L2-Net: Deep Learning of Discriminative Patch Descriptor in Euclidean Space

Yurun Tian[1,2] Bin Fan[1] Fuchao Wu[1]
[1]National Laboratory of Pattern Recognition, Institute of Automation,
Chinese Academy of Sciences, Beijing, China
[2]University of Chinese Academy of Science, Beijing, China
{yurun.tian,bfan,fcwu}@nlpr.ia.ac.cn

Abstract

The research focus of designing local patch descriptors has gradually shifted from handcrafted ones (e.g., SIFT) to learned ones. In this paper, we propose to learn high performance descriptor in Euclidean space via the Convolutional Neural Network (CNN). Our method is distinctive in four aspects: (i) We propose a progressive sampling strategy which enables the network to access billions of training samples in a few epochs. (ii) Derived from the basic concept of local patch matching problem, we emphasize the relative distance between descriptors. (iii) Extra supervision is imposed on the intermediate feature maps. (iv) Compactness of the descriptor is taken into account. The proposed network is named as L2-Net since the output descriptor can be matched in Euclidean space by L_2 distance. L2-Net achieves state-of-the-art performance on the Brown datasets [16], Oxford dataset [18] and the newly proposed Hpatches dataset [11]. The good generalization ability shown by experiments indicates that L2-Net can serve as a direct substitution of the existing handcrafted descriptors. The pre-trained L2-Net is publicly available[1].

1. Introduction

Comparing local patches across images is at the base of various computer vision problems, such as wide-baseline matching [17], image retrieval [19] and object recognition [8]. Ever since the advent of the famous SIFT [15] descriptor, encoding local image patches into representative vectors, i.e., descriptors, has been the dominating method. Desired descriptors should be invariant (e.g., robust to view point change, illumination change, or other photometric and geometric changes) for matching patches and distinctive for non-matching patches.

Along with the booming of handcrafted descriptors in the past decade, more and more learning based descriptors appear [12, 24, 20, 16, 21, 7]. Different from handcrafted descriptors which are mostly driven by intuition or researcher's expertise, learning based methods are driven by data. Deep learning has revolutionized many research areas [6, 14], and the public available of large scale dataset with ground truth correspondences [16, 18] makes deep learning possible for local patch matching. The application of Convolutional Neural Network (CNN) for local patch matching can be divided into two categories by whether there are metric learning layers. CNNs with metric learning layers [10, 25, 9] typically treat the matching of local patch pairs as binary classification, so there does not exist the concept of descriptor. An obvious drawback of these models is that they can not perform nearest neighbor search (NNS). On the other hand, CNNs without metric learning layers [2, 5, 9] (i.e., the output descriptors can be matched by L_2 distance) can be used as a direct replacement to previous handcrafted descriptors in many applications, such as the fast approximate nearest neighbor matching (e.g., KD-tree) for large scale structure from motion and the bag of visual words related applications. On the widely used Brown dataset [16], however, models with metric learning generally perform better, and the gap is non-ignorable. Moreover, the generalization of the CNN based descriptors to other datasets(e.g., Oxford dataset [18]) does not show overwhelming superiority to handcrafted descriptors.

As most matching tasks require NNS, we aim at learning high performance descriptor that can be matched by L_2 distance. The proposed L2-Net is a CNN based model without metric learning layers, and it outputs 128 dimensional descriptors, which can be directly matched by L_2 distance. In this paper, we draw inspiration from the basic concept of matching: for a certain local patch, to find its matching counterpart is to do NNS in the descriptor space. Thus, all we need to do is to make sure that the descriptors of matching pairs to be the nearest neighbor (under specific metric like L_2 distance in this paper) of each other, while the magnitude of distance does not really matters. The essence behind this inspiration is relative distance. Although the con-

[1]https://github.com/yuruntian/L2-Net

cept of relative distance is not new, its potentiality in descriptor matching and other related applications is far from being fully explored. Following this idea, we train L2-Net by optimizing the relative distance among descriptors in a batch. Specifically, L2-Net transforms a batch of patches into a batch of descriptors, for each descriptor, our training strategy aims at making its nearest neighbor in a batch to be its correct matching descriptor. In this way, it is actually a one-vs-many operation which considers distances among many patch pairs, going beyond the widely used pairwise or triplet-wise operation [10, 25, 2, 5]. The training of L2-Net is built on a progressive sampling strategy (section 3.3) and a loss function (section 3.4) consists of three error terms. The proposed progressive sampling strategy can be implemented by just one matrix multiplication, which enables fast access to billions of patch pairs in a few dozens of training epochs. As far as we know, the only methods that may share some common concept with ours are [9] and [27]. However, [9] works on the distribution of matching and non-matching pairs while we emphasis on specific pairs, which is more sensitive. The sampling strategy of [27] leads to a non-convex loss function that can not be optimized directly. By comparison, our sampling strategy is fast, efficient and easy to implement. Besides, we integrate three error terms in the loss function: one term accounts for the relative distance among descriptors, one term controls descriptor compactness as well as overfitting, and one term is an extra supervision imposed on the intermediate feature maps, which is named as Discriminative Intermediate Feature maps (DIF). The proposed network is very powerful although not very deep, it achieves state-of-the-art performance on several standard benchmark datasets, receiving significant improvement over previous descriptors and even surpassing those CNN models with metric learning layers. The L2-Net descriptor can be used as a direct substitution of existing handcrafted descriptors since it also uses L2 distance.

2. Related work

The research of designing local descriptor has gradually moved from handcrafted ones to learning based ones. Since the purpose of this paper is descriptor learning, below we give a brief review of descriptor learning methods in the literatures, ranging from traditional methods to the recently proposed CNN based methods. For handcrafted descriptors, please refer to [18] for an overview of classical methods and [13] for recent advances.

Traditional descriptor learning. Early efforts in learning descriptors are not restricted to any specific machine learning method, thus leading to a lot of unique works. PCA-SIFT [12] applies Principal Components Analysis (PCA) to the normalized gradient patch instead of directly using smoothed weighted histograms like SIFT. ASD [24] assumes that local patches under various affine transformations lie in a subspace and PCA is used to extract the base of the subspace as descriptor. [20, 16] emphasizes the learning of pooling region and dimensionality reduction, achieving remarkable performance. Besides float descriptors, there are also learned binary descriptors. BOLD [3] presents a method for adaptive online selection of binary intensity tests so that each bit ensures low variance for intra-class and high variance for inter-class. In Binboost [21], each bit of the descriptor is computed by a boosted binary hash function. RFD [7] proposes to do binary test on the most discriminative receptive fields based on the labeled training data. RMGD [26] introduces a kind of spatial ring-region based pooling method for binary intensity tests, together with an extended Adaboost bit selection. [29] presents a formulation for patch description based on sparse quantization. All the descriptors mentioned above start learning from low level features like gradient or pure binary intensity test, thus, inevitably suffering from information loss. With the help of CNN, we can learn descriptor directly from the raw image patches.

CNN based descriptor learning. Recently, Siamese and triplet networks are the main stream architectures in CNN based descriptor learning. To improve performance, a fully connected layer acting as a metric network is favored by many researchers. MatchNet [10], a typical Siamese network, consists of a feature network for extracting feature representation, a bottleneck layer for reducing feature dimension, and a metric network for measuring similarity of features pairs. It significantly improves previous results, showing a great potential of CNN in descriptor learning. Also based on Siamese network, [25] gose further than MatchNet. It explores different kinds of network architectures and proposes to use a kind of central-surround structure to improve performance. [9] uses triplet network and proposes a global loss function to separate the distribution of matching and non-matching pairs. Together with the metric learning layer and the central surround structure, [9] achieves the currently best performance on the Brown [16] dataset. Despite metric learning improves matching ability, it also limits the versatility of the network. To solve this problem, networks trained without metric learning layers have been proposed. DeepDesc [5] trains the network using L_2 distance by adopting a mining strategy to select hard patches. However, it essentially requires huge amounts of training data to ensure performance. PN-Net [2] uses triplet CNN with a softPN loss that optimizes the distances among patch triplets. Our work also aims to get rid of the metric network and learn high performance descriptor that can be matched by L_2 distance.

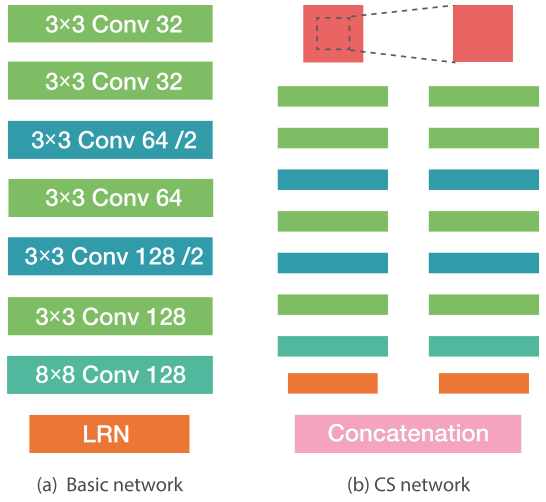

Figure 1. Network Architecture. 3×3 Conv = Convolution +Batch Normalization + Relu. 8×8 Conv = Convolution +Batch Normalization.

3. L2-Net

In this section, we describe in detail of the architecture, training data, sampling strategy, loss function and training of the proposed L2-Net.

3.1. Network architecture

The architecture of L2-Net is depicted in Fig. 1-(a). It takes an all convolution structure, and down sampling is achieved by stride 2 convolution. Batch normalization (BN) [28] is used after each convolutional layer, but with minor modifications, i.e., we do not update the weighting and bias parameters of the BN layers and fix them to be 1 and 0 respectively. Since normalization is an important step in designing descriptors, we use a Local Response Normalization layer (LRN) as the output layer to produce unit descriptors. L2-Net converts 32×32 input patches to 128 dimensional descriptors . As in [25, 9], we also implement a central-surround (CS) L2-Net. It is the concatenation of two separate L2-Nets with a two tower structure as shown in Fig. 1-(b). The input of the tower on the left is the same with a solo L2-Net, while the input of the tower on the right is generated by cropping and resizing the central part of the original patches.

3.2. Training data and preprocessing

For network training, we use the Brown dataset [16] and the newly proposed HPatches dataset [11]. These two datasets are composed of local patches extracted from different scenes. Although diverse in properties, they organize patches in the same way: (i) Each patch in the dataset has a unique 3D point index, patches with identical 3D point index are matching ones. (ii) For each 3D point, there are at least 2 matching patches. Brown dataset consists of three subsets, namely, Yosemite, Notredame, and Liberty. Usually, one of the subsets is picked as training set and the other two subsets are used for testing. The training data of HPatches dataset is composed of four subsets, namely, train-hard (easy) -viewpoint, and train-hard (easy) -illum, indicating that the patches exhibit viewpoint and illumination changes with different degrees. Since the label of its test data is not published at the time we finish this paper, we just use HPatches as training set. There are approximately 500K (1.5M) and 190K (1.2M) 3D points (patches) in Brown dataset and HPatches dataset respectively. All patches are down sampled to the size of 32×32 for training. Based on our experiments, we did not notice any performance degeneration caused by shrinking the patch size. For each patch, we remove the pixel mean calculated across all the training patches, and then contrast normalization is applied, i.e., subtracted by the mean and divided by the standard deviation.

3.3. Progressive sampling of training data

In local patch matching problem, the number of potential non-matching (negative) patches is orders of magnitude larger than the number of matching (positive) patches. Due to the so large amount of negative pairs, it is impossible to traverse all of them, therefore a good sampling strategy is very crucial. Existing methods typically sample equal numbers of positive and negative pairs in training, while the proposed progressive sampling strategy is to break the balance by sampling more negative pairs. Suppose there are P 3D points in the training set. In each iteration, we take p_1 points from the whole set sequentially to traverse all the P points, and then we take an extra of p_2 points form the rest $P - p_1$ points randomly. The randomness brought by the extra p_2 points gives the network a chance to go over what it has learned and be prepared for what it will learn. To form a training batch, we randomly pick a pair of matching patches for each of the p (equals to $p_1 + p_2$) points (thus there are $2p$ patches in a batch). Let $\mathbf{X} = \left\{\mathbf{x}_1^1, \mathbf{x}_1^2, \cdots, \mathbf{x}_i^1, \mathbf{x}_i^2, \cdots, \mathbf{x}_p^1, \mathbf{x}_p^2\right\}_{32\times32\times2p}$ be the 2D patches in a batch, where the subscript is the 3D point index and the superscript is the 2D patch index (e.g., $\mathbf{x}_i^1$ and $\mathbf{x}_i^2$ represents a matching pair from 3D point i). Given $\mathbf{X}$ as input to L2-Net, the output descriptors is denoted as $\mathbf{Y} = \left[\mathbf{y}_1^1, \mathbf{y}_1^2, \cdots, \mathbf{y}_i^1, \mathbf{y}_i^2, \cdots, \mathbf{y}_p^1, \mathbf{y}_p^2\right]_{q\times2p}$, where q is the dimension of the descriptor (128 in this paper). Note that $\mathbf{Y}$ is a batch of unit vectors as the output layer of L2-Net is LRN. Thus, we define the distance matrix $\mathbf{D} = [d_{ij}]_{p\times p}$, where $d_{ij} = \left\|\mathbf{y}_i^2 - \mathbf{y}_j^1\right\|_2$ ($\|\|_2$ is L_2 norm), and $\mathbf{D}$ can be computed by one simple matrix multiplication as

$$\mathbf{D} = \sqrt{\mathbf{2}\left(\mathbf{1} - \mathbf{Y_1^T Y_2}\right)} \tag{1}$$

where $\mathbf{Y_s} = \left[\mathbf{y}_1^s, \cdots, \mathbf{y}_i^s, \cdots, \mathbf{y}_p^s\right]_{q\times p}, (s = 1, 2)$. As a result, $\mathbf{D}$ contains distances of p^2 pairs, namely p positive pairs as the diagonal elements and $p^2 - p$ negative pairs as the off-diagonal elements. For a typical training set with 160K 3D points, with p set to be 128, it means that each training epoch consists of 2500 batches. In each epoch, over 40M ($128^2 \times 2500$) pairs are fed to the network. In our experiments, L2-Net typically needs about 40 training epochs. This indicates that about 1.6 billions pairs (despite the inevitable repetition as a result of randomness, it is still a huge number) are used for training, with the overwhelming majority to be negative pairs and positive pairs only takes up 12.8M ($128 \times 2500 \times 40$).

A possible question would be why we use $\mathbf{Y}_1^\mathbf{T}\mathbf{Y}_2$ instead of $\mathbf{Y}^\mathbf{T}\mathbf{Y}$ to compute $\mathbf{D}$. It is because that if $\mathbf{Y}^\mathbf{T}\mathbf{Y}$ is used, the diagonal elements of D will be all zeros (distances between identical patches), and all positive and negative pairs would be distributed on the off-diagonal elements, making the calculation of the gradient troublesome. In fact, our early work used $\mathbf{Y}^\mathbf{T}\mathbf{Y}$, however, it does not show superiority in performance than using $\mathbf{Y_1^T Y_2}$.

3.4. Loss function

Built upon the progressive sampling strategy, our loss function integrates three objectives. First, we use relative distance to separate matching and non-matching pairs. Second, we emphasize compactness of the output descriptor, which means that all dimensions of the descriptor should be less correlated. Finally, instead of just concentrating on the final output, we also impose constraints on the intermediate feature maps to achieve better performance. According to these objectives, we design three error terms in the loss function.

1) Error term for descriptor similarity. This error term is based on relative distance, i.e., the nearest neighbor of each descriptor in the batch should be its matching counterpart. In $\mathbf{D}$, it would be ideal if

$$\min_{(i,j)\in[1,p]} \{d_{ik}, d_{kj}\} = d_{kk} \tag{2}$$

Equation (2) means that the diagonal element d_{kk} should be the smallest among the kth row and kth coloum. It is equivalent to

$$\begin{cases} \min\limits_{i\in[1,p]} \{d_{ik}\} = d_{kk} \\ \min\limits_{j\in[1,p]} \{d_{kj}\} = d_{kk} \end{cases} \tag{3}$$

For easy implementation, we operate on columns and rows separately. Define the column similarity matrix $\mathbf{S}^c = [s_{ij}^c]_{p\times p}$ and the row similarity matrix $\mathbf{S}^r = [s_{ij}^r]_{p\times p}$ as

$$\begin{aligned} s_{ij}^c &= \exp(2 - d_{ij}) / \sum_m \exp(2 - d_{mj}) \\ s_{ij}^r &= \exp(2 - d_{ij}) / \sum_n \exp(2 - d_{jn}) \end{aligned} \tag{4}$$

where 2 is the maximum L_2 distance between two unit vectors. In equation (4), s_{ij}^c can be interpreted as the probability that y_i^2 is matched to y_j^1, and s_{ij}^r is the probability that y_i^1 is matched to y_j^2. By applying softmax function to each column and each row of $\mathbf{D}$, we can get $\mathbf{S}^c$ and $\mathbf{S}^r$ respectively. The error term for descriptor similarity is defined as

$$E_1 = -\frac{1}{2}\left(\sum_i \log s_{ii}^c + \sum_i \log s_{ii}^r\right) \tag{5}$$

E_1 encourages the descriptors to be closer to their matching counterparts in Euclidean space, while ignoring the specific magnitude of distances, which is the essence of NNS.

2) Error term for descriptor compactness. As the progressive sampling strategy gives L2-Net the access to massive training samples, overfitting becomes inevitable in our initial experiments. An interesting finding is that the degree of overfitting is directly related to the degree of correlation among descriptor dimensions. Thus we introduce an error term that accounts for compactness of the descriptor. By compactness we mean that there should be less redundancy among different dimensions and each dimension should carry as much information as possible so that fewer dimensions could be used to achieve the same performance. In fact, compactness is commonly used in the learning of binary descriptors (such as BOLD [7], RFD [3]), which is typically achieved by greedy selection of bits with high variation. To make it differentiable, we adopt the correlation matrix. Again, we use $\mathbf{Y_s}$ instead of $\mathbf{Y}$ to guarantee that the descriptors for the computation of correlation matrix come from different 3D points. We denote $\mathbf{Y_s^T}$ as $\left[\mathbf{b}_1^s, \cdots, \mathbf{b}_i^s, \cdots, \mathbf{b}_q^s\right]$, where $\mathbf{b}_i^s$ is the row vector.

The correlation matrix $\mathbf{R}_s = [r_{ij}^s]_{q\times q}$ is defined as

$$r_{ij}^s = \frac{(\mathbf{b}_i^s - \bar{b}_i^s)^T(\mathbf{b}_j^s - \bar{b}_j^s)}{\sqrt{(\mathbf{b}_i^s - \bar{b}_i^s)^T(\mathbf{b}_i^s - \bar{b}_i^s)}\sqrt{(\mathbf{b}_j^s - \bar{b}_j^s)^T(\mathbf{b}_j - \bar{b}_j^s)}} \tag{6}$$

where $\bar{b}_i^s$ refers to the mean of the ith row of $\mathbf{Y_s}$. The off-diagonal elements of $\mathbf{R_s}$ is expected to be 0, thus we simply minimize the sum of the squared off-diagonal elements.

$$E_2 = \frac{1}{2}\left(\sum_{i\neq j} \left(r_{ij}^1\right)^2 + \sum_{i\neq j} \left(r_{ij}^2\right)^2\right) \tag{7}$$

We find it is more efficient to put E_2 before the LRN layer, i.e., after the last BN layer. This is because that BN will normalize each channel by subtracting the mean and dividing the standard deviation (note that the weighing and bias are fixed to be 1 and 0). As a result, the computation of correlation matrix can be simplified as

$$\mathbf{R}_s = \mathbf{Y}_s\mathbf{Y}_s^\mathbf{T}/q \tag{8}$$

3) Error term for intermediate feature maps. Existing CNN based methods only focus on the final output descriptors, ignoring the importance of intermediate feature maps. In this paper, we find that it is possible to further increase the performance of L2-Net with extra supervision information provided by the intermediate feature maps. The design of this error term is driven by the same motivation of E_1, i.e., the intermediate feature maps of a patch should also be similar for matching pairs, while distinct for non-matching pairs. Denote the batch of feature maps of the kth layer as $\mathbf{F} = \left[\mathbf{f}_1^1, \mathbf{f}_1^2, \cdots, \mathbf{f}_i^1, \mathbf{f}_i^2, \cdots, \mathbf{f}_p^1, \mathbf{f}_p^2\right]_{(wh)\times 2p}$, where $\mathbf{f}_i^s$ is the vectorized feature map with width w and height h, and index k is omitted for concision. The inner product matrix $\mathbf{G} = [g_{ij}]_{p\times p}$ for intermediate feature maps is computed as

$$\mathbf{G} = (\mathbf{F}_1)^{\mathbf{T}}\mathbf{F}_2 \tag{9}$$

where $\mathbf{F}_s = \left[\mathbf{f}_1^s, \cdots, \mathbf{f}_i^s, , \cdots, \mathbf{f}_p^s\right]_{(wh)\times p}$ $(s = 1, 2)$. Like in equation (3), it would be ideal if

$$\begin{cases} \min\limits_{i\in[1,p]} \{g_{ik}\} = g_{kk} \\ \min\limits_{j\in[1,p]} \{g_{kj}\} = g_{kk} \end{cases} \tag{10}$$

Similarly, relative distance (here measured by inner product) is used to build an error term on $\mathbf{G}$. Same to the definition of E_1, we define the column similarity matrix $\mathbf{V}^c = [v_{ij}^c]_{p\times p}$ as well as the row similarity matrix $\mathbf{V}^r = [v_{ij}^r]_{p\times p}$ on $\mathbf{G}$, where

$$\begin{aligned} v_{ij}^c &= \exp(g_{ij}) / \sum_m \exp(g_{mj}) \\ v_{ij}^r &= \exp(g_{ij}) / \sum_n \exp(g_{jn}) \end{aligned} \tag{11}$$

Thus, the error term for intermediate feature maps is defined as

$$E_3 = -\frac{1}{2}\left(\sum_i \log v_{ii}^c + \sum_i \log v_{ii}^r\right) \tag{12}$$

We name this method as Discriminative Intermediate Feature maps (DIF). Experiments show that it is better to use DIF on normalized feature maps, so we put DIF on the feature maps after BN layers, specifically, only after the first and the last BN layers. This is because that before the first and after the last convolutional layers there are no other convolutional layers, so the order of feature maps is fixed, i.e., the first convolution is directly applied to the input data (each channel of the input data has fixed mathematical or physical meaning) and the output of the last convolutional layer corresponds to the final descriptor. Except for these two, we do not restrict the flexibility of all other feature maps.

To sum up, E_1 is computed on the final output, E_2 is computed after the last BN layer, and E_3 is computed after the first and the last BN layers. The total loss is $E_1 + E_2 + E_3$.

3.5. Training

We train the network from scratch using SGD with a starting learning rate of 0.01, momentum of 0.9 and weight decay of 0.0001. The learning rate is divided by 10 every 20 epochs, and the training is done with no more than 50 epochs. For the training of CS L2-Net, we initialize the two towers using the well trained L2-Net. The parameters of the left tower in Fig. 1-(b) is fixed and we fine tune the right tower until convergence. We let p1 = p2 = q/2 = 64, Data augmentation (optional) is achieved online by randomly rotating (90, 180, 270 degree) and flipping.

4. Experiments

In this section, we provide comparison of the proposed model to the state-of-the-arts. Meanwhile, a series of experiments are conducted to analyze the proposed model.

4.1. Brown dataset

We follow the evaluation protocol of [16] by using the 100K pairs provided by the authors and report the false positive rate at 95% recall. L2-Net is compared with other CNN based models with SIFT(results provided by [10]) as the baseline. Accompany with the float L2-Net descriptor, we further obtain a binary descriptor by simply taking the sign of the float descriptor (± 1). The resulting binary descriptors are denoted as Binary L2-Net and Binary CS L2-Net. To testify the generalization ability of L2-Net, we also train it on HPatches dataset [11]. Results are listed in Table 1 and Table 2.

As can be clearly seen from Table 1, L2-Net performs the best across all the training/testing splits, with remarkable improvement. Besides CS L2-Net, L2-Net already surpasses all models. For the other methods, CS SNet-GLoss clearly outperforms the remaining ones. However, applying CS structure to the models with metric learning will introduce extra parameters to the fully connected layers, thus increasing the time of feature extraction and matching. On the contrary, for CS L2-Net, we use simple concatenation without introducing any extra parameter and the two towers can be used independently. At the same time, binary L2-Net descriptor significantly outperforms those specially designed binary descriptors, and even surpass all float descriptors. Note that the performance of the proposed binary descriptor can be further increased by a better threshold rather than 0 or a better hashing method. Although trained on a totally different dataset (Table 2), L2-Net still achieves state-of-the-art performance, showing its great generalization ability.

4.2. Oxford dataset

In order to further validate the generalization ability of the proposed network, we test it on another totally different

Training	Feature	Notredame Yosemite		Liberty Yosemite		Liberty Notredame		
Test	Dim	Liberty		Notredame		Yosemite		Mean
Metric Learning								
SIFT [15]	128	29.84		22.53		27.29		26.55
MatchNet [10]	4096	6.9	10.77	3.87	5.67	10.88	8.39	7.74
DeepCompare 2ch-2stream [25] +	256	4.85	7.20	1.90	2.11	5.00	4.10	4.19
DeepCompare 2ch-deep [25] +	256	4.55	7.40	2.01	2.52	4.75	4.38	4.26
SNet-GLoss [9] +	256	6.39	8.43	1.84	2.83	6.61	5.57	5.27
CS SNet-GLoss [9] +	384	3.69	4.91	0.77	1.14	3.09	2.67	2.71
Float Descriptors								
TNet-TGLoss [9] +	256	9.91	13.45	3.91	5.43	10.65	9.47	8.8
TNet-TLoss [9] +	256	10.77	13.90	4.47	5.58	11.82	10.96	9.58
PN-Net [2]	256	8.13	9.65	3.71	4.23	8.99	7.21	6.98
DeepDesc [5]	128	10.9		4.40		5.69		6.99
L2-Net	128	3.64	5.29	1.15	1.62	4.43	3.30	3.23
L2-Net +	128	2.36	4.7	0.72	1.29	2.57	1.71	2.22
CS L2-Net	256	2.55	4.24	0.87	1.39	3.81	2.84	2.61
CS L2-Net +	256	**1.71**	**3.87**	**0.56**	**1.09**	**2.07**	**1.3**	**1.76**
Binary Descriptors								
$\mathrm{RFD_R}$ [7]	293-598	19.35	19.40	13.23	11.68	16.99	14.50	15.85
$\mathrm{RFD_G}$ [7]	406-563	17.77	19.03	12.49	11.37	17.62	14.14	15.4
BinBoost [21]	64	20.49	21.67	16.90	14.54	22.88	18.97	19.24
RMGD [26]	1376-1600	15.09	17.42	10.15	10.86	14.46	13.82	13.63
Boixet al [29]	1360	15.6	15.52	-	8.52	-	8.87	12.12
Binary L2-Net	128	10.3	11.71	6.37	6.76	13.5	11.57	10.03
Binary L2-Net +	128	7.44	10.29	3.81	4.31	8.81	7.45	7.01
Binary CS L2-Net	256	5.25	7.83	3.07	3.52	8.49	6.92	5.84
Binary CS L2-Net +	256	**4.01**	**6.65**	**1.9**	**2.51**	**5.61**	**4.04**	**4.12**

Table 1. Performance on the Brown dataset. The numbers are false positive rate at 95% recall. + indicates data augmentation.

Test	Liberty	Notredame	Yosemite	Mean
L2-Net	4.16	1.54	4.41	3.37
L2-Net+	3.2	1.3	3.6	2.7
CS L2-Net	2.43	0.92	2.58	1.97
CS L2-Net+	**1.9**	**0.73**	**1.85**	**1.49**
Binary L2-Net	12.4	6.4	13.16	10.65
Binary L2-Net+	10.74	5.44	11.07	9.08
Binary CS L2-Net	6.43	2.88	6.91	5.4
Binary CS L2-Net+	**5.4**	**2.44**	**5.88**	**4.57**

Table 2. Performance of networks on the Brown dataset when they are trained on HPatches dataset .

dataset, i.e., the Oxford dataset [18]. We evaluate L2-Net on six image sequences, namely, graf (viewpoint), bikes(blur), ubc(JPEG compression), leuven(light), boat(zoom and rotation), and wall(viewpoint). In each image sequence, there are six images sorted in an order of increasing degree of distortions with respect to the first image. Keypoints are detected by Harris-Affine detector and local patches are normalized to the size of 32×32 (64×64 for DeepDesc [5] and TNet-TGLoss [9]) with a scaling factor of 3. We follow strictly the evaluation protocol of [18]. The results of other methods such as [10, 25] on the same dataset can be found in [2], where no improvement over PN-Net is observed. One should note that CNN models with specific learned metric are not suitable for evaluation on the Oxford dataset, as the nearest neighbor search can not be well performed using similarity score. For a fair comparison and without lose of generality, all models are trained on Liberty (DeepDesc [5] is trained on Liberty and Notredame). Besides learned descriptors, we use LIOP [23] as the baseline of handcrafted descriptors, since it was reported to surpass most of the handcrafted descriptors on this dataset. Meanwhile, Binary L2-Net and Binary CS L2-Net are compared to other state-of-the-art binary descriptors. Moreover, we report results with different training data. Experimental results are shown in Fig 2 with mean average precision (mAP) as performance indicator.

As can be clearly seen from Fig 2, L2-Net outperforms all the other descriptors on average and even the binary L2-Net descriptor surpasses all other float descriptors. Moreover, there are some other interesting observations: i) The

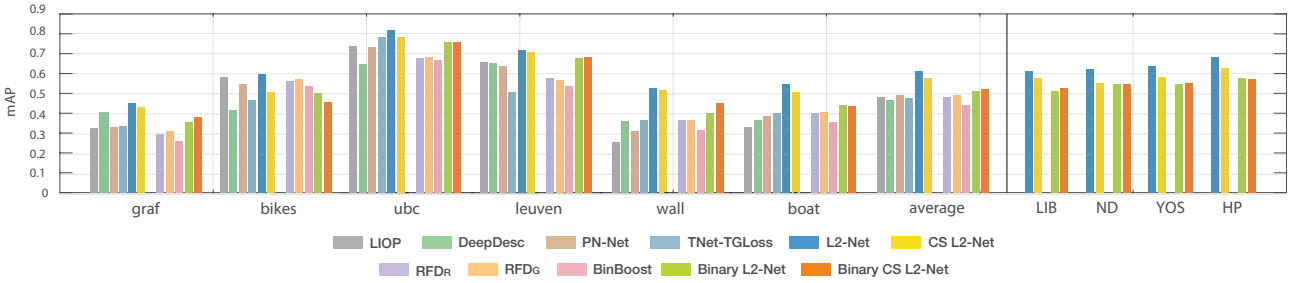

Figure 2. Performance comparison on Oxford dataset in terms of mAP. Performance with respect to different training set is shown in the right part of the figure (the average mAP over six image sequence).

CS structure does not guarantee performance improvement on all datasets and all types of descriptors (float and binary). Since CS structure needs to crop the central part of the patch, how to choose the scale of the patch becomes a problem. Patches in Brown dataset and Hpatches dataset are of similar scale, thus CS structure works fine. However, with different detector and scale, arbitrarily cropping the central 50% of the patch (can be less textured) may not be a good choice. ii) In accordance with [4], we also find that CNN based methods is very sensitive to image blur. iii) Hpatches dataset shows better generalization ability.

4.3. Hpatches dataset

Results of the prototype L2-Net (casia-yt) on the test data of Hpatches dataset can be found at the webpage of ECCV 2016 workshop "Local Features: state of the art, open problems and performance evaluation" [2], where our method ranked No.1 in all the three tasks.

4.4. Discussion and analysis

In this section, we discuss how each of the proposed error terms contributes to the final performance and give some qualitative analysis for the binarized descriptor.

Importance of compactness. We try to train L2-Net without E_2, however, the network does not converge. Due to the large amount of training samples fed to the network, it is easier for the network to memorize the training data rather than learn to generalize. Without E_2, strong overfitting happens and the dimensions of the output descriptor are highly correlated. Therefore, compactness is of crucial importance to the progressive sampling strategy. By restricting compactness, the network actually tends to extract uncorrelated features containing more information.

Advantage of relative distance. E_1 is quite different from the widely used hinge loss. Typically, hinge loss for patch pair and triplet can be written as

$$
\begin{aligned}
E_{pair} &= \delta_{ij} \max\left(0, \left\|\mathbf{y}_i - \mathbf{y}_j\right\|_2 - t_p\right) \\
&+ (1-\delta_{ij}) \max\left(0, t_n - \left\|\mathbf{y}_i - \mathbf{y}_j\right\|_2\right) \\
E_{triplet} &= \max\left(0, 1 - \frac{\left\|\mathbf{y}_i - \mathbf{y}_i^-\right\|_2}{\left\|\mathbf{y}_i - \mathbf{y}_i^+\right\|_2 + t}\right)
\end{aligned}
\tag{13}
$$

where δ_{ij} equals to 1 if y_i and y_j are matching, otherwise δ_{ij} equals to 0. t, t_p, t_n are thresholds, whose optimal values are difficult or even impossible to find, so they are mostly decided by experience. A major drawback of hinge loss is the unstable gradient caused by thresholding. As training proceeds, it is not sure how many samples in a batch are contributing to the overall gradient, and unstable gradient may lead to a bad local minima. To tackle this problem, many researchers resort to hard sample mining, however, the nature of mining is still thresholding (more strict). By utilizing relative distance, the absolute value of distances becomes useless, thus there is no need to use thresholds.

Effectiveness of DIF. First, we simply remove E_3 from the error function to prove the effectiveness of DIF, and then we impose DIF after every BN layer to test its performance. Comparing curve A with curve B and D in Fig. 3-(b), it can be found that since DIF can provide more supervision in training, L2-Net with DIF works consistently better than that without it. However, DIF can not be over used as it will limit the solution space of the network.

Batch normalization. The weighting α and bias β are fixed to be 1 and 0 in our BN layers, as we find learning them makes the output feature maps (and descriptors) in poor distribution. In this experiment, the weighing and bias parameters of all BN layers is learned except the two BN layers before DIF (as DIF depends on the normalized features). The training procedure is shown by curve C in Fig. 3-(a). Comparing curve A with C, we can find that updating α and bias β leads to minor performance decline. As an illustration to this phenomenon, suppose $a \sim N(\mu_1, \sigma_1)$ and $b \sim N(\mu_2, \sigma_2)$ are two random variables obeying gaus-

[2] http://www.iis.ee.ic.ac.uk/ComputerVision/DescrWorkshop/index.html

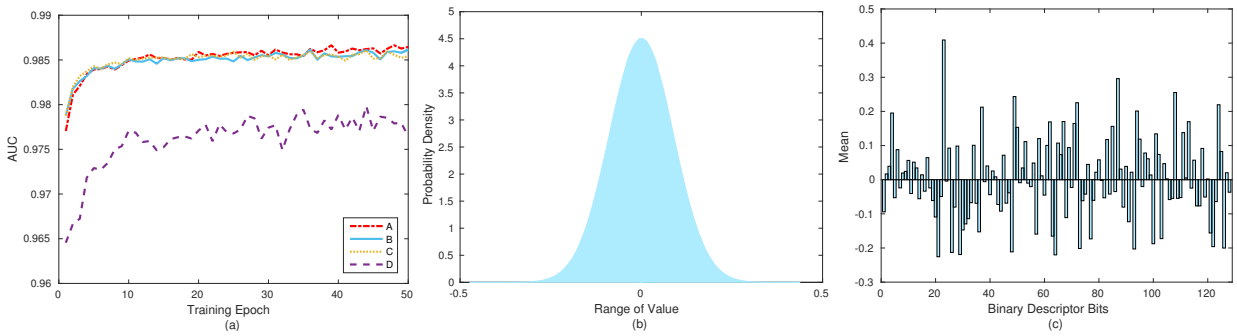

Figure 3. Model analysis. L2-Net is trained on Hpatches dataset and tested on Brown dataset. (a): Effect of DIF and BN. Area under the ROC curve (AUC) of different training epoch (averaged over three subsets of Brown dataset) is served as an indicator. Curve A: default setting. Curve B: default - DIF. Curve C: default + learned BN parameters. Curve D: default + extra DIF. (b): Distribution of the float L2-Net descriptor. (c): Mean of each bit of the binary L2-Net descriptor.

sian distribution. It is not difficult to understand that the easiest way to separate them is to increase $|\mu_1 - \mu_2|$ while decrease $|\sigma_1|$ and $|\sigma_2|$. As a result, learning α and β may cause the extracted feature to be sharp and non-zero distributed, which damages the performance. Fixing them is based on the principal that we want the feature maps (and descriptors) of different patches to be independent identically distributed. In this way, the network is forced to extract features that are highly discriminative rather than biased.

Property of the binarized descriptor. We aim to provide an insightful explanation for the good performance of the binarized descriptor, despite the fact that it is just a by-product of the proposed float descriptor. Thus, we randomly select 100K patches from different 3D points to investigate the value distribution of the proposed descriptor. Fig. 3-(b) shows that the output values of the L2-Net approximately obey the Gaussion distribution with zero mean. In Fig. 3-(c), each bit of the binarized descriptor has a mean near 0, which is highly desirable for a good binary descriptor. Moreover, we know that hamming distance can be computed by inner product (for vectors consist of +1 and -1) and DIF is just built on inner product, which means there could be strong connections between DIF and the performance of the binary descriptor. We will leave the in depth analysis in the future work.

4.5. Trianing and extraction speed

We use a GTX 970 GPU in MatConvNet [22]. L2-Net reaches maximum performance in 20 to 50 epochs. Without online data augmentation, it takes only 2 to 4 hours (4 to 6 hours with online data augmentation). Note that with a more powerful GPU, the training time will undoubtedly reduce. L2-Net can extract descriptors at the speed of approximately 21.3K patch/sec.

5. Conclsion

In this paper, we propose a new data-driven descriptor that can be matched in Euclidean space and significantly outperforms state-of-the-arts. Its good performance is mainly attributed to a new progressive sampling strategy and a dedicated loss function containing three terms. By progressive sampling, we manage to visit billions of training samples. By going back to the basic concept of matching (NNS), we thoroughly explore the information in each batch. By requiring compactness, we successfully handle overfitting. By utilizing intermediate feature maps, we further boost the performance. Moreover, a powerful binary descriptor is obtained by directly taking the sign of the learned float descriptor, which gives the best performance among existing binary descriptors and even outperforms most float descriptors. Finally, L2-Net should be further extended to more applications such as image classification and retrieval. We will leave these problems as the future work.

6. Acknowledgments

This work was supported by the National High Technology Research and Development Program of China (863 Program 2015AA020504) and the National Natural Science Foundation of China (Nos. 61375043, 61672032, 61472119, 61403375). Y. Tian thanks Shenglong Guo and Chenhua Li for some useful discussions.

References

[1] J. Bromley, I. Guyon,Y. LeCun, E. Sckinger, and R. Shah. Signature verification using a siamese time delay neural network. In *NIPS*, 1994.

[2] V. Balntas, E. Johns, L. Tangand K. Mikolajczyk. PN-Net: Conjoined triple deep network for learning local image descriptors. *Arxiv*, 2016. 1, 2, 6

[3] V. Balntas, L. Tang, and K. Mikolajczyk. Bold: binary online learned descriptor for efficient image matching. In *CVPR*, 2015. 2, 4

[4] A. Dosovitskiy, P. Fischer, Jost. Springenberg, M. Riedmiller, and T. Brox Discriminative Unsupervised Feature Learning with Exemplar Convolutional Neural Network *Arxiv*, 2015 7

[5] E. Simo-Serra, E. Trulls, L. Ferraz, I. Kokkinos, P. Fua, and F. Moreno-Noguer. Discriminative learning of deep convolutional feature point descriptors. In *ICCV*, 2015. 1, 2, 6

[6] D. Erhan, C. Szegedy, A. Toshev, and D. Anguelov Scalable object detection using deep neural networks. In *CVPR*, 2014. 1

[7] B. Fan, Q. Kong, T. Trzcinski, Z. Wang, C. Pan, and P. Fua. Receptive fields selection for binary feature description. *TIP*, 23(6):2583–2595, 2014. 1, 2, 4, 6

[8] R. Fergus, P. Perona, and A. Zisserman. Object class recognition by unsupervised scale-invariant learning. In *CVPR*, 2003. 1

[9] V. Kumar B G, G. Carneiro, and I. Reid. Learning local image descriptors with deep siamese and triplet convolutional networks by minimising global loss functions. In *CVPR*, 2016. 1, 2, 3, 6

[10] X. Han, T. Leung, Y. Jia, R. Sukthankar, and A.C. Berg. Matchnet: Unifying feature and metric learning for patch-based matching. In *CVPR*, 2015. 1, 2, 5, 6

[11] V. Balntas, K. Lenc, A. Vedaldi and K. Mikolajczyk. HPatches: A benchmark and evaluation of handcrafted and learned local descriptors. In *CVPR*, 2017. 1, 3, 5

[12] Y. Ke and R. Sukthankar. Pca-sift: A more distinctive representation for local image descriptors. In *CVPR*, 2004. 1, 2

[13] B. Fan, Z. Wang and F. Wu. Local Image Descriptor: Modern Approaches. *Springer*, 2015. 2

[14] A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. In *NIPS*, 2012. 1

[15] D. Lowe. Distinctive image features from scale-invariant keypoints. *IJCV*, 20(2), 2004. 1, 6

[16] M. Brown, G. Hua and S.A.G. Winder. Discriminative learning of local image descriptors. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2010. 1, 2, 3, 5

[17] J. Matas and O. Chum. Robust wide-baseline stereo from maximally stable extremal regions. *Image and Vision Computing*, 22(10), 2004. 1

[18] K. Mikolajczyk and C. Schmid. A performance evaluation of local descriptors. *PAMI*, pages 257–263, 2003. 1, 2, 6

[19] J. Philbin, O. Chum, M. Isard, J. Sivic, and A. Zisserman. Object retrieval with large vocabularies and fast spatial matching. In *CVPR*, 2007. 1

[20] K. Simonyan, A. Vedaldi, and A. Zisserman. Learning local feature descriptors using convex optimisation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2014. 1, 2

[21] T. Trzcinski, M. Christoudias, P. Fua, and V. Lepetit. Boosting Binary Keypoint Descriptors. In *CVPR*, 2013. 1, 2, 6

[22] A. Vedaldi and K. Lenc. Matconvnet - convolutional neural net-works for matlab. *CoRR, abs/1412.4564*. 8

[23] Z. Wang, B. Fan, and F. Wu. Local intensity order pattern for feature description. In *ICCV*, 2011. 6

[24] Z. Wang, B. Fan, and F. Wu. Affine subspace representation for feature description. In *ECCV*, 2014. 1, 2

[25] S. Zagoruyko and N. Komodakis. Learning to compare image patches via convolutional neural networks. In *CVPR*, 2015. 1, 2, 3, 6

[26] Y. Gao, W. Huang, and Y. Qiao Local Multi-Grouped Binary Descriptor with Ring-based Pooling Configuration and Optimization. *IEEE Transactions on Image Processing*, 24(12):4280–4833, Jan 2014. 2, 6

[27] H. O. Song, Y. Xiang, S. Jegelka, S. Savarese Deep Metric Learning via Lifted structured feature Embedding. In *CVPR*, 2016 2

[28] Ioffe, Sergey and Szegedy, Christian. Batch normalization: Accelerating deep network training by reducing internal covariate shift. *Arxiv*, 2015. 3

[29] X. Boix, M. Gygli, G. Roig and L. V. Gool. Sparse Quantization for Patch Description. In *CVPR*, 2013. 2, 6