classification based on random-scale stretched convolutional neural network for high-spatial resolution remote sensing imagery,” in *Proc. 2016 IEEE Int. Geosci. Remote Sens. Symp.*, Beijing, China, Jul. 2016, pp. 763–766.
[67] J. Demšar, “Statistical comparisons of classifiers over multiple data sets,” *J. Mach. Learn. Res.*, vol. 7, pp. 1–30, Jul. 2006.
[68] O. Tamuz, C. Liu, S. Belongie, O. Shamir, and A. Kalai, “Adaptively learning the crowd kernel,” in *Proc. 28th Int. Conf. Mach. Learn.*, Bellevue, WA, USA, Jun. 2011, pp. 673–680.
[69] A. Sitthi, M. Nagai, M. Dailey, and S. Ninsawat, “Exploring land use and land cover of geotagged social-sensing images using naive Bayes classifier,” *Sustainability*, vol. 8, no. 9, pp. 921-1–92-22, 2016.
[70] H. Sun, S. Liu, S. Zhou, and H. Zou, “Transfer sparse subspace analysis for unsupervised cross-view scene model adaptation,” *IEEE J. Sel. Topics Appl. Earth Observ. Remote Sens.*, vol. 9, no. 7, pp. 2901–2909, Jul. 2016.

Yishu Liu received the B.S. degree in computational mathematics from Nanjing University, Nanjing, China, in 1996, and the M.S. and Ph.D. degrees in computational mathematics from Sun Yat-sen University, Guangzhou, China, in 2002 and 2013, respectively.

From 1996 to 1999, she was a Research Assistant with the Institute of Mathematics, Shantou University, Shantou, China. She joined the School of Geography, South China Normal University, Guangzhou, in 2004 as a Lecturer, and currently is an Associate Professor. Her current research interests include machine learning, remote sensing image interpretation, and remote sensing information fusion.

Chao Huang received the B.S. degree in mathematics and applied mathematics from Hunan University, Changsha, China, and the Ph.D. degree in computational mathematics from Sun Yat-sen University, Guangzhou, China, in 2006 and 2011, respectively.

Since 2011, he has been a Lecturer with the College of Mathematics and Statistics, Shenzhen University, China. His current research interests include time-frequency analysis and signal processing.