DeepCD: Learning Deep Complementary Descriptors for Patch Representations

Tsun-Yi Yang[1,2] Jo-Han Hsu[1,2] Yen-Yu Lin[1] Yung-Yu Chuang[2]
[1]Academia Sinica, Taiwan [2]National Taiwan University, Taiwan
shamangary@citi.sinica.edu.tw r04922012@ntu.edu.tw yylin@citi.sinica.edu.tw cyy@csie.ntu.edu.tw

Abstract

This paper presents the DeepCD framework which learns a pair of complementary descriptors jointly for image patch representation by employing deep learning techniques. It can be achieved by taking any descriptor learning architecture for learning a leading descriptor and augmenting the architecture with an additional network stream for learning a complementary descriptor. To enforce the complementary property, a new network layer, called data-dependent modulation (DDM) layer, is introduced for adaptively learning the augmented network stream with the emphasis on the training data that are not well handled by the leading stream. By optimizing the proposed joint loss function with late fusion, the obtained descriptors are complementary to each other and their fusion improves performance. Experiments on several problems and datasets show that the proposed method[1] is simple yet effective, outperforming state-of-the-art methods.

1. Introduction

Representing a local image patch by a descriptor is a fundamental computer vision task with a variety of applications such as image matching [7, 15], alignment [14], structure from motion and object recognition. Many descriptor methods have been proposed and early descriptors are mostly hand-crafted [22, 36, 27, 6, 19, 4, 33, 30]. Although with decent performance in many applications, manually designed rules can not be thorough and often lead to sub-optimal results. Recent advances on deep learning have inspired several descriptor methods to further improve descriptor performance by exploring convolutional neural networks (CNNs).

Deep learning has been used for learning both descriptor embeddings and similarity metrics. Metric learning has been achieved by decision network [40] or pairwise similarity learning [17]. Significant improvement can be obtained but at the cost of very expensive computation during testing. As an example, one of such metric learning methods could take 10^4 times of computation than using the L_2 norm as the distance metric between patches. Such a high testing cost prevents them from being employed in many applications despite their excellent performance. Thus, in this paper, we focus on descriptor embedding and simply use L_2 norm or the Hamming distance as the dissimilarity metrics.

This paper proposes to learn two complementary descriptors jointly. The approach is inspired by feature fusion. The fusion of multiple features has been shown effective in vision and multimedia applications. It has been shown that, as long as the features are complementary, even simple late fusion schemes by adding or multiplying scores can significantly improve results. However, for manually crafted features, it is impossible to know in advance whether they are complementary. Thus, the main challenge of feature fusion is to select complementary features. By employing deep learning for descriptor embedding, we could learn descriptors that are guaranteed complementary to each other by formulating the complementary property into the loss function. By complementary, we mean that using these descriptors jointly outperforms using them individually.

This paper proposes the DeepCD framework for learning complementary descriptors and formulates a joint loss function for enforcing the complementary property. The proposed DeepCD framework learns a complementary descriptor along with a leading descriptor. It can be done by taking any descriptor learning architecture and augmenting it with a network stream for learning the complementary descriptor. By optimizing the proposed joint loss function, we obtain a leading descriptor which performs well by itself and its complementary descriptor which focuses on helping the leading one. Note that the complementary descriptor could perform poorly by itself. It is important that the descriptors are asymmetric. By requiring that one is primary, the other can focus on learning residual information to help the primary one. It would be easier for the complementary one to focus on the failure cases that the primary one can not perform well. We further propose a novel network layer termed data-dependent modulation (DDM) layer for adaptively modulating the learning rate of the complementary

[1]Source code avaliable at https://github.com/shamangary/DeepCD

network stream in a data-specific fashion. It follows that the corner cases are better addressed with the extra complementary descriptor.

The main contributions of the paper are as follows. First, we propose the concept of joint embedding of complementary descriptors. Second, we provide a design of the joint loss function coupled with the DDM layer for enforcing the complementary property properly. Finally, the proposed method is simple yet effective. We applied it to several problems and datasets and obtained significant improvement against the state of the art.

2. Related work

Early hand-crafted descriptor methods, such as SIFT [22], LIOP [36] and CovOpt [30], encode spatial information into the descriptors. Our method employs deep learning techniques for extracting descriptors and we will review related deep learning methods in this section.

2.1. Deep learning methods for descriptors

Fischer *et al*. presented an early attempt on applying deep learning to descriptors [8]. Although showing significant improvement against SIFT, the output dimension is too high and the method requires a lot of training data. Later, more progresses on descriptors have been made on learning both descriptors and metrics.

Learning descriptor embedding. Methods of this category learn descriptor embedding but use conventional metrics such as L_2 norm. Several loss functions have been proposed for improving descriptor embedding, such as hinge loss [29], SoftMax ratio [13], SoftPN [2, 3], and Margin ranking loss [3]. They have been used in siamese and triplet networks. Different sampling schemes have also been proposed in Deepdesc [29] and PNNet [2]. Several training schemes were proposed in different application areas, such as face feature embedding by using triplet network training [28, 1] or by two-stage hashing scheme [41], and image retrieval using lifted structure [31]. Some focused on learning a pipline for image matching [39, 38].

Decision network. In addition to embedding, several methods also learn similarity metrics, such as Deepcompare [40] and Matchnet [11]. For computing the similarity, several fully connected layers called decision network are used. Kumar *et al*. further improved the performance by optimizing global loss [17]. These methods often achieve great performance at an expensive cost on memory and computation.

2.2. Other related deep learning methods

Fusion of information. For fine-grained recognition, Lin *et al*. proposed to use two distinct streams and combine them into a single result using bilinear pooling [21]. This method greatly improves the recognition performance, but creates a very high dimensional feature vector. A more compact method called compact bilinear pooling (CBP) [10, 9] has been proposed.

Joint loss functions. Several joint loss functions have been proposed for knowledge distillation [12]. For instance, FitNets [26] adopts two separate network models, the teacher and the student networks, and the optimization focuses on copying the output results from the teacher network to the student network with a smaller size. Although it achieves an impressive compression rate, for our application, mimicking the existing teacher network will not provide any gain.

With a fixed learning rate in the leading stream, the proposed DDM layer dynamically determines the learning rate of the augmented stream for each training sample. The DDM layer enables the adaptive collaboration between the leading and the augmented streams.

3. DeepCD framework

Given an image patch q, the goal of a feature descriptor method is to find a mapping from q to its representation $d(q)$, which is an n-d vector (either real-valued or binary). In the past, most descriptor mappings were hand-crafted. Recently, convolutional neural networks (CNNs) have been employed for learning the embedding of feature descriptors.

Given a pair of patches q_i and q_j, the dissimilarity Δ_{ij} between them is defined as

$$\Delta_{ij} \triangleq D(d(q_i), d(q_j)), \tag{1}$$

where D is the distance function. Typically, L_2 norm is used for real-valued descriptors and the Hamming distance is often used for binary ones. However, some recent deep learning descriptor methods also learn complex functions D for measuring distances between descriptors. Such a learned metric often requires a decision network and is often very slow to compute. Thus, in this paper, we only focus on learning the descriptor embedding and use the more efficient L_2 norm or Hamming distance as the metric. This way, various well-studied techniques such as KD-tree or ANN could be more readily applied [2].

Recent deep learning methods of feature descriptors utilize patch pairs $\{q_i, q_j\}$ and find a good mapping by minimizing the Hinge embedding loss $L(\Delta_{ij})$, which penalizes corresponding pairs that are placed far apart and non-corresponding pairs that are close in the descriptor space. The descriptor d can then be found by optimizing

$$d = \arg\min_{d} \sum_{ij} L(\Delta_{ij}). \tag{2}$$

Instead of learning a single descriptor, the proposed DeepCD method learns two descriptors jointly for a single patch q, a leading descriptor $d(q)$ and a complementary descriptor $\overline{d}(q)$. That is, we would learn a joint embedding

$(d, \overline{d})$ for a patch. Similar to the dissimilarity Δ_{ij} for the leading descriptor, we define the dissimilarity $\overline{\Delta}_{ij}$ for the complementary descriptor as

$$\overline{\Delta}_{ij} \triangleq \overline{D}(\overline{d}(q_i), \overline{d}(q_j)), \tag{3}$$

where $\overline{D}$ is the distance function for complementary descriptors. Again, we simply use L_2 norm if the complementary descriptor is real-valued and the Hamming distance if it is binary. Finally, we define a joint loss function as:

$$J(q_i, q_j) = L(\Delta_{ij}) + \lambda L'(\Phi(\Delta_{ij}, \overline{\Delta}_{ij})), \tag{4}$$

where λ is a weighting factor for determining the relative importance of both terms; Φ is a late fusion function which combines two dissimilarity scores into a single one; and both L and L' play the similar role as explained above. The first term in Equation 4 encourages the leading descriptor performs well by itself while the second term requires that the combination of the leading descriptor and the complementary descriptor performs well. Thus, the leading descriptor learns to be a good descriptor alone while the complementary descriptor focuses on helping the leading one by learning residual information. The proposed DeepCD framework finds the joint embedding by optimizing

$$(d, \overline{d}) = \arg\min_{d, \overline{d}} \sum_{ij} J(q_i, q_j). \tag{5}$$

In the DeepCD framework, one has to determine the late fusion function Φ and the network architecture for learning the leading descriptor d and the complementary descriptor $\overline{d}$. For late fusion, there are several strategies for combining the scores of multiple features, such as the sum, product and maximum rules. In this paper, we employ the product rule for similar reasons as Zheng *et al.* [35]. First, it has been demonstrated that the product rule has very similar, if not superior, performance to the sum rule. Second, the product rule adapts well to input data with different scales and does not require heavily a proper normalization of the data.

3.1. Network architectures

We decide to use a real-valued leading descriptor while having a binary complementary descriptor. The leading descriptor is real-valued because real-valued descriptors are usually more effective. We opt for a binary complementary descriptor for two reasons. First, binary descriptors are more compact. Second, different types of descriptors have a better chance to be complementary to each other.

As for network architecture, in this paper, we choose to model after the architecture of PNNet [2] because of its state-of-the-art performance. The top of Figure 1 displays the architecture of the PNNet. There are essentially two streams in the DeepCD network, one for the leading descriptor and the other for the complementary descriptor. We

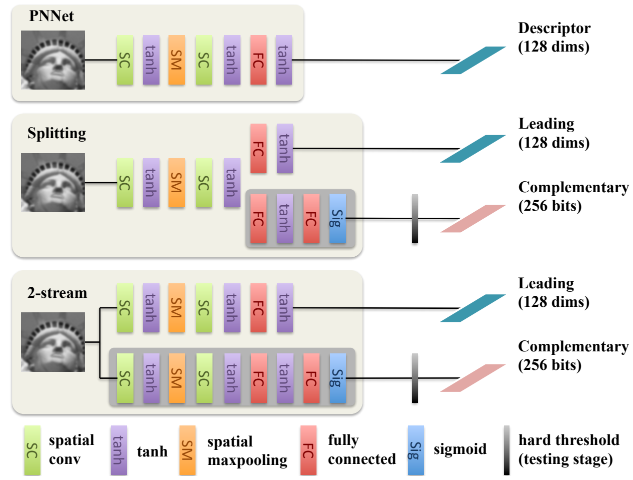


Figure 1. Network architectures of the proposed models. The top shows the PNNet model [2]. The middle illustrates the splitting model. Starting with a single stream, the model splits into two streams after the fifth layer. The leading stream continues to model after PNNet while the complementary stream has its own fully-connected layers and sigmoid transfer layer for making a binary descriptor. The bottom shows the 2-stream model which uses two separated streams from the beginning, one for leading and the other for complementary. The gray blocks denote the augmented parts of the proposed DeepCD framework.

propose two network models, the splitting model and the 2-stream model. They are demonstrated in Figure 1, the gray blocks denote the parts augmented to the PNNet model.

Splitting model. The PNNet model can be represented as {Conv(7,7)-Tanh-SpatialMaxPool(2,2)-Conv(6,6)-Tanh-FC(4096→128)-Tanh}. In this model, there is a single stream modelled after PNNet at the beginning. After the fifth layer, the stream is split into two streams, the leading stream continues PNNet while the complementary stream is formed by augmenting {FC(4096→128)-Tanh-FC(128→256)-Sigmoid[2]}. The sigmoid function plays the role for smooth binarization. The inputs of the leading stream and the complementary stream are the same from the previous layer[3]. The model is demonstrated at the middle of Figure 1. Note that, during training, we do not actually quantize the complementary descriptor into binary so that it is easier to optimize.

2-stream model. This model uses two separate streams for the two descriptors from the beginning to the end. The leading stream is the same as the PNNet model, while the complementary stream is {Conv(7,7)-Tanh-SpatialMaxPool(2,2)-Conv(6,6)-Tanh-FC(4096→128)-Tanh-FC(128→256)-Sigmoid}. The bottom of Figure 1 illustrates the model. It is obvious that the model size of the splitting model is much smaller than that

[2]To neutralize the effect of quantization error, we use the sigmoid function, $1/(1 + exp(-100 * t))$.

[3]It was implemented by using torch function $torch.ConcatTable()$.

of the 2-stream model. On the other hand, the 2-stream model offers more flexibility and has better potential for obtaining a better complementary binary descriptor.

3.2. Joint optimization for triplets

Similar to PNNet, we use triplets for training rather than pairs in Equation 5. A triplet contains three patches, q_a, q_p and q_n, where q_a is the anchor patch; q_p is a positive patch which is a correspondence of q_a; and q_n is a negative patch which is not a correspondence of q_a. There are three patch pairs in a triplet, one positive ($\{q_a, q_p\}$) and two negative ($\{q_a, q_n\}, \{q_p, q_n\}$). The joint loss function for a triplet is

$$J(q_a, q_p, q_n) = L(\Delta_{ap}, \Delta_{an}, \Delta_{pn}) + \lambda L' \left(\sqrt{\Delta_{ap}\overline{\Delta}_{ap}}, \sqrt{\Delta_{an}\overline{\Delta}_{an}}, \sqrt{\Delta_{pn}\overline{\Delta}_{pn}} \right), \quad (6)$$

where $\lambda = 5$ and both L and L' are essentially the SoftPN function defined in the PNNet model [2],

$$L(\delta^+, \delta_1^-, \delta_2^-) = \left[e^{\delta^+} / \left(e^{\min(\delta_1^-, \delta_2^-)} + e^{\delta^+} \right) \right]^2 + \left(\left[e^{\min(\delta_1^-, \delta_2^-)} / \left(e^{\min(\delta_1^-, \delta_2^-)} + e^{\delta^+} \right) \right] - 1 \right)^2, \quad (7)$$

but for L', we do not take $\min$. Note that we assign a higher weight to the second term of Equation 6 since the fusion score will be taken as the final score during the testing stage.

One thing to note is that the ranges of the distances for the leading and the complementary descriptors are different and a proper normalization is necessary. Since $tanh$ is used as the transfer function for the leading descriptor, each component is within the range $[-1, +1]$. Assume that we use a 128-d leading descriptor and a 256-bit complementary descriptor, the maximum distance for the leading descriptor is 128×2^2 while the maximum distance for the complementary descriptor is 256×1. Thus, the distance of the complementary descriptor needs to be scaled by 2 to match up.

3.3. Data-dependent modulation layer

To further enhance the complementary property between the two network streams, we introduce a data-dependent modulation (DDM) layer. The DDM layer dynamically learns the *attenuation factors* of the learning rate in the augmented stream conditioned on training data. That is, the learning rate will vary from training sample to training sample in the augmented stream. This way, the augmented stream is equipped with the flexibility in putting more emphasis on the training data that are not well represented by the leading stream.

When stochastic gradient descent is used for network optimization, the DDM layer takes as input the set of all pairwise leading and complementary distances in the batch, *i.e.*, $\mathbf{x} = [\cdots \Delta_{ij}, \overline{\Delta}_{ij}, \cdots]$. The DDM layer is composed of a fully connected layer followed by an activation function. We use the sigmoid function here so that each element of the output, $\mathbf{w} = [\cdots w_k \cdots] \in \mathbb{R}^b$, is ranged between 0 and 1, where b is the batch size. The output $\mathbf{w}$ then serves as the attenuation factors of the learning rate in the augmented stream. Namely, the learning rate for the kth training sample of the batch in backward propagation is adaptively set to the original learning rate multiplied by w_k. Since the learning rate in the leading stream remains fixed, the dynamic learning rate in the augmented stream controls the relative importance of the two streams in a sample-dependent way. It is worth mentioning that the DDM layer is involved only in backward propagation. Thus, no extra computation is required in the stage of testing.

Unlike SoftPN [3, 2] and sMCL [18] which only update a single network stream for a training sample, the DDM layer allows the two streams in the DeepCD framework to be learned adaptively and jointly so that they can better collaborate to make predictions through late fusion. For smoother optimization, we follow the strategy of STN [16]: the learning rate of the DDM layer is set to $10^{-3} \sim 10^{-4}$ of the base learning rate for the rest of the network.

3.4. Comparing two patches

Both L and L' in Equation 6 are functions of descriptors d and $\overline{d}$, the outputs of DeepCD. They are both differentiable. Thus, the proposed DeepCD framework can be optimized by using stochastic gradient descent (SGD). After optimization, we obtain the joint feature descriptor $(d, \overline{d})$. Given two patches q_i and q_j, we first extract their descriptors $(d(q_i), \overline{d}(q_i))$ and $(d(q_j), \overline{d}(q_j))$. The dissimilarity of the leading descriptor is $\Delta_{ij} = \|d(q_i) - d(q_j)\|_2^2$ and the one for the complementary is $\overline{\Delta}_{ij} = \|\overline{d}(q_i) - \overline{d}(q_j)\|_2^2$ (implemented by calculating Hamming distance since the descriptor is binary). The final dissimilarity between two patches is calculated by the product late fusion as the product $\Delta_{ij}\overline{\Delta}_{ij}$.

Note that in practice, the proposed scheme allows us to compute $\overline{\Delta}_{ij}$ first by using efficient Hamming distance calculation or even hash-table-based search [24, 25] and then determine whether it is necessary to calculate the more expensive Δ_{ij}. For example, for similar patches with very small $\overline{\Delta}_{ij}$ values, one could skip the calculation of Δ_{ij} and declare that they are matched.

4. Experiments

We implemented the proposed method with Torch and Matlab. The experiments were performed on a Linux machine with NVIDIA GTX1070. Our models were trained using SGD with the parameters, 0.1 learning rate, 10^{-6} decay rate, 10^{-4} weight decay and 0.9 momentum. The batch size is 128. We applied the proposed method to a few problems and datasets, including a local image patch benchmark (Brown dataset [5]), a wide baseline matching

Test	Train	des.	2-stream PNNet +CBP	Deep comp	Deep desc	TNet	PNNet		TFeat Margin*	DeepCD splitting	DeepCD 2-stream	DeepCD 2-stream (DDM)
		bytes	256	512	128	256	128	256	128	128+32		
Notre.	Lib.		16.99	4.54	5.16	3.91	3.81	3.71	3.12	2.98	2.87	**2.59**
	Yose.		18.22	5.58		5.43	4.45	4.23	3.85	3.51	3.02	**2.95**
Yose.	Lib.		34.46	13.24	18.21	10.65	9.55	8.99	7.82	8.27	7.78	**7.03**
	Notre.		30.70	13.02		9.47	7.74	7.21	7.08	6.85	**6.65**	6.69
Lib.	Notre.		21.33	8.79	9.56	9.91	8.27	8.13	7.22	6.32	6.08	**5.85**
	Yose.		32.47	12.84		13.45	9.76	9.65	9.79	9.10	**7.76**	7.82
mean			25.70	9.67	10.98	8.8	7.26	6.98	6.47	6.17	5.69	**5.48**

Table 1. FPR95 for real-valued descriptors on the Brown dataset.

Test	Train	des.	BinBoost	CovOpt	Deepbit	DeepCD splitting		DeepCD 2-stream		DeepCD 2-stream (DDM)	
		bits	64	1024	256	128+64	512+256	128+64	512+256	128+64	512+256
Notre.	Lib.		16.90	8.25	26.66	13.34	3.96	8.43	3.99	8.31	**3.73**
	Yose.		14.54	7.09	29.60	11.59	4.95	9.07	4.68	8.97	**4.35**
Yose.	Lib.		22.88	14.84	57.61	26.41	**9.88**	17.62	10.31	16.29	9.97
	Notre.		18.97	8.5	63.68	17.72	7.76	13.99	7.86	13.62	**7.67**
Lib.	Notre.		20.49	12.16	32.06	16.07	8.06	13.29	**7.22**	14.45	7.82
	Yose.		21.67	15.15	34.41	21.85	12.06	18.06	**11.50**	18.38	11.75
mean			19.24	11.00	40.67	17.83	7.78	13.41	7.59	13.34	**7.55**

Table 2. FPR95 for binary descriptors on the Brown dataset.

dataset (Strecha dataset [32]) and a local descriptor evaluation benchmark (Oxford dataset [23]).

For training, we used the Brown dataset [5] which consists of three subsets: Liberty, Notredame, and Yosemite. For most methods and problems, we used the Liberty subset for training. However, for the evaluation on the Brown dataset, we followed the convention by taking different combinations of training and testing subsets which will be detailed later. Next, we describe the methods we compared with and then present the results and analysis in detail.

4.1. Competing methods

We compare the proposed DeepCD method with a set of state-of-the-art deep-learning-based descriptor methods. Note that some deep-learning descriptor methods learn both feature embeddings and similarity metrics. As discussed in the introduction, the learned metrics often incur very expensive computation cost. Thus, we only utilize their embeddings for comparisons. (1) **Deepcompare** [40]. The paper explores several deep architectures for embedding and metrics. We compare with the embedding model $siam-2st-l2$ with 2-stream and concatenation fusion. (2) **Deepdesc** [29]. The method performs stochastic sampling and aggressive mining when training CNNs for embedding patch representations. We used their pre-trained models. Note that, unlike other methods, the training was done with two subsets of the Brown dataset rather than one. (3) **Global loss** [17]. This method explores the similar network structures as Deepcompare but enhances performance by optimizing global loss over the whole dataset instead of summing individual loss within sample pairs. They provide $TNet$ as the embedding method and $CS\ SNet\ Gloss$ as the decision network for the similarity metric. Only $TNet$ was used in the experiments. (4) **PNNet** [2] and **TFeat** [3]. These two methods are related to each other and are most related to our method. We choose PNNet 128-d embedding as the leading descriptor because of its great performance. By using the original data from the papers and the pre-trained models provided by the authors, we compare with four models in this paper: $PNNet-128dim$, $PNNet-256dim$, $TFeat-ratio^*$, $TFeat-margin^*$. Note that, in practice, the real-valued descriptors often store a real number with only one byte.

4.2. Local image patch benchmark

In this evaluation, we used the standard benchmark, the *Brown dataset* [5]. The dataset includes more than $450,000$ patches cropped from real world image using the DoG detector [22]. The patches are of two sizes, 32×32 and 64×64. They are normalized in both the scale and the orientation. As mentioned above, there are three subsets in this dataset, *Liberty*, *Notredame*, and *Yosemite*. We followed the convention by alternatively using one of them for training the model, and the other two for testing. As previous papers, the results were evaluated with FPR95, the false positive rate under 95% recall.

Table 1 reports results of all competing real-valued descriptors on all six combinations and the average. The top performer of each setting is highlighted in bold. All proposed DeepCD models outperform all other competing methods. Not surprisingly, the 2-stream model performs better than the splitting model since the 2-stream model offers more flexibility. Note that both our models augment $PNNet-128dim$ with a 32-byte binary complementary descriptor. The leading and complementary descriptors can be binary or real-valued. We evaluated different combinations and found that a real-valued leading descriptor with a binary complementary descriptor works the best. The performance is improved significantly from 7.26 to 6.17 (splitting) and 5.69 (2-stream) with a minor space increase.

An interesting question to answer is whether DeepCD is a more effective way to improve performance by increasing the descriptor's size. For answering the question, we performed two experiments. First, we simply increase the dimensionality of PNNet to 256. Table 1 shows that FPR95 only improves mildly to 6.98 even the space is doubled. We have also tried to combine two PNNets by using compact blinear pooling (CBP) [10], which has been shown effective on combining two information sources for the applications of VQA [9] and fine-grained recognition [21]. The results are displayed in the column named "2-stream PNNet+CBP." It is obvious that the combination with CBP is not effective. Both experiments show that it is not a trivial task to improve the descriptor's performance by increasing space or combining information. The success of DeepCD is attributed to its design with a complementary descriptor focusing on helping the leading descriptor and the joint optimization using the loss function with late fusion. By adding the DDM layer, the performance can be further improved. To show that the improvement achieved by DeepCD does not result from simply increasing the size of the network, we compare it with variants of PNNet of similar network sizes in the supplementary material.

It is also possible to use DeepCD methods to generate a pair of binary descriptors. For achieving this, the leading descriptor is turned binary by adding a fully connected layer and a sigmoid transfer layer at the end of its pipeline in the same way we did for the complementary descriptor. We compare our binary descriptor with the state-of-the-art binary descriptors. (1) **BinBoost** [33]. The method adopts an AdaBoost-like method for training binary descriptors using positive and negative patch pairs in the gradient domain. (2) **CovOpt** [30]. It uses convex optimization for learning the robust region of the image patch. (3) **Deepbits** [20]. By using an unsupervised method, it learns a descriptor in the binary form while keeping the balance among quantization loss, the invariant property and even distributions.

Table 2 compares these methods with our methods. It is clear that the proposed DeepCD descriptors outperform other competing methods by a margin. Note that BinBoost uses only 64 bits but achieves a decent performance. Unfortunately, as reported in the original paper, its performance nearly saturates after 64 bits because increasing the bit number to 256 just reduces FPR95 from 19.24% to 19.0%. Deepbits does not perform as well as others since it is an unsupervised method. Our binary descriptors can even compete with the real-valued descriptors by using 512+256 bits, less than 128 bytes used by many real-valued descriptors. The main advantage of the binary descriptor is that the distance between descriptors can be computed using the very efficient Hamming distance. Thus, descriptor matching can be performed much faster.

Note that, although decision network methods, Deepcompare 2ch-2stream [40] and CS SNet Gloss [17], reported better performances on Brown dataset, as mentioned in Section 1, they have extremely high computation costs during the testing stage, making them less practical to be used. In addition, their descriptors require 768 bytes per patch, $3 \sim 4$ times larger than others.

4.3. Wide baseline matching evaluation

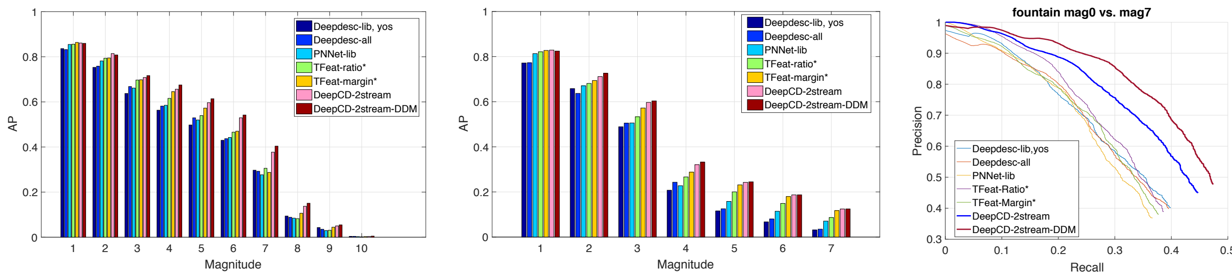
For studying how well our descriptors perform with perspective transformations, we compared our descriptors with others for the application of wide baseline matching on the *Strecha dataset* [32]. The dataset contains two image sequences with large perspective changes. The *fountain* sequence contains 11 images of the same scene from different perspectives while the *herzjesu* sequence contains 8 perspectives. The ground truth depth maps are provided with the dataset. We sampled 5,000 points randomly and used the ground truth depth maps for determining correspondences. We extract 64×64 patches densely over the images for matching. The patches were normalized using the mean and the standard deviation of the intensity values of all sampled patches per image. Descriptors were then extracted for patches. For each image pair, matches were determined by finding the closest patches in the other image. Only matches passing the left-right consistency check are considered valid. All valid matches were then sorted by similarity. A PR curve was computed and the mAP value was calculated for each descriptor.

Table 3 reports mAP values for Deepdesc, PNNet, TFeat and DeepCD descriptors. All models were trained on the liberty subset of the Brown dataset except Deepdesc. We used the pre-trained Deepdesc models released by the authors. The models were trained on either two subsets or all three. With more training data, Deepdesc seems to perform better. But, even with all three subsets for training, its performance is still the worst. The proposed DeepCD model outperforms all competing descriptors. Since the 2-stream model is generally better than splitting, we only report the results of DeepCD 2-stream in the following experiments.

To study how robust the descriptors are against the perspective changes, we report the average precision (AP) along with different magnitudes of perspective changes for both sequences in Figure 2. In each sequence, the first im-

	Deepdesc Lib, Yose	Deepdesc All	PNNet	TFeat Ratio*	TFeat Margin*	DeepCD 2-stream	DeepCD 2-stream (DDM)
Fountain	0.4155	0.4226	0.4239	0.4388	0.4486	0.4733	**0.4829**
Herzjesu	0.3343	0.3425	0.3658	0.3912	0.4157	0.4303	**0.4347**
mean	0.3749	0.3826	0.3948	0.4150	0.4321	0.4518	**0.4588**

Table 3. Comparisons of descriptors on the Strecha dataset using mAP.



(a) (b) (c)

Figure 2. Comparisons of descriptors on wide baseline matching for the two sequences in the Strecha dataset, fountain (a) and herzjesu (b). For each sequence, we show the AP values of all descriptors along with the magnitude of view change. The proposed DeepCD method provides more significant improvement for more challenging cases with larger view changes. (c) The precision-recall curve of different methods on the case (mag0 vs. mag7 on fountain) with a wide baseline.

age (the rightmost view) serves as the base image that other images were matched against. The last image (the leftmost view) sees the largest view change. From Figure 2, the proposed DeepCD descriptor offers more improvement for the more challenging cases with larger perspective changes. Figure 2 also shows the precision-recall (PR) curve for an example of matching between images with a large perspective change. It is obvious that DeepCD-stream outperforms other methods by a margin and the performance can be further boosted by adding the DDM layer.

4.4. Local descriptor performance evaluation

In this experiment, we tested how the descriptors perform under a variety of transformations using the *Oxford dataset* [23]. The dataset contains 48 images in 8 sequences with different variations and magnitudes, including blurring, compression, viewpoint changes, zoom, lighting changes and others. The ground truth homographies are given. We took 1,000 points per image by using DoG affine approximated detector [34]. The correspondences were sorted using the distance ratios between the best match and the second best match. The PR curves were formed and mAP values were calculated. In addition to the CNN-based descriptors, we have also included two hand-crafted descriptors, SIFT and ASV-SIFT [37] which aggregates information across different scales.

Table 4 summarizes mAP values for all competing descriptors. Again, DeepCD descriptors perform the best among all methods. Interestingly, the hand-crafted ASV-SIFT descriptor performs quite well with its performance better than Deepdesc and similar to PNNet. It is because ASV-SIFT explores information across multiple scales while all competing CNN-based descriptors including ours only use information at a fixed scale. CNN-based descriptors could be further improved by exploring multi-scale information.

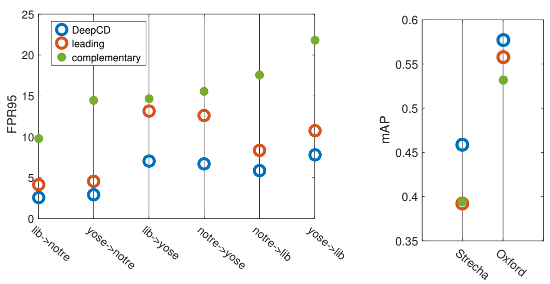
4.5. Discussions

To investigate how the complementary descriptor improves the performance, we conducted a couple of experiments. We first examine the matching performances of individual descriptors and their combination on the Brown dataset. Figure 3(a) shows the performances of descriptors for different training-testing configurations. In general, the leading descriptor's performances are very good, close to PNNet's, showing that it is a good descriptor alone. The complementary descriptor alone generally does not perform well. However, their combination improves the performance. In some cases, the performance boost are quite significant, such as lib->yose. Similar observations can be made in Figure 3(b) for Strecha and Oxford datasets.

Next, we show a more detailed analysis for complementary effects of our descriptors on lib->notre. In Figure 4(a), each point represents a pair of patches. Blue points are positive matches (the patches are correspondences) and red points are negative matches (they are not). Each point is plotted according to the distances in the feature space of the leading descriptor (y-axis) and the complementary one (x-axis). The distances are normalized within the range $[0, 1]$. When using only the leading descriptor and taking the threshold for 80% recall (the horizontal line in Figure 4(a), some points are mis-classified (the blue points

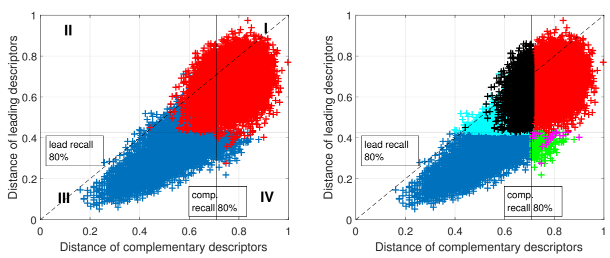
	SIFT	ASV-SIFT	Deepdesc Lib, Yose	Deepdesc All	PNNet	TFeat Ratio*	TFeat Margin*	DeepCD 2-stream	DeepCD 2-stream (DDM)
mAP	0.4766	0.5522	0.5267	0.5299	0.5541	0.5614	0.5594	0.5726	**0.5773**

Table 4. Comparisons of descriptors on the Oxford dataset using mAP.



(a) (b)

Figure 3. Performances of individual descriptors and their combination on (a) Brown and (b) Strehca and Oxford datasets. In (a), several training-testing configurations are shown. For example, lib->notre represents the one in which Liberty was used for training and Notredame for testing.



(a) (b)

Figure 4. Analysis of complementary effects. (a) The plot of patch matches according to the distances in the feature space of the leading descriptor (y-axis) and the complementary descriptor (x-axis). (b) Matches are colored for better reference. For example, the cyan points represents mis-classified matches by the leading descriptor.

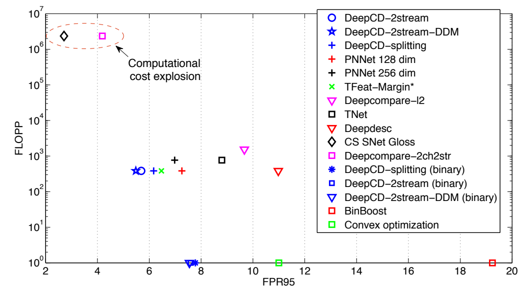


Figure 5. Performance-computation tradeoffs of different methods on the Brown dataset.

above the line and the red points below). The vertical line at 80% recall is similarly drawn for the complementary descriptor. This time, mis-classified points are blue points on the right and red ones on the left. It is clear that the complementary descriptors made more mistakes than the leading one, leading to a worse performance.

To better refer points in Figure 4(a), we label them with different colors in Figure 4(b). In the 2nd quadrant, the cyan points indicate the mis-classified matches by the leading descriptor. They cannot be distinguished from the (correctly classified) black points since their distance distributions overlap on the y-axis. However, it is clear that the cyan group and the black group have more distinguishable distributions vertically on the x-axis. Thus, they can be better separated vertically using the complementary descriptor. Similarly, in the 4th quadrant, the green points and magenta points cannot be separated vertically using the complementary descriptor, but they can be better separated horizontally using the leading descriptor. It shows that the descriptors are complementary and help each other in the 2nd and 4th quadrants. Thus, their combination boosts the performance.

Figure 5 plots tradeoffs between the matching performance and the computation cost for different methods. Since floating point operations are much more expensive than bit operations, we use floating point operations per pair (FLOPP) as the computation cost. Although learned metrics such as global loss [17] have great performance, their computation costs during the testing stage are nearly 10^4 times higher than the descriptor embedding methods. The detailed analysis is given in the supplementary material.

5. Conclusions

In this paper, we present the DeepCD framework which learns two complementary descriptors jointly and an instance of it based on the PNNet. The proposed DDM layer adaptively adjusts learning rates in accordance with training samples and further encourages the complementary descriptor to emphasize on correcting mistakes made by the leading one. Experiments show that the proposed framework is effective in improving the performance of the patch descriptors for various applications and transformations. There are several research directions worth of exploring. We would like to try other architectures, for example, employing very different structures for the leading descriptor and the complementary descriptor. A possible extension is to learn more than two descriptors jointly. Finally, we plan to extend the idea of joint complementary learning to other domains.

Acknowledgement. This work was supported by Ministry of Science and Technology (MOST) under grants 105-2221-E-001-030-MY2, 105-2218-E-001-006, 105-2218-E-002-032 and 105-2218-E-002-011. The work was also supported by the grants from NVIDIA.

References

[1] B. Amos, B. Ludwiczuk, and M. Satyanarayanan. OpenFace : A General-Purpose Face Recognition Library with Mobile Applications. 2015. 2.1

[2] V. Balntas, E. Johns, L. Tang, and K. Mikolajczyk. PN-Net: Conjoined Triple Deep Network for Learning Local Image Descriptors. *arXiv*, 2016. 2.1, 3, 3.1, 1, 3.2, 3.3, 4.1

[3] V. Balntas, E. Riba, D. Ponsa, and K. Mikolajczyk. Learning Local Feature Descriptors with Triplets and Shallow Convolutional Neural Networks. In *BMVC*, 2016. 2.1, 3.3, 4.1

[4] V. Balntas, L. Tang, and K. Mikolajczyk. BOLD - Binary Online Learned Descriptor For Efficient Image Matching. In *CVPR*, 2015. 1

[5] G. H. M. Brown and S. Winder. Discriminative Learning of Local Image Descriptors. *TPAMI*, 2011. 4, 4.2

[6] M. Calonder, V. Lepetit, C. Strecha, and P. Fua. BRIEF: Binary Robust Independent Elementary Features. In *ECCV*, 2010. 1

[7] H.-Y. Chen, Y.-Y. Lin, and B.-Y. Chen. Co-Segmentation Guided Hough Transform for Robust Feature Matching. *TPAMI*, 2015. 1

[8] P. Fischer, A. Dosovitskiy, and T. Brox. Descriptor Matching with Convolutional Neural Networks: A Comparison to SIFT. *arXiv*, 2014. 2.1

[9] A. Fukui, D. H. Park, D. Yang, A. Rohrbach, T. Darrell, and M. Rohrbach. Multimodal Compact Bilinear Pooling for Visual Question Answering and Visual Grounding. *arXiv*, 2016. 2.2, 4.2

[10] Y. Gao, O. Beijbom, N. Zhang, and T. Darrell. Compact Bilinear Pooling. In *CVPR*, 2016. 2.2, 4.2

[11] X. Han. MatchNet : Unifying Feature and Metric Learning for Patch-Based Matching. In *CVPR*, 2015. 2.1

[12] G. Hinton, O. Vinyals, and J. Dean. Distilling the Knowledge in a Neural Network. In *arXiv*, 2015. 2.2

[13] E. Hoffer and N. Ailon. Deep Metric Learning Using Triplet Network. In *ICLRW*, 2015. 2.1

[14] K.-J. Hsu, Y.-Y. Lin, and Y.-Y. Chuang. Robust Image Alignment with Multiple Feature Descriptors and Matching-Guided Neighborhoods. In *CVPR*, 2015. 1

[15] Y.-T. Hu and Y.-Y. Lin. Progressive Feature Matching with Alternate Descriptor Selection and Correspondence Enrichment. In *CVPR*, 2016. 1

[16] M. Jaderberg, K. Simonyan, A. Zisserman, and K. Kavukcuoglu. Spatial Transformer Networks. In *NIPS*, 2015. 3.3

[17] B. G. V. Kumar, 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, 2.1, 4.1, 4.2, 4.5

[18] S. Lee, S. Purushwalkam, M. Cogswell, V. Ranjan, D. J. Crandall, and D. Batra. Stochastic Multiple Choice Learning for Training Diverse Deep Ensembles. In *NIPS*, 2016. 3.3

[19] S. Leutenegger, M. Chli, and R. Y. Siegwart. BRISK: Binary Robust Invariant Scalable Keypoints. In *ICCV*, 2011. 1

[20] K. Lin, J. Lu, C.-S. Chen, and J. Zhou. Learning Compact Binary Descriptors with Unsupervised Deep Neural Networks. In *CVPR*, 2016. 4.2

[21] T.-Y. Lin, A. RoyChowdhury, and S. Maji. Bilinear CNN Models for Fine-Grained Visual Recognition. In *ICCV*, 2015. 2.2, 4.2

[22] D. Lowe. Distinctive Image Features from Scale-Invariant Keypoints. *IJCV*, 2004. 1, 2, 4.2

[23] K. Mikolajczyk and C. Schmid. A Performance Evaluation of Local Descriptors. *TPAMI*, 2003. 4, 4.4

[24] M. Norouzi, A. Punjanio, and D. Fleet. Fast Exact Search in Hamming Space with Multi-Index Hashing. *TPAMI*, 2014. 3.4

[25] E.-J. Ong and M. Bober. Improved Hamming Distance Search using Variable Length Hashing. In *CVPR*, 2016. 3.4

[26] A. Romero, N. Ballas, S. E. Kahou, A. Chassang, C. Gatta, and Y. Bengio. FitNets: Hints For Thin Deep Nets. In *ICLR*, 2015. 2.2

[27] E. Rublee, V. Rabaud, K. Konolige, and G. Bradski. ORB: An Efficient Alternative to SIFT or SURF. In *ICCV*, 2011. 1

[28] F. Schroff, D. Kalenichenko, and J. Philbin. FaceNet: A Unified Embedding for Face Recognition and Clustering. In *CVPR*, 2015. 2.1

[29] 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. 2.1, 4.1

[30] K. Simonyan, A. Vedaldi, and A. Zisserman. Learning Local Feature Descriptors Using Convex Optimisation. *TPAMI*, 2013. 1, 2, 4.2

[31] H. O. Song, Y. Xiang, S. Jegelka, and S. Savarese. Deep Metric Learning via Lifted Structured Feature Embedding. In *CVPR*, 2016. 2.1

[32] C. Strecha, W. Hansen, L. V. Gool, P. Fua, and U. Thoennessen. On Benchmarking Camera Calibration and Multi-View Stereo for High Resolution Imagery. In *CVPR*, 2008. 4, 4.3

[33] T. Trzcinski, M. Christoudias, P. Fua, and V. Lepetit. Boosting Binary Keypoint Descriptors. In *CVPR*, 2013. 1, 4.2

[34] A. Vedaldi and B. Fulkerson. VLFeat - An Open and Portable Library of Computer Vision Algorithms. In *ACM MM*, 2010. 4.4

[35] L. Z. S. Wang, L. Tian, F. He, Z. Liu, and Q. Tian. Query-Adaptive Late Fusion for Image Search and Person Re-identification. In *CVPR*, 2015. 3

[36] Z. Wang, B. Fan, and F. Wu. Local Intensity Order Pattern for Feature Description. In *ICCV*, 2011. 1, 2

[37] T.-Y. Yang, Y.-Y. Lin, and Y.-Y. Chuang. Accumulated Stability Voting: A Robust Descriptor from Descriptors of Multiple Scales. In *CVPR*, 2016. 4.4

[38] K. M. Yi, E. Trulls, V. Lepetit, and P. Fua. LIFT: Learned Invariant Feature Transform. In *ECCV*, 2016. 2.1

[39] K. M. Yi, Y. Verdie, P. Fua, and V. Lepetit. Learning to Assign Orientations to Feature Points. In *CVPR*, 2016. 2.1

[40] S. Zagoruyko and N. Komodakis. Learning to Compare Image Patches via Convolutional Neural Networks. In *CVPR*, 2015. 1, 2.1, 4.1, 4.2

[41] B. Zhuang, G. Lin, C. Shen, and I. Reid. Fast Training of Triplet-based Deep Binary Embedding Networks. In *CVPR*, 2016. 2.1