# Local Descriptors Optimized for Average Precision

Kun He
Boston Univesity
hekun@bu.edu

Yan Lu*
Honda Research Institute USA
sinoluyan@gmail.com

Stan Sclaroff
Boston University
sclaroff@bu.edu

## Abstract

*Extraction of local feature descriptors is a vital stage in the solution pipelines for numerous computer vision tasks. Learning-based approaches improve performance in certain tasks, but still cannot replace handcrafted features in general. In this paper, we improve the learning of local feature descriptors by optimizing the performance of descriptor matching, which is a common stage that follows descriptor extraction in local feature based pipelines, and can be formulated as nearest neighbor retrieval. Specifically, we directly optimize a ranking-based retrieval performance metric, Average Precision, using deep neural networks. This general-purpose solution can also be viewed as a listwise learning to rank approach, which is advantageous compared to recent local ranking approaches. On standard benchmarks, descriptors learned with our formulation achieve state-of-the-art results in patch verification, patch retrieval, and image matching.*

## 1. Introduction

Extracting feature descriptors from local image patches is a common stage in many computer vision tasks involving alignment or matching. To replace handcrafted feature engineering, recently much attention has been paid to learning local feature descriptors. Despite exciting progress, certain levels of handcrafting are currently present in the design of learning objectives for local feature descriptors, making it difficult to have performance guarantees when the learned descriptors are integrated into larger pipelines. Indeed, according to a recent study [28], traditional handcrafted features such as SIFT [21] can still outperform learned ones in complicated tasks such as 3D reconstruction. In this paper, we aim to improve the learning of local feature descriptors by optimizing better objective functions.

Our thesis is that local feature descriptor learning is not a standalone problem, but rather a component in the optimization of larger pipelines. Therefore, the learning objectives should be designed in accordance with other pipeline components. Upon inspection of common local feature matching pipelines, we find that feature matching can be exactly formulated as nearest neighbor retrieval. Thus, we propose a novel listwise *learning to rank* formulation for learning local feature descriptors, based on the direct optimization of a ranking-based retrieval performance metric: Average Precision. Our formulation uses deep neural networks, and works for both binary and real-valued descriptors. Compared to recent approaches, our method optimizes a commonly adopted evaluation metric, and eliminates complex optimization heuristics. Descriptors learned with our formulation achieve state-of-the-art results in benchmarks including UBC Phototour [37], HPatches [2], RomePatches [26], and the Oxford dataset [23].

An important feature of our proposed formulation is that it is general-purpose, as it optimizes the performance of the task-independent nearest neighbor matching stage, rather than a task-specific pipeline. Nevertheless, to better tailor the learned descriptors for feature matching, we also augment our formulation with task-specific improvements. First, we make use of the Spatial Transformer module [12] to effectively handle geometric noise and improve the robustness of matching, without requesting extra supervision. Also, for the challenging HPatches dataset, we design a clustering-based technique to mine additional patch-level supervision, which improves the performance of learned descriptors in the image matching task.

In summary, we propose a general-purpose learning to rank formulation that optimizes local feature descriptors for nearest neighbor matching. Our learned descriptors achieve state-of-the-art performance, and are further enhanced by task-specific improvements. We believe that our contribution can serve as a stepping stone for the direct optimization of larger computer vision pipelines.

## 2. Related Work

### Learning Local Features

Parallel with the long history of handcrafted computer vision pipelines (the most prominent example being SIFT [21]), numerous researchers have attempted to replace

*Now with Nvidia.

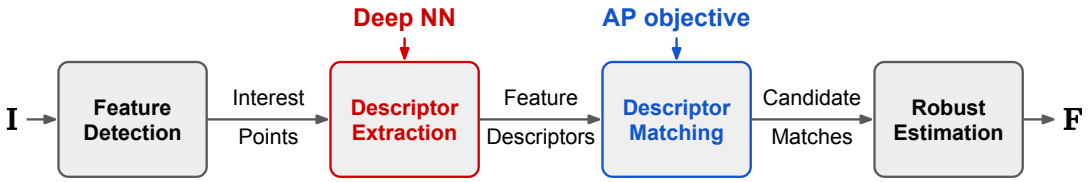


Figure 1. An example local feature-based image matching pipeline, where the task is to estimate the fundamental matrix $\mathbf{F}$ between images $\mathbf{I} = (I_1, I_2)$, using robust estimation techniques such as RANSAC [9]. We model the feature descriptor extractor using deep neural networks, and directly optimize a ranking-based objective (Average Precision) for the subsequent stage of descriptor matching.

handcrafted components with learned counterparts. There exist many formulations for learning different components in local feature based pipelines. For example, interest point detectors are learned in [18, 27, 35], LIFT [39] learns three components separately in a feature matching pipeline, and DSAC [4] approximately learns a camera localization pipeline end-to-end.

For learning local feature descriptors, some early works use simple architectures [33, 37] and convex optimization [31]. Later approaches use deep neural networks: PhilippNet [8] learns by fitting pseudo-classes, DeepDesc [30] applies Siamese networks, MatchNet [10] and DeepCompare [40] learn nonlinear distance metrics for matching, and [26] uses Convolutional Kernel Networks. A series of recent works have considered more advanced model architectures and triplet-based deep metric learning formulations, including UCN [7], TFeat [3], GLoss [15], L2Net [32], HardNet [24], and GOR [41].

Instead of optimizing triplet-based surrogate losses, we employ listwise learning to rank to directly optimize the performance of the matching stage. Although end-to-end optimization of the pipeline is attractive, it is unfortunately highly difficult and task-dependent. By focusing on the two task-independent stages (descriptor extraction and matching), our solution is general-purpose and can be potentially integrated into larger optimization pipelines.

**Evaluating Local Feature Descriptors**

Local features ideally should be evaluated in terms of final task performance, *e.g*. Mikolajczyk and Schmid [23] use precision and recall derived from image matching, and Schonberger *et al*. [28] use a benchmark based on 3D reconstruction. However, in complex vision pipelines, final task performance can be affected by individual components. For example, [2] observes that without controlling for components such as interest point detection in image-based benchmarks, different conclusions can be drawn when comparing the relative performance of feature descriptors.

Patch-based benchmarks provide unambiguous evaluation for local feature descriptors. The *patch verification* task is first proposed in [37], formulated as binary classificaton on the relationship between patch pairs. RomePatches [26] and HPatches [2] both consider the *patch retrieval* task, which simulates nearest neighbor matching, and is shown [2] to be more realistic and challenging compared to patch verification. A ranking-based evaluation metric, Average Precision, is adopted in both benchmarks.

**Ranking Optimization in Metric Learning**

Metric learning [14] is a general family of methods that learn distance functions from data. While much previous effort focused on learning Mahalanobis distances, recently the metric learning community has focused on learning vector embeddings to be used with standard (*e.g*. Euclidean) distance metrics. In this light, the problem of learning local feature descriptors is an instance of metric learning.

Learning vector embeddings necessarily calls for task-dependent formulations. For nearest neighbor retrieval, optimization of ranking performance has been studied in metric learning. For example, learning to rank formulations for Mahalanobis distances are proposed in [19, 22]. Triplet-based deep metric learning approaches [16, 25, 34, 38] can also be viewed as optimizing surrogate ranking losses. In the "learning to hash" subcommunity that considers the special case of learning binary embeddings, He *et al*. [11] directly optimize ranking-based retrieval performance measures with deep neural networks, based on an approximation to histogram binning originally proposed in [34], which is also adopted in learning binary descriptors by [5]. We make use of their optimization technique in the learning of binary and real-valued descriptors for our problem.

## 3. Optimizing Descriptors for Matching

In this section, we motivate our approach by analyzing the descriptor matching stage, and point out that it corresponds to nearest neighbor retrieval. Then we discuss a learning to rank formulation to optimize ranking-based retrieval performance.

### 3.1. Nearest Neighbor Matching

Consider Fig. 1, which depicts a pipeline for estimating the fundamental matrix between matching images $I_1$ and $I_2$. It consists of four stages: feature detection, descriptor extraction, descriptor matching, and robust estimation. Suppose we detect and extract $M$ local features from each image. The descriptor matching stage operates as follows:

it computes the pairwise distance matrix with $M^2$ entries, and for each feature in $I_1$, looks for its nearest neighbor in $I_2$, and vice versa. Feature pairs that are mutual nearest neighbors[1] become candidate matches in the robust estimation stage, such as RANSAC [9].

We point out that this matching process is exactly performing nearest neighbor retrieval: each feature in $I_1$ is used to query a database, which is the set of features in $I_2$. For good performance, true matches should be returned as top retrievals, while false matches are ranked as low as possible. Performance of the matching stage also directly reflects the quality of the learned descriptors, since it has no learnable parameters (only performs distance computation and sorting). To assess nearest neighbor matching performance, we adopt Average Precision (AP), a commonly used evaluation metric. AP evaluates the performance of retrieval systems under the *binary relevance* assumption: retrievals are either "relevant" or "irrelevant" to the query. This naturally fits the local feature matching setup, where given a reference feature, features in a target image are either its true match or false match. Next, we learn binary and real-valued local feature descriptors to optimize AP.

### 3.2. Optimizing Average Precision

We first introduce mathematical notation. Let $\mathcal{X}$ be the space of image patches, and $S \subset \mathcal{X}$ be a database. For a query patch $q \in \mathcal{X}$, let $S_q^+$ be the set of its matching patches in $S$, and let $S_q^-$ be the set of non-matching patches. Given a distance metric $D$, let $(x_1, x_2, \ldots, x_n)$ be a ranking of items in $S_q^+ \cup S_q^-$ sorted by increasing distance to $q$, *i.e.* $D(x_1, q) \leq D(x_2, q) \ldots \leq D(x_n, q)$. Given the ranking, AP is the average of precision values ($Prec@K$) evaluated at different positions:

$$Prec@K = \frac{1}{K} \sum_{i=1}^{K} \mathbf{1}[x_i \in S_q^+], \tag{1}$$

$$AP = \frac{1}{|S_q^+|} \sum_{K=1}^{n} \mathbf{1}[x_K \in S_q^+] Prec@K, \tag{2}$$

where $\mathbf{1}[\cdot]$ is the binary indicator. AP achieves its optimal value if and only if every patch from $S_q^+$ is ranked above all patches from $S_q^-$.

The optimization of AP can be cast as a metric learning problem, where the goal is to learn a distance metric $D$ that gives optimal AP when used for retrieval. Ideally, if all the above steps can be formulated in differentiable forms, then AP can be optimized by exploiting chain rule. However, this is not possible in general: the sorting operation, required in producing the ranking, is non-differentiable, and continuous changes in the input distances induce discontinuous "jumps" in the value of AP. Thus, appropriate smoothing is necessary to derive differentiable approximations of AP.

[1]For simplicity, the distance ratio check [21] is not considered.

Our solution is based on a recent result in the metric learning community. For the problem of learning binary image-level descriptors for image retrieval, He *et al.* [11] observe that sorting on integer-valued Hamming distances can be implemented as histogram binning, and employ a differentiable approximation to histogram binning [34] to optimize ranking-based objectives with gradient descent. We use this optimization framework to optimize AP for both binary and real-valued local feature descriptors. In the latter case, the optimization is enabled by a novel quantization-based approximation that we develop.

**Binary Descriptors**

Binary descriptors offer compact storage and fast matching, which are useful in applications with speed or storage restrictions. Although binary descriptors can be learned one bit at a time [33], here we take a gradient-based relaxation approach to learn fixed-length "hash codes".

Formally, a deep neural network $F$ is used to model a mapping from patches to a low-dimensional Hamming space: $F : \mathcal{X} \rightarrow \{-1, 1\}^b$. For the Hamming distance $D$, which takes integer values in $\{0, 1, \ldots, b\}$, AP can be computed in closed form using entries of a histogram $\mathbf{h}^+ = (h_0^+, \ldots, h_b^+)$, where $h_k^+ = \sum_{x \in S_q^+} \mathbf{1}[D(q, x) = k]$. The closed-form AP can further be continuously relaxed, and differentiated with respect to $\mathbf{h}^+$ [11].

The next step in the chain rule is to differentiate entries of $\mathbf{h}^+$ with respect to the network $F$. Usnitova and Lempitsky [34] approximate the histogram binning operation as

$$h_k^+ \approx \sum_{x \in S_q^+} \delta(D(q, x), k), \tag{3}$$

replacing the binary indicator with a differentiable function $\delta$ that peaks when $D(q, x) = k$. This allows to derive approximate gradients as

$$\frac{\partial h_k^+}{\partial F(q)} \approx \sum_{x \in S_q^+} \frac{\partial \delta(D(q, x), k)}{\partial D(q, x)} \frac{\partial D(q, x)}{\partial F(q)}, \tag{4}$$

$$\frac{\partial h_k^+}{\partial F(x)} \approx \mathbf{1}[x \in S_q^+] \frac{\partial \delta(D(q, x), k)}{\partial D(q, x)} \frac{\partial D(q, x)}{\partial F(x)}. \tag{5}$$

Note that the partial derivative of the Hamming distance is obtained via this differentiable formulation:

$$D(x, x') = \frac{1}{2} \left( b - F(x)^\top F(x') \right). \tag{6}$$

Finally, the thresholding operation used to produce binary bits is smoothed using the $\tanh$ function,

$$F(x) = (\mathrm{sgn}(f_1(x)), \ldots, \mathrm{sgn}(f_b(x))) \tag{7}$$

$$\approx (\tanh(f_1(x)), \ldots, \tanh(f_b(x))), \tag{8}$$

where $f_i$ are real-valued neural network activations. With these relaxations, the network can be trained end-to-end.

**Real-Valued Descriptors**

To complete our formulation, we next consider real-valued descriptors, which are preferred in high-precision scenarios. We model the the descriptor as a vector of real-valued network activations, and apply $L_2$ normalization: $\|F(x)\| = 1, \forall x$. In this case, the Euclidean distance $D$ is given as

$$D(x, x') = \sqrt{2 - 2F(x)^\top F(x')}. \tag{9}$$

The main challenge in optimizing AP for real-valued descriptors is again the non-differentiable sorting, but real-valued sorting has no simple alternative form. However, histogram binning can be used as an approximation: we *quantize* real-valued distances using histogram binning, obtain the histograms $\mathbf{h}^+$, and then reduce the optimization problem to the previous one. With $L_2$-normalized vectors, the quantization is easy to implement as the Euclidean distance has closed range $[0, 2]$: we simply uniformly divide $[0, 2]$ into $b + 1$ bins. To derive the chain rules in this case, only the partial derivatives of the distance function needs modification in (4) and (5). The differentiation rules for the $L_2$ normalization operation are well known, and we give full derivations in the supplementary material.

Differently from the case of binary descriptors, the number of histogram bins $b$ is now a free parameter, which involves a tradeoff. On the one hand, a large $b$ reduces quantization error, which in fact achieves zero if each histogram bin contains at most one item. On the other hand, gradient computation for approximate histogram binning has linear complexity in $b$. Nevertheless, in our experiments, we consistently obtain good results using $b \leq 25$.

### 3.3. Comparison with Other Ranking Approaches

We would like to contrast our approach with others in the learning to rank context. Some recent methods, *e.g.* [3,24,32,41], learn feature descriptors by optimizing losses defined on triplets in the form of $(a, p^+, p^-)$, where $a$ is an anchor patch, $p^+$ is its matching patch, and $p^-$ is a non-matching patch. The loss typically encourages the learned distance metric $D$ to satisfy $D(a, p^+) < D(a, p^-) - \rho$, where $\rho$ is a margin. Triplet losses have a long history in metric learning [6, 29], and are better suited for ranking tasks than pair-based losses used in Siamese networks (*e.g.* [30]). In learning to rank terminology [20], triplets define local *pairwise* ranking losses, while our approach is *listwise* since the evaluation metric that we optimize (AP) is defined on a ranked list.

Despite their simplicity, triplet losses can be very challenging to optimize. For $N$ training examples, the set of triplets is of size $O(N^3)$, but most of them get classified correctly early on during learning. To maintain stable progress, carefully tuned heuristics such as hard negative mining [24], anchor swap [3], or distance-weighted sampling [38] are

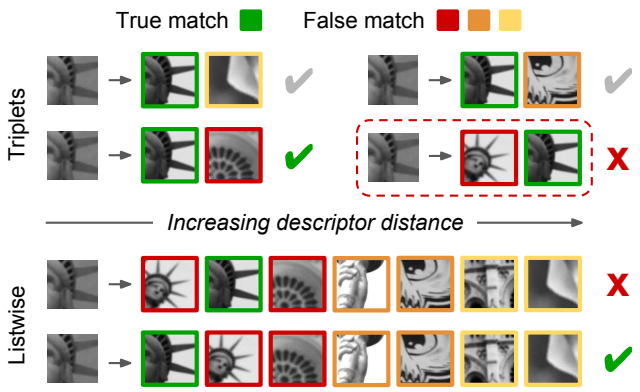


Figure 2. Comparison between triplet-based and listwise ranking approaches. Top: in triplet-based training, most triplets get correctly classified early (first row), and it is crucial to find and correct high-rank errors (red dashed box), with a heuristic known as hard negative mining. Bottom: in listwise ranking which is *position-sensitive*, the high-rank error would reduce AP from 1 to 0.5, thus automatically receiving a heavy penalty. Our listwise optimization corrects such errors without using complex mining heuristics. Best viewed in color.

crucial. We note that these optimization difficulties stem from a fundamental mismatch between triplet losses and listwise evaluation. As shown in Fig. 2, listwise metrics are *position-sensitive*, while local losses are insensitive; an error made on a single triplet may have a big impact on the result if it occurs near the top of the list. Therefore, heuristics are needed to focus on reducing high-rank errors. In contrast, our method directly optimizes the listwise evaluation metric, Average Precision, and is free of such heuristics. The listwise optimization also implicitly encodes hard negative mining: it requires matching patches to be ranked above all non-matching patches, which automatically enforces correct classification of the hardest triplet in the batch without explicitly finding it.

## 4. Task-Specific Improvements

In addition to the general-purpose learning to rank formulation, we develop two improvements that take the nature of local feature matching into account.

### 4.1. Handling Geometric Noise

To improve the robustness of local features for matching, it is key to build invariance to geometric noise into the descriptor: SIFT [21] estimates orientation and affine shape to normalize input patches, and LIFT [39] includes a learned orientation estimation module. Likewise, we can also include a geometric alignment module in our descriptor networks. Our choice is the Spatial Transformer [12], which aligns input patches by predicting a 6-DOF affine transformation, without requiring extra supervision. In our exper-

iments, this module is able to correct geometric distortion, and consistently improve performance.

In contrast to the image-based UCN [7], which also includes Spatial Transformers, our patch-based networks have limited input size, and the predicted affine transformation can often lead to out-of-boundary sampling, which corrupts sampled patches. We address this challenge by using appropriate boundary padding. Details are given in the supplementary material.

### 4.2. Label Mining for Image Matching

While our formulation directly optimizes for the task of *patch retrieval*, it is also possible to address higher-level tasks. We demonstrate this with the *image matching* task in the challenging HPatches dataset [2], which contains patches extracted from matching image sequences.

The image matching task in HPatches is formulated similarly as patch retrieval, which involves retrieving matching patches in a pool of "distractors". However, the distractors are defined differently. In patch retrieval, distractors do not include patches in the same image sequence as the query, due to concern of repeating structures in images. In image matching, images are matched against others in the same sequence, which means that all distractors are actually in-sequence. Thus, image matching performance can be improved by including in-sequence distractors when optimizing patch retrieval.

We perform *label mining* to augment the set of distractors when optimizing patch retrieval in HPatches. To avoid noisy labels in the presence of repeating structures, we use a simple heuristic: clustering. For each image sequence, we cluster all patches based on visual appearance. Then, patches having high inter-cluster distance are marked as distractors for each other (with 3D verification). Note that label mining is not related to the hard negative mining heuristic, since its goal is to add additional supervision. Please see Sec. 5.2 and supplementary material for more details.

## 5. Experiments

We experiment with three patch-based datasets (examples are in Fig. 3): UBC Phototour [37], HPatches [2], and RomePatches [26]. We use the CNN architecture recently proposed in L2Net [32], which consists of seven convolution layers, and is regularized with Batch Normalization and Dropout. We do not use the more complex "Center Surround" architecture. The input to the network is 32x32 grayscale, and we resize input patches to this size. When adding the Spatial Transformer module, we increase the input size to 42x42, and use 3 convolution layers to predict a 6-DOF affine transformation, which is then used to sample a 32x32 patch.

We name our descriptor DOAP (**D**escriptors **O**ptimized for **A**verage **P**recision), and test its binary and real-valued versions. Our networks are trained using SGD with momentum 0.9 and weight decay $10^{-4}$, and the learning rate is decayed linearly to zero within a fixed number of epochs. The initial learning rate (always on the order of 0.1) and number of epochs are tuned during training. Input normalization is as follows: patches are normalized by subtracting the mean pixel value in the patch and then dividing by the standard deviation.

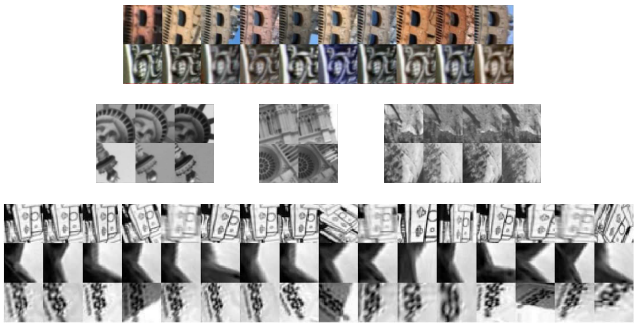

Figure 3. Examples from three patch-based datasets (top to bottom): RomePatches [26], UBC Phototour [37], HPatches [2]. In all datasets, patches are grouped such that patches in the same group correspond to the same 3D point.

### 5.1. UBC Phototour

We first conduct experiments on the UBC Phototour dataset [37], a classical benchmark of descriptor performance. Patches are extracted from Difference-of-Gaussian detections in three image sequences: *Liberty*, *Notre Dame*, and *Yosemite*. Following the standard setup, we use six training/test combinations formed by the three sequences, and report patch verification performance in terms of false positive rate at 95% recall (FPR95).

We train our models on UBC Phototour with data augmentation, in the form of random flipping and 90-degree rotations, which showed consistent performance improvement in previous work. We compare to a range of existing descriptors, including both binary and real-valued, listed in Table 1. L2Net [32] and HardNet [24] are two leading methods, which optimize triplet-based losses with the same CNN architecture as ours. We also include methods that use the "Center Surround" architecture: CS-SNet-Gloss [15] and CS-L2Net, and we have applied the recent global regularization technique in [41] to HardNet, resulting in a more competitive method which we call HardNet-GOR. Compared to existing approaches, DOAP achieves state-of-the-art performance with both binary and real-valued descriptors, and results are further improved by DOAP-ST, which includes the Spatial Transformer module.

We attribute the performance of DOAP and DOAP-ST to the listwise AP optimization. As mentioned in Sec. 3.3, listwise optimization automatically includes the "hard negative mining" heuristic in local ranking approaches, since it

| Method | Train | Notredame | Yosemite | Liberty | Yosemite | Liberty | Notredame | FPR95 |
|---|---|---|---|---|---|---|---|---|
| | Test | Liberty | | Notredame | | Yosemite | | Mean |
| *Real-valued descriptors* | | | | | | | | |
| SIFT [21] | 128 | 29.84 | | 22.53 | | 27.29 | | 26.55 |
| MatchNet [10] | 128 | 7.04 | 11.47 | 3.82 | 5.65 | 11.6 | 8.70 | 8.05 |
| TFeat-M* [3] | 128 | 7.39 | 10.31 | 3.06 | 3.80 | 8.06 | 7.24 | 6.64 |
| TL-AS-GOR [41] | 128 | 4.80 | 6.45 | 1.95 | 2.38 | 5.40 | 5.15 | 4.36 |
| DC-2ch2st+ [40] | 512 | 4.85 | 7.20 | 1.90 | 2.11 | 5.00 | 8.39 | 4.19 |
| CS-SNet-GLoss+ [15] | 256 | 3.69 | 4.91 | 0.77 | 1.14 | 3.09 | 2.67 | 2.71 |
| L2Net+ [32] | 128 | 2.36 | 4.7 | 0.72 | 1.29 | 2.57 | 1.71 | 2.23 |
| HardNet+ [24] | 128 | 2.28 | 3.25 | 0.57 | 0.96 | 2.13 | 2.22 | 1.90 |
| HardNet-GOR+ [24, 41] | 128 | 1.89 | 3.03 | 0.54 | 0.90 | 2.41 | 2.39 | 1.86 |
| CS-L2Net+ [32] | 256 | 1.71 | 3.87 | 0.56 | 1.09 | 2.07 | 1.30 | 1.76 |
| DOAP+ | 128 | 1.54 | 2.62 | 0.43 | 0.87 | 2.00 | **1.21** | 1.45 |
| DOAP-ST+ | 128 | **1.47** | **2.29** | **0.39** | **0.78** | **1.98** | 1.35 | **1.38** |
| *Binary descriptors* | | | | | | | | |
| BinBoost [33] | 64 | 20.49 | 21.67 | 16.90 | 14.54 | 22.88 | 18.97 | 19.24 |
| L2Net+ [32] | 128 | 7.44 | 10.29 | 3.81 | 4.31 | 8.81 | 7.45 | 7.01 |
| CS-L2Net+ [32] | 256 | 4.01 | 6.65 | 1.90 | 2.51 | 5.61 | 4.04 | 4.12 |
| DOAP+ | 256 | 3.18 | 4.32 | 1.04 | **1.57** | 4.10 | 3.87 | 3.01 |
| DOAP-ST+ | 256 | **2.87** | **4.17** | **0.96** | 1.76 | **3.93** | **3.64** | **2.89** |

Table 1. Patch verification performance on UBC Phototour, where metric is false positive rate at 95% recall (FPR95). The best results are in **bold**. Second column shows dimensionality, and methods with suffix "+" are trained with data augmentation. Both the binary and real-valued versions of DOAP and DOAP-ST achieve state-of-the-art results.

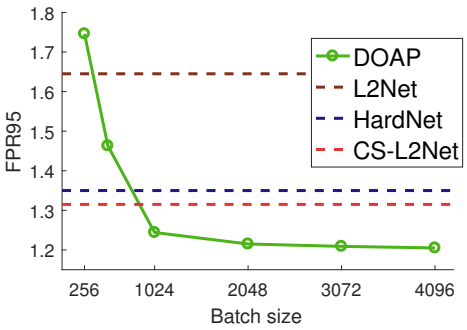


Figure 4. Influence of training batch size for the 128-d DOAP descriptor trained on *Liberty*, with data augmentation. Vertical axis: average of FPR95 on *Notre Dame* and *Yosemite*.

implicitly enforces the correct classification of all induced pairs and triplets. We then expect performance to improve when increasing training batch size, as larger batches lead to longer lists and increased likelihood of including hard negatives. We validate this by training the 128-dimensional DOAP model on *Liberty*, varying batch size between 256 and 4096, and monitoring the average of FPR95 on *Notre Dame* and *Yosemite*. Indeed, Fig. 4 shows that performance improves with batch size and saturates after 2048. Similar trends are also observed in HardNet [24], with saturation occurring at batch size 512. In contrast, the listwise optimization allows the performance of DOAP to saturate at a later stage.

### 5.2. HPatches

HPatches [2] consists of a total of over 2.5 million patches extracted from 116 image sequences, each with 6 images with known homography. Both viewpoint and illumination changes are included, and test cases have levels of difficulty *easy*, *hard*, and *tough*, according to the amount of geometric noise. Three evaluation tasks are considered (in increasing order of difficulty): patch verification, patch retrieval, and image matching.

In this experiment, we focus on comparing real-valued descriptors. We first include four baselines reported in [2]: SIFT [21], RootSIFT [1], DeepDesc [30], and TFeat [3]. Next, as results for L2Net and HardNet trained on the *Liberty* sequence of UBC Phototour are reported in [24], for fair comparison, we also report results for our models trained on *Liberty*. Finally, we train and evaluate three versions of our descriptor on HPatches: DOAP, DOAP-ST with the Spatial Transformer, and DOAP-ST-LM, which additionally uses label mining. We compare to the L2Net model trained on HPatches, and HardNet++, trained on the union of *Liberty* and HPatches. Note that CS-L2Net is excluded as it performs worse than L2Net in this more realistic dataset, which is consistent with the observations in [15,32]. When determining training/test sets, we use the "a" split: the test set contains 40 image sequences (20 viewpoint and 20 illumination), and the training set contains the other 76 sequences.

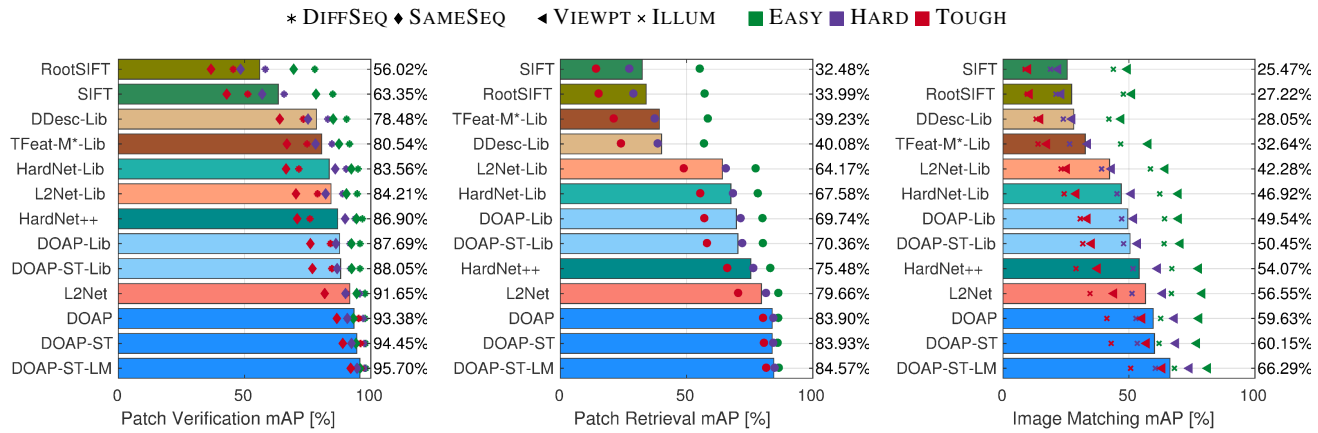


Figure 5. Results on the HPatches dataset, evaluated on the test set of the "a" split. No ZCA normalization [2] is used. Suffix indicates training set used (Lib: *Liberty*, no suffix: HPatches). HardNet++ is trained on the union of *Liberty* and HPatches. DOAP outperforms competing methods in all tasks, and all of its variants excel in handling tough test cases.

Fig. 5 presents results on HPatches.[2] Our descriptors achieve state-of-the-art results for all three tasks, and all variants are better at handling *tough* test cases than competing methods. Specifically, DOAP and DOAP-ST obtain the best patch retrieval performance, which directly results from the optimization of patch retrieval mAP. This optimization also gives state-of-the-art performance in patch verification. For the most challenging task of image matching, as mentioned in [2], patch retrieval performance is well correlated. However, due to the difference in task definition that we mentioned in Sec. 4.2, all methods see lower performance when tested for image matching. With the clustering-based label mining, DOAP-ST-LM significantly improves image matching mAP compared to the next best models: around 6% and 10% over DOAP-ST and L2Net, respectively. Notably, it achieves **over 50% mAP** even in the toughest test cases (*tough* geometric noise, illumination change). The inclusion of extra supervision also boosts patch retrieval performance, since in-sequence distractors provide harder negatives to learn from.

### 5.3. RomePatches

We next consider the RomePatches dataset [26], which contains 20,000 image patches of size 51x51, split equally into training and test sets. The task is patch retrieval. This dataset is constructed by performing SIFT matching on images taken in Rome, and keeping matching patches that satisfy 3D constraints. With such tailored construction, SIFT is unsurprisingly a strong baseline on RomePatches. In fact, in terms of test set mAP, previous methods, including pretrained AlexNet [13] and PhilippNet [8], could not surpass SIFT. The only method to do so was the CKN-grad variant proposed in [26], using 1024-dimensional descriptors.

[2] Results for L2Net and HardNet are obtained using their publicly released models and may slightly differ from those reported in [24].

| Method | Coverage | Dim. | Train | Test |
|---|---|---|---|---|
| SIFT [21] | 51x51 | 128 | 91.6 | 87.9 |
| AlexNet-conv3 [13] | 99x99 | 384 | 81.6 | 79.2 |
| PhilippNet [8] | 64x64 | 512 | 86.1 | 81.4 |
| CKN-grad [26] | 51x51 | 1024 | 92.5 | 88.1 |
| DOAP | 51x51 | 128 | **95.9** | **88.4** |
| Binary DOAP | 51x51 | 256 | 95.2 | 86.8 |

Table 2. Patch retrieval mAP comparison on RomePatches. SIFT is a strong baseline, previously only surpassed by the high-dimensional CKN-grad [26]. DOAP is the first descriptor to outperform SIFT with the same dimensionality.

Due to the small size of RomePatches, we found it necessary to increase weight decay in SGD to $5 \times 10^{-4}$, and Dropout rate from $0.1$ to $0.5$ in the L2Net architecture. Also, adding Spatial Transformers did not improve results, possibly because the patches are already well aligned (see examples in Fig. 3); therefore we only report results for the binary and real-valued DOAP. As seen in Table 2, the real-valued DOAP outperforms SIFT and other descriptors with **88.4% mAP** on the test set, while the binary version also performs competitively. The comparison between DOAP and SIFT is fair, since they have the same input coverage and output dimensionality. Note that the closest competitor to DOAP, CKN-grad [26], is unsupervised and needs high dimensionality to perform well. By exploiting supervised learning and directly optimizing the evaluation metric, we are able to get better training and test performance while using 8x fewer dimensions (128 *vs*. 1024).

### 5.4. Image Matching in Oxford Dataset

Lastly, we use our learned descriptors to perform image matching in six image sequences from the classical Oxford dataset [23], where the matching pipeline also in-

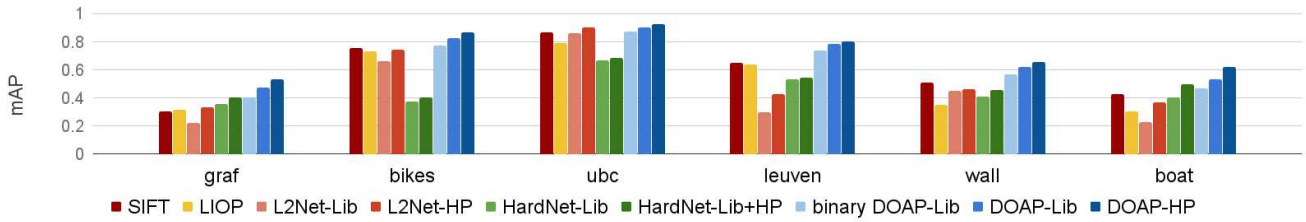


Figure 6. Image matching performance on the Oxford dataset [23]. Suffixes indicate the training set used (Lib: *Liberty*, HP: HPatches). Here, all versions of DOAP include the Spatial Transformer.

cludes interest point detection. We use the implementation from `VL-Benchmarks` [17]; features are detected by the Harris-Affine detector, and then patches are extracted with a magnification factor of 3 relative to the detected feature frames. The evaluation metric is mean Average Precision (mAP), computed as the area under the precision-recall curve derived from nearest neighbor matching.

We compare to SIFT, LIOP [36] (the best-performing handcrafted descriptor in [32]'s experiment), and 128-d real-valued versions of L2Net and HardNet with different training sets. We use the 256-bit binary and 128-d versions of DOAP trained on *Liberty*, and the 128-d version trained on HPatches. From the results in Fig. 6, we can see that SIFT is indeed difficult to beat, and good results on the UBC benchmark does not guarantee high-level task performance, especially in the case of HardNet. The real-valued DOAP consistently outperforms SIFT and other descriptors with significant margins, especially in the more challenging sequences such as *graf* and *boat*. The binary DOAP trained on *Liberty* also outperforms other real-valued descriptors on average, including L2Net trained on HPatches, and HardNet trained on the union of *Liberty* and HPatches.

### 5.5. Discussion

**Minibatch Sampling.** We discuss the minibatch sampling strategy used in training our models. First, note that in all datasets considered, patches are provided in groups: patches within a group correspond to the same 3D point and thus match each other (see Fig. 3). The group size, denoted $n$, is between 2 and 3 on average in UBC Phototour, and equals 10 in RomePatches. For HPatches, $n = 16$, as each patch has a reference version, and five variations from each difficulty level.

Our sampling strategy differs from those in local ranking approaches, where patch groups are often broken up to form pairs or triplets in a pre-processing step before training. Instead, we directly sample *groups* to construct training minibatches, so that patches belonging to the same group are always in the same batch. This allows our listwise optimization to utilize supervision with maximum efficiency. Let minibatch size be $M$, every training patch is associated with a listwise ranking constraint, that its $n-1$ matches need to be ranked at the top of a list of size $M-1$. This constraint alone needs $(n-1)(M-n)$ triplets to fully capture. Take UBC Phototour as an example, assuming $n = 2.5$ on average, a single minibatch of size $1024$ induces about $1.6 \times 10^6$ triplets, which is already $1/32$ of the total number of training triplets used in HardNet. For HPatches ($n = 16$), this number would be $1.5 \times 10^7$. However, triplets do not need to be explicitly generated in our listwise optimization.

**Time Complexity.** For a minibatch of size $M$, the pairwise distances between all examples are computed, and then binned into $b$-bin histograms. The time complexity is $O(bM^2)$. The quadratic dependency on $M$ is in fact optimal, due to distance computation.

There is also a tradeoff involving the batch size $M$. A larger batch size leads to longer lists and better performance, but slows training. Nevertheless, even with $M = 4096$, a single training epoch on *Liberty* takes less than 4 minutes on an Nvidia Titan X Pascal GPU. Similar to the case of UBC (Fig. 4), performance saturation is also observed around $M = 2048$ in HPatches and RomePatches.

## 6. Conclusion

In this work, we use deep neural networks to learn binary and real-valued local feature descriptors that optimize nearest neighbor matching performance. This is achieved through a listwise learning to rank formulation that directly optimizes Average Precision. Our formulation is general-purpose, and is superior to recent local ranking approaches. We further enhance our formulation with task-specific components: handling geometric noise with the Spatial Transformer, and mining labels using clustering. The learned descriptors achieve state-of-the-art performance in patch verification, patch retrieval, and image matching. Future work will explore the optimization of larger portions in vision pipelines, for example, by incorporating differentiable versions of robust estimation.

## Acknowledgements

A major part of this work was done during KH's internship at Honda Research Institute. This work is also partly conducted at Boston University, supported by a BU IGNITION award, NSF grant 1029430, and gifts from Nvidia.

## References


[1] Relja Arandjelović and Andrew Zisserman. Three things everyone should know to improve object retrieval. In *Proc. IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2012.

[2] Vassileios Balntas, Karel Lenc, Andrea Vedaldi, and Krystian Mikolajczyk. HPatches: A benchmark and evaluation of handcrafted and learned local descriptors. In *Proc. IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017.

[3] Vassileios Balntas, Edgar Riba, Daniel Ponsa, and Krystian Mikolajczyk. Learning local feature descriptors with triplets and shallow convolutional neural networks. In *Proc. British Machine Vision Conference (BMVC)*, 2016.

[4] Eric Brachmann, Alexander Krull, Sebastian Nowozin, Jamie Shotton, Frank Michel, Stefan Gumhold, and Carsten Rother. DSAC-differentiable RANSAC for camera localization. In *Proc. IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017.

[5] Fatih Cakir, Kun He, Sarah Adel Bargal, and Stan Sclaroff. MIHash: Online hashing with mutual information. In *Proc. IEEE International Conference on Computer Vision (ICCV)*, 2017.

[6] Gal Chechik, Varun Sharma, Uri Shalit, and Samy Bengio. Large scale online learning of image similarity through ranking. *Journal of Machine Learning Research*, 11:1109–1135, 2010.

[7] Christopher B Choy, JunYoung Gwak, Silvio Savarese, and Manmohan Chandraker. Universal correspondence network. In *Advances in Neural Information Processing Systems (NIPS)*, 2016.

[8] Philipp Fischer, Alexey Dosovitskiy, and Thomas Brox. Descriptor matching with convolutional neural networks: a comparison to SIFT. *arXiv preprint arXiv:1405.5769*, 2014.

[9] Martin A Fischler and Robert C Bolles. Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography. *Communications of the ACM*, 24(6):381–395, 1981.

[10] Xufeng Han, Thomas Leung, Yangqing Jia, Rahul Sukthankar, and Alexander C Berg. MatchNet: Unifying feature and metric learning for patch-based matching. In *Proc. IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015.

[11] Kun He, Fatih Cakir, Sarah Adel Bargal, and Stan Sclaroff. Hashing as tie-aware learning to rank. In *Proc. IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018.

[12] Max Jaderberg, Karen Simonyan, Andrew Zisserman, and Koray Kavukcuoglu. Spatial transformer networks. In *Advances in Neural Information Processing Systems (NIPS)*, 2015.

[13] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in Neural Information Processing Systems (NIPS)*, 2012.

[14] Brian Kulis. Metric learning: A survey. *Foundations and Trends® in Machine Learning*, 5(4):287–364, 2013.

[15] BG Kumar, Gustavo Carneiro, and Ian Reid. Learning local image descriptors with deep siamese and triplet convolutional networks by minimising global loss functions. In *Proc. IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.

[16] Marc T Law, Raquel Urtasun, and Richard S Zemel. Deep spectral clustering learning. In *Proc. International Conference on Machine Learning (ICML)*, 2017.

[17] Karel Lenc, Varun Gulshan, and Andrea Vedaldi. VLBenchmarks. `http://www.vlfeat.org/benchmarks/`.

[18] Karel Lenc and Andrea Vedaldi. Learning covariant feature detectors. In *ECCV Workshops*, pages 100–117, 2016.

[19] Daryl Lim and Gert Lanckriet. Efficient learning of mahalanobis metrics for ranking. In *Proc. International Conference on Machine Learning (ICML)*, 2014.

[20] Tie-Yan Liu. Learning to rank for information retrieval. *Foundations and Trends in Information Retrieval*, 3(3):225–331, 2009.

[21] David G Lowe. Distinctive image features from scale-invariant keypoints. *International Journal of Computer Vision*, 60(2), 2004.

[22] Brian McFee and Gert R Lanckriet. Metric learning to rank. In *Proc. International Conference on Machine Learning (ICML)*, 2010.

[23] Krystian Mikolajczyk and Cordelia Schmid. A performance evaluation of local descriptors. *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, 27(10):1615–1630, 2005.

[24] Anastasiya Mishchuk, Dmytro Mishkin, Filip Radenovic, and Jiri Matas. Working hard to know your neighbor's margins: Local descriptor learning loss. In *Advances in Neural Information Processing Systems (NIPS)*, 2017.

[25] Hyun Oh Song, Yu Xiang, Stefanie Jegelka, and Silvio Savarese. Deep metric learning via lifted structured feature embedding. In *Proc. IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.

[26] Mattis Paulin, Matthijs Douze, Zaid Harchaoui, Julien Mairal, Florent Perronnin, and Cordelia Schmid. Local convolutional features with unsupervised training for image retrieval. In *Proc. IEEE International Conference on Computer Vision (ICCV)*, 2015.

[27] Nikolay Savinov, Akihito Seki, Lubor Ladicky, Torsten Sattler, and Marc Pollefeys. Quad-Networks: Unsupervised learning to rank for interest point detection. In *Proc. IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017.

[28] Johannes L. Schönberger, Hans Hardmeier, Torsten Sattler, and Marc Pollefeys. Comparative evaluation of hand-crafted and learned local features. In *Proc. IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017.

[29] Matthew Schultz and Thorsten Joachims. Learning a distance metric from relative comparisons. In *Advances in Neural Information Processing Systems (NIPS)*, 2004.

[30] Edgar Simo-Serra, Eduard Trulls, Luis Ferraz, Iasonas Kokkinos, Pascal Fua, and Francesc Moreno-Noguer. Discriminative learning of deep convolutional feature point descriptors. In *Proc. IEEE International Conference on Computer Vision (ICCV)*, 2015.

[31] Karen Simonyan, Andrea Vedaldi, and Andrew Zisserman. Learning local feature descriptors using convex optimisation. *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, 2014.

[32] Yurun Tian, Bin Fan, and Fuchao Wu. L2-Net: Deep learning of discriminative patch descriptor in Euclidean space. In *Proc. IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017.

[33] Tomasz Trzcinski, Mario Christoudias, Pascal Fua, and Vincent Lepetit. Boosting binary keypoint descriptors. In *Proc. IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2013.

[34] Evgeniya Ustinova and Victor Lempitsky. Learning deep embeddings with histogram loss. In *Advances in Neural Information Processing Systems (NIPS)*, 2016.

[35] Yannick Verdie, Kwang Yi, Pascal Fua, and Vincent Lepetit. TILDE: a temporally invariant learned detector. In *Proc. IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015.

[36] Zhenhua Wang, Bin Fan, and Fuchao Wu. Local intensity order pattern for feature description. In *Proc. IEEE International Conference on Computer Vision (ICCV)*, 2011.

[37] Simon Winder, Gang Hua, and Matthew Brown. Picking the best DAISY. In *Proc. IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2009.

[38] Chao-Yuan Wu, R. Manmatha, Alexander J. Smola, and Philipp Krähenbühl. Sampling matters in deep embedding learning. In *Proc. IEEE International Conference on Computer Vision (ICCV)*, 2017.

[39] Kwang Moo Yi, Eduard Trulls, Vincent Lepetit, and Pascal Fua. LIFT: Learned Invariant Feature Transform. In *Proc. European Conference on Computer Vision (ECCV)*, 2016.

[40] Sergey Zagoruyko and Nikos Komodakis. Learning to compare image patches via convolutional neural networks. In *Proc. IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015.

[41] Xu Zhang, Felix X. Yu, Sanjiv Kumar, and Shih-Fu Chang. Learning spread-out local feature descriptors. In *Proc. IEEE International Conference on Computer Vision (ICCV)*, 2017.