Learning local feature descriptors with triplets and shallow convolutional neural networks

Vassileios Balntas[1]
http://www.iis.ee.ic.ac.uk/~vbalnt
Edgar Riba[2]
eriba@cvc.uab.es
Daniel Ponsa[2]
daniel@cvc.uab.es
Krystian Mikolajczyk[1]
k.mikolajczyk@imperial.ac.uk

[1] Imperial College London
London, UK

[2] Computer Vision Center, Computer Science Department
Universitat Autònoma de Barcelona
Bellaterra (Barcelona), Spain

Abstract

It has recently been demonstrated that local feature descriptors based on convolutional neural networks (CNN) can significantly improve the matching performance. Previous work on learning such descriptors has focused on exploiting pairs of positive and negative patches to learn discriminative CNN representations. In this work, we propose to utilize triplets of training samples, together with in-triplet mining of hard negatives. We show that our method achieves state of the art results, without the computational overhead typically associated with mining of negatives and with lower complexity of the network architecture. We compare our approach to recently introduced convolutional local feature descriptors, and demonstrate the advantages of the proposed methods in terms of performance and speed. We also examine different loss functions associated with triplets.

1 Introduction

Finding correspondences between images via local descriptors is one of the most extensively studied problems in computer vision due to the wide range of applications. The field has witnessed several breakthroughs in this area such as SIFT [14], invariant region detectors [17], fast binary descriptors [4], optimised descriptor parameters [20, 22] which have made a significant and wide impact in various computer vision tasks. Recently end-to-end learnt descriptors [9, 11, 19, 24] based on CNN architectures and training on a large dataset of positive and negative sample pairs, were demonstrated to significantly outperform state of the art features. This was a natural adoption of CNN to local descriptors as deep learning had already been shown to significantly improve in many computer vision areas [13].

Recent work on deep learning for learning feature embeddings examines the use of triplets of samples instead of solely focusing on pairs [12, 21, 23]. Different loss functions are proposed in these works, but a systematic study of their characteristics is yet to be done.

Pages 119.1-119.11
DOI: https://dx.doi.org/10.5244/C.30.119

In addition, these works are focused on more general embeddings (e.g. product similarity, 3D description of objects, MNIST classification). In this work, we investigate the use of triplets in learning local feature descriptors with convolutional neural networks. Other contributions include: 1) we examine different different loss functions for triplet based-learning, 2) we investigate the performance of these methods in terms of patch matching, and patch pairs classification in widely used benchmarks, 3) we show that in-triplet hard negative mining can lead to improved results, 4) we demonstrate that excellent descriptor performance can be obtained with a shallow network thus avoiding computationally complex architectures and expensive mini-batch hard negative mining.

2 Related work

The design and implementation of local descriptors has undergone a remarkable evolution over the past two decades ranging from differential or moment invariants, correlations, PCA projected patches, histograms of gradients or other measurements, etc. An overview of pre-2005 descriptors with SIFT [14] identified as the top performer can be found in [16]. Its benchmark data accelerated the progress in this field and there have been a number of notable contributions, including recent DSP-SIFT [7], falling into the same category of descriptors as SIFT but the improvements were not sufficient to supersede SIFT in general. The research focus shifted to improve the speed and memory footprint e.g. as in BRIEF [4] and the follow up efforts. Introduction of datasets with correspondence ground truth [22] stimulated development of learning based descriptors which try to optimise descriptor parameters and learn projections or distance metrics [15, 20] for better matching.

End-to-end learning of patch descriptors using CNN has been attempted in several works [9, 11, 19, 24] and consistent improvements were reported over the state of the art descriptors. Interest in the field started from results shown in [9] that the features from the last layer of a convolutional deep network trained on ImageNet [18] can outperform SIFT. This was a significant result, since the convolutional features from ImageNet were not specifically learnt for such local representations.

Learning a CNN from local patches extracted from local features only, based on a siamese architecture with hinge contrastive loss [10] was demonstrated in [11, 19, 24] to significantly improve the matching performance. This approach was originally proposed in [25], however due to the limited evaluation this work was not immediately followed.

Note that in [11, 24] both feature layers and metric layers are learnt in the same network. Thus, the final contrastive loss is optimised in terms of the abstract metric learned in the last layer of the network. On the contrary, [19] directly uses the features extracted after the convolutional layers of the CNN, without training a specialised distance layer. This allows the extracted descriptors to be used in traditional pipelines. However, the experiments from [24] show that metric learning performs better than generic $L2$ matching. In our experiments, we also use features from CNNs without any metric learning layer.

Another important observation from [24] is that multiscale architectures perform better than the single scale ones. However, this is not unique to the CNNs, since previous works have shown that aggregating descriptors from multiple scales, improves the results. We focus on single scale architectures, since these are the building blocks for multiscale descriptors, and improving them, will improve the final multiscale results as argued by the DSP theory from [7].

3 Learning patch descriptors

In this section, we first discuss the two most commonly used loss functions when learning with triplets, and we then investigate their characteristics. A patch descriptor is considered as a non-linear encoding resulting from a final layer of a convolutional neural network. Let $\boldsymbol{x} \in \mathbb{R}^{n \times n}$ represent the patch given as input to the network, and $f(\boldsymbol{x}) \in \mathbb{R}^D$ represent the D features given as output from the network. For all of the methods below, the goal is to learn the embedding $f(\boldsymbol{x})$ s.t. $||f(\boldsymbol{x}_1) - f(\boldsymbol{x}_2)||_2$ is low if $\boldsymbol{x}_1$ and $\boldsymbol{x}_2$ are extracted from the same physical point location i.e. positive match, and high otherwise.

3.1 Learning with pairs

Learning with pairs involves training from samples of the form $\{\boldsymbol{x}_1, \boldsymbol{x}_2, \ell\}$, with ℓ being a label for the patch pair, which is -1 for negative pairs, and 1 for positive pairs. The contrastive loss is defined as

$$l(\boldsymbol{x}_1, \boldsymbol{x}_2; \ell) = \begin{cases} ||f(\boldsymbol{x}_1) - f(\boldsymbol{x}_2)||_2 & \text{if } \ell = 1 \\ max(0, \mu - ||f(\boldsymbol{x}_1) - f(\boldsymbol{x}_2)||_2) & \text{if } \ell = -1 \end{cases} \quad (1)$$

where μ is an arbitrarily set margin. Note that the weights of the CNN in $f(\cdot)$ need to be regularised, otherwise the margin would have no effect. Intuitively the hinge embedding loss penalizes positive pairs that have large distance and negative pairs that have small distance (less than μ).

Note that learning local feature descriptors is a more specific problem than general image classification such as in ImageNet, since the transformations a local patch can undergo are limited compared to different objects of the same visual category. In addition, patches in pairs representing negative examples are usually very different, thus make it easy for the learning process to optimize the distances. This issue is identified in [19], where the majority of the negative patch pairs ($\ell = -1$) do not contribute to the update of the gradients in the optimization process as their distance is already larger than μ parameter in Eq. (1). To address this issue hard negative mining was proposed [19] to include more negative pairs in the training. The hard negative training pairs were identified by their distance and a subset of these examples were re-fed to the network for gradient update in each iteration. Note that while this process leads to more discriminative convolutional features, it also comes at a very high computational cost, since in each epoch, a subset of the training data need to be backpropagated again through the network. Specifically, the best performing architecture from [19], required 67% of the computational cost to be spent for mining hard negatives.

3.2 Learning with triplets

Recent work in [12] shows that learning representations with triplets of examples, gives much better results than learning with pairs using the same network. Inspired by this, we focus on learning feature descriptors based on triplets of patches.

Learning with triplets involves training from samples of the form $\{\boldsymbol{a}, \boldsymbol{p}, \boldsymbol{n}\}$, where a is the *anchor*, p *positive*, which is a different sample of the same class as a, and n *negative* is a sample belonging to a different class. In our case, a and p are different viewpoints of the same physical point, and n comes from a different keypoint. Furthermore, optimising the parameters of the network brings a and p close in the feature space, and pushes a and n far apart. For brevity, we shall write that $\delta_+ = ||f(\boldsymbol{a}) - f(\boldsymbol{p})||_2$ and $\delta_- = ||f(\boldsymbol{a}) - f(\boldsymbol{n})||_2$.

We can categorise the loss functions that have been proposed in the literature for learning convolutional embeddings with triplets into two groups, the *ranking-based losses* and the *ratio-based losses* [12, 21, 23]. Below we give a brief review of both categories, and discuss their differences.

3.2.1 Margin ranking loss

This ranking loss that was first proposed for learning embeddings using convolutional neural networks in [21] is defined as

$$\lambda(\delta_+, \delta_-) = max(0, \mu + \delta_+ - \delta_-) \tag{2}$$

where μ is a margin parameter. The margin ranking loss is a convex approximation to the $0-1$ ranking error loss, which measures the violation of the ranking order of the embedded features inside in the triplet. The correct order should be $\delta_- > \delta_+ + \mu$. If that is not the case, then the network adjusts its weights to achieve this result. As it can be seen the formulation also involves a margin, similarly to Eq.(1). Note that if this marginal distance difference is respected, the loss is 0, and thus the weights are not updated. Fig. 1 (b) illustrates the loss surface of $\lambda(\delta_+, \delta_-)$. The loss remains 0 until the margin is violated, and after that, there is a linear increase. Also note that the loss in not upper bounded, only lower bounded to 0.

3.2.2 Ratio loss

In contrast to the ranking loss that forces the embeddings to be learned such that they satisfy ranking of the form $\delta_- > \delta_+ + \mu$, a ratio loss is investigated in [12] which optimises the ratio distances within triplets. This loss learns embeddings such that $\frac{\delta_-}{\delta_+} \to \infty$.

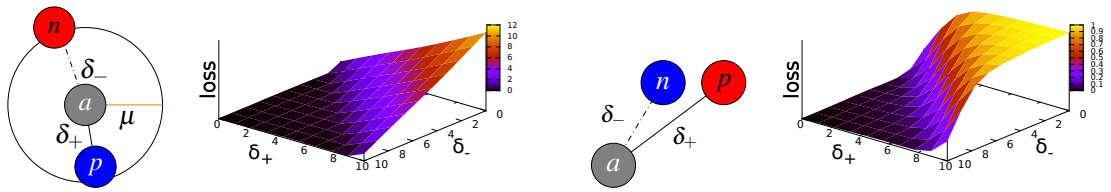
$$\hat{\lambda}(\delta_+, \delta_-) = (\frac{e^{\delta_+}}{e^{\delta_+} + e^{\delta_-}})^2 + (1 - \frac{e^{\delta_-}}{e^{\delta_+} + e^{\delta_-}})^2 \tag{3}$$

As one can examine from Eq. 3, the goal of this loss function is to force $(\frac{e^{\delta_+}}{e^{\delta_+}+e^{\delta_-}})^2$ to 0, and $(\frac{e^{\delta_-}}{e^{\delta_+}+e^{\delta_-}})^2$ to 1. Note that both are achieved by the first term of the equation, but we report here the original formulation from [12]. There is no margin associated with this loss, and by definition we have $0 \leq \hat{\lambda} \leq 1$ for all values of δ_-, δ_+. Note that unlike the margin-ranking loss, where $\lambda = 0$ is possible, every training sample in this case is associated with some non-negative loss value. Fig. 1 (d) shows the loss surface of $\hat{\lambda}(\delta_+, \delta_-)$, which compared to the ranking based loss has a clear slope between the two loss levels, and the loss reaches a plateau quickly when $\delta_- > \delta_+$. Also note that this loss is upper bounded to 1.

3.3 In-triplet hard negative mining with anchor swap

All previous works that exploit the idea of triplet based learning use only two of the possible three distances within each triplet w.r.t. one sample used as an *anchor*, thus ignoring the third distance $\delta'_- = ||f(\boldsymbol{p}) - f(\boldsymbol{n})||_2$. Note that since the feature embedding network already computes the representations for $f(\boldsymbol{a}), f(\boldsymbol{p}), f(\boldsymbol{n})$, there is no need for extra convolutional overhead to compute δ'_- except evaluating the L_2 distance.

We define the *in-triplet hard negative* as $\delta_* = min(\delta_-, \delta'_-)$. If $\delta_* = \delta'_-$, we swap $\{a, p\}$, and thus p becomes the *anchor*, and a becomes the *positive* sample. This ensures that the



(a) (b) (c) (d)

Figure 1: (a) Margin ranking loss. It seeks to push n outside the circle defined by the margin μ, and pull p inside. (b) Margin ranking loss values in function of δ_-, δ_+ (c) Ratio loss. It seeks to force δ_+ to be much smaller than δ_-. (d) Ratio loss values in function of δ_-, δ_+

hardest negative inside the triplet is used for backpropagation. Subsequently, the margin ranking loss becomes $\lambda(\delta_+, \delta_*) = max(0, \mu + \delta_+ - \delta_*)$. A similar expression can be devised for the ratio loss. This simple technique can lead to improved results without computational overhead, as we experimentally show in section 4.1.

3.4 Implementation details

To demonstrate the impact that triplet based training has on the performance of CNN descriptors we use a simple network architecture : {Conv(7,7)-Tanh-Pool(2,2)-Conv(6,6)-Tanh-FC(128)} implemented in `Torch` [6] with the following simplified training process. CNN is trained from $5M$ triplets sampled on-the-fly using patches from [15]. We do not use data augmentation unlike in typical CNNs for general classification or convolutional feature descriptors from [24][11]. When forming a triplet for training we choose randomly a positive pair of patches that originate from the same physical point and a randomly sampled patch from another keypoint. This is in contrast to other works where carefully designed schemes of choosing the training data are used in order to enhance the performance [11, 21]. For the optimization the Stochastic Gradient Descend [3] is used, and the training is done in batches of 128 items, with a learning rate of 0.1 which is temporally annealed, momentum of 0.9 and weight decay of 10^{-6}. We also reduce the learning rate every epoch. The convolution methods are from the NVIDIA cuDNN library [5]. The training of a single epoch with $5M$ training triplets takes approximately 10 minutes in an NVIDIA Titan X GPU.

It is worth noting that the CNN used in our experiments consists of only two convolutional layers, while all of the other state-of-the art deep feature descriptors consist of four or more layers [11, 19, 24]. Our motivation for such shallow network is to develop a descriptor for practical applications including those requiring real time processing. This is a challenging goal given that all previously introduced descriptors are computationally very intensive, thus impractical for most applications. This design is also inspired by the approach introduced in [20], where pooling of the responses of Gaussian filters and a simple linear projection produced very good results. Thus, we build a simple hierarchical network that is based on 100 convolutional filters, followed by a linear transformation that projects the responses of the filters to the desired output dimensionality. Several other implementation variants are possible such as different non-linearity layers (e.g. ReLU as in [11, 24]), extra normalization layers, or multiscale architectures but these are likely to further improve the results and are beyond the scope of this work.

4 Experimental evaluation

In this section we evaluate the proposed local feature descriptor within the two most popular benchmarks in the field of local descriptor matching and we test on different datasets to show that it can generalise well. We compare our method to SIFT [14], Convex optimization [20] and the recently introduced convolutional feature descriptors MatchNet [11], DeepCompare [24] and DeepDesc [19], which are currently the state of the art in terms of matching accuracy. The original code was used in all the experiments. More details can be found in the supplementary materials. We name our four variants `TFeat-ranking` for the networks learnt with the ranking loss, `TFeat-ranking*` for the networks learnt with the ranking loss with anchor swap, `TFeat-ratio` for the ratio loss, and `TFeat-ratio*` for the ratio loss with anchor swap.

Note that for a fair comparison, we do not use the multi-scale *2ch architectures* from [24]. Multi-scale approaches use multiple patches from each example, with extra inputs in form of cropped sub-patches around the center of each patch. This introduces information from different samples in the scale-space and it has been shown to lead to significant improvements in terms of matching accuracy [7]. Such approach can be used for various descriptors (e.g. MatchNet-2ch, TFeat-2ch, DeepDesc-2ch). The evaluation is done with two different evaluation metrics frequently found in the literature, patch pair classification success in terms of ROC curves [22], and mean average precision in terms of correct matching of feature points between pairs of images [16]. Note that these two metrics are of very different nature, the former measures how succesfull a classification of positive and negative patch pairs is, and the latter is evaluating the performance of a descriptor in nearest neighbour matching scenario where the task is to find correspondences in two large sets of descriptors.

4.1 Patch pair classification

The patch pair classification benchmark measures the ability of a descriptor to discriminate positive patch pairs from negative ones in the Photo Tour dataset [15]. This dataset consists of three subsets *Liberty*,*Yosemite* & *Notredame*, with each containing more than 500k patch pairs extracted around keypoints. We follow the protocol proposed in [15] where the ROC curve is generated by thesholding the distance scores between patch pairs. The number reported here is the false positive rate at 95% true positive rate (FPR95), as used in many influencial works in the field. For the evaluation we use the $100K$ patch pairs proposed as defined in the benchmark. Note that DeepDesc [19], does not report performance with training based on a single dataset, therefore for each test set, the training is performed on the other two datasets.

The results for each of the combinations of training and testing using the three subsets of the Photo Tour dataset are shown in Table 1 including the average across all possible combinations. Our networks outperform all the previously introduced single-scale convolutional feature descriptors, and in some cases with large margins except from one training-test combination where the 4096-dimensional version of MatchNet outperforms our TFeat variants. However, even in this case, the version of MatchNet with comparable dimensionality to our descriptors is outperformed by three of our variants. Also note that MatchNet is specifically designed for patch pair classification, since it also includes a similarity metric layer trained on top of the feature layer.

Table 1: Results form the Photo-Tour dataset [15]. Numbers are reported in terms of *FPR95* following state of the art in this field (see text for more details). *Italics* indicate the descriptors introduced here, and **bold** numbers indicate the top performing descriptor. Yos:Yosemite, Lib:Liberty, Not:Notredame.

Training		Not	Lib	Not	Yos	Yos	Lib	
Testing		Yos		Lib		Not		
Descriptor	#							mean
SIFT [14]	128	27.29		29.84		22.53		26.55
ImageNet$_{4conv}$ [9]	128	30.22		14.26		9.64		18.04
ConvexOpt [20]	80	10.08	11.63	11.42	14.58	7.22	6.17	10.28
DeepCompare siam [24]	256	15.89	19.91	13.24	17.25	8.38	6.01	13.45
Deepcompare *siam2stream*	512	13.02	13.24	8.79	12.84	5.58	4.54	9.67
DeepDesc [19]	128	16.19		8.82		4.54		9.85
MatchNet [11]	512	11	13.58	8.84	13.02	7.7	4.75	9.82
MatchNet [11]	4096	8.39	10.88	**6.90**	10.77	5.76	3.87	7.75
TFeat-ratio	128	**8.32**	**10.25**	8.93	**10.13**	**4.12**	**3.79**	**7.59**
*TFeat-ratio**	128	**7.24**	**8.53**	8.07	**9.53**	**4.23**	**3.47**	**6.84**
TFeat-margin	128	**7.95**	**8.10**	7.64	**9.88**	**3.83**	**3.39**	**6.79**
*TFeat-margin**	128	**7.08**	**7.82**	7.22	**9.79**	**3.85**	**3.12**	**6.47**

4.2 Nearest neighbour patch matching

To measure the nearest neighbour matching performance, we establish correspondence ground truth using the homographies and the overlap error from [16]. We consider two feature points between the two images in correspondence if the overlap error between the detected regions is less than 50%. Note that a region from one image can be in correspondence with several regions from the other image. Each image has an associated set of approximately 1K patches. The results are presented with precision-recall curves as it was originally proposed in [16]. More specifically, for each patch from the left image we find its nearest neighbour in the right image. Based on the ground truth overlap we identify the false positives and true positives, and generate precision-recall curves. The area under the precision-recall curve is the reported mean-average precision [7, 22, 24]. For this experiment, we use the `vl_benchmarks` [27] library (`vl_covdet` function), with some minor modifications to limit the descriptors extracted from an image to 1K, which is important to avoid bias by different numbers of features in different images. For all the experiments below, the descriptors are trained on Liberty-DoG patches [15].

For the nearest neighbor matching test two datasets are mainly used in the literature, *Oxford matching dataset* [16], which is of small size, but include images acquired by a camera, and the *generated matching dataset* [9] which is much larger in volume but created synthetically. In the following sections, we discuss our results in those two datasets.

4.2.1 Ratio loss vs. margin loss

Fig 2 shows the performance of the same network trained for the same number of epochs on the Liberty dataset. We report the *mAP* of image matching in the Oxford dataset. It can be observed that the margin based loss increases the performance as more epochs are used in the training process. No over-fitting is noticed when training and testing patch classification

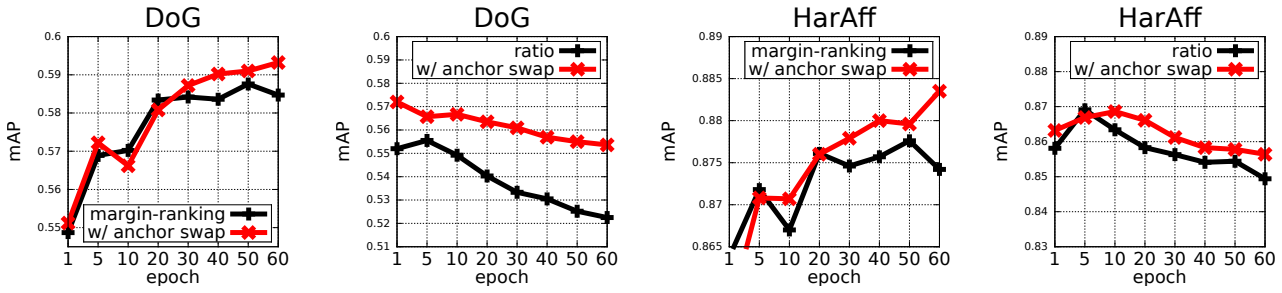


Figure 2: Ratio based loss function overfits in the process of separating the positive and negative pairs within a triplet, and does not perform well in the nearest neighbour matching experiment. On the contrary, learning with triplets and margin ranking does not suffer from this problem which shows that ranking methods are more suitable for nearest neighbor matching scenarios.

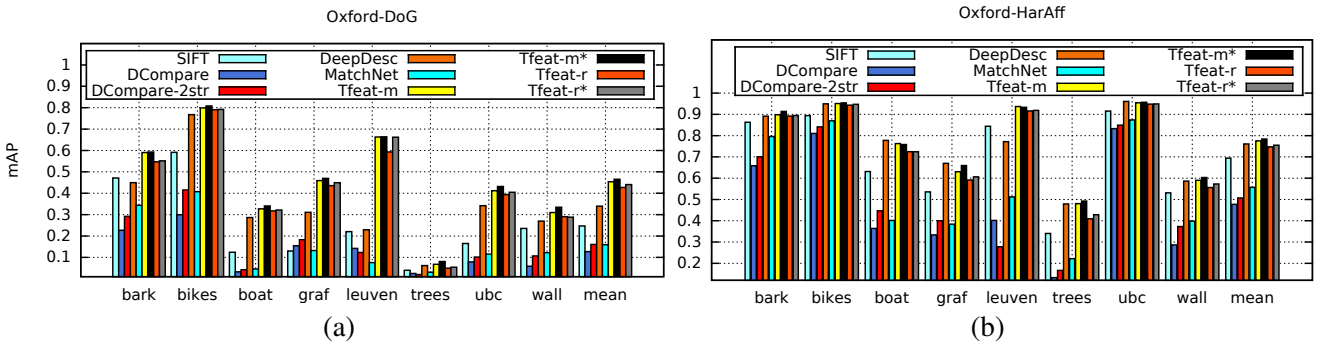


Figure 3: Evaluation on the Oxford image matching dataset [16], for two different types of feature extractors, DoG and HarrisAffine.

(e.g. training with ratio loss on Liberty and testing on Yosemite or Notredame). Interestingly, the ratio loss seems to decrease the patch matching performance as the network is trained for more epochs. This also hints that other methods from the literature that were only tested in the patch classification scenario, may not perform well in matching. In our view, this shows that evaluating descriptors only in terms of ROC curves is not representative for realistic matching scenarios.

Finally, the results show that the loss functions with anchor swapping perform better than without swapping. Note that this simple technique can lead to improved results with no additional computational overhead.

4.2.2 Keypoint matching

Figure 3 presents the *mAP* results for Oxford benchmark, across all image sequences from the Oxford dataset, for two different keypoint detectors, DoG and Harris-Affine. Note that all networks are trained on DoG keypoints. In the case of our ratio loss, we use the networks from the first epoch, since all the next epochs would exhibit lower performance (cf. Fig 2). In the case of the DoG keypoints, our networks outperform all the others in terms of *mAP*. The second best performing descriptor is the DeepDesc descriptor from [19]. We stress again here, that this descriptor was below the state of the art in terms of ROC curves and FPR95 as shown in Table 1. This confirms our findings that the classification benchmark is not a representative measure for the common real-world application of descriptors which often relies on nearest neighbor matching. When using Harris-Affine keypoints our descriptor still outperforms the others, although with a smaller margin.

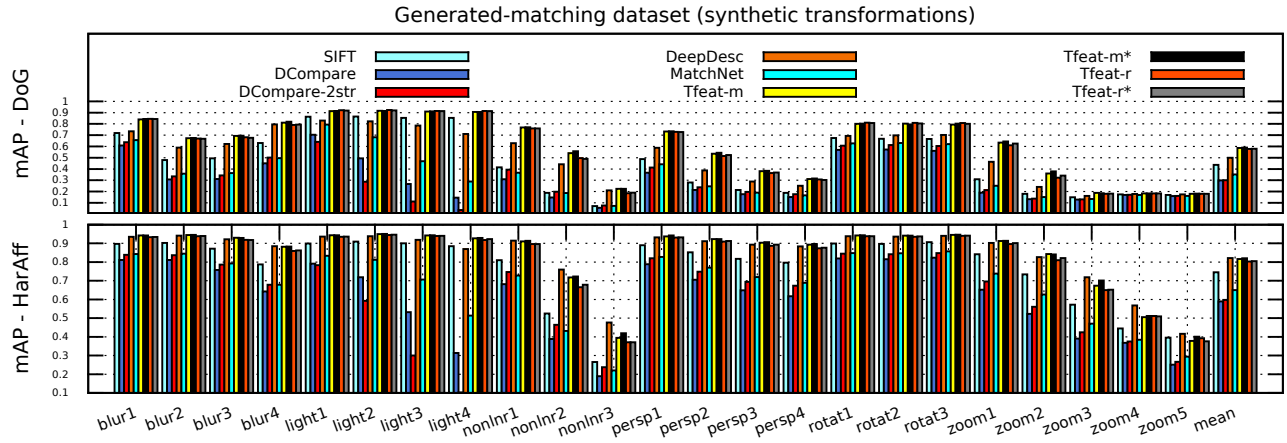


Figure 4: Evaluation on the generated-matching dataset [9], for two different types of feature extractors, DoG and HarrisAffine.

4.2.3 Image transformations

Figure 4 shows the results across various synthetic transformations of image pairs. Our descriptor gives the top scores in most sequences. It is also worth noting, that even though this dataset has some severe deformations as well as nonlinear filtering, the overall performance for both types of feature extractors is higher than for the Oxford dataset. This shows that synthetic deformations are less challenging for descriptors than some real-world changes as the ones found in Oxford dataset.

4.3 Efficiency

One of the main motivations behind this work, was the need for a fast and practical feature descriptor based on CNN. The small network trained with triplets that we used in our experiments, is very efficient in terms of descriptor extraction time. We compare the extraction time per patch, averaged over $20K$ patches, of recently introduced convolutional feature descriptors. The extraction is done with NVIDIA Titan X GPU. Our descriptor is 10 times faster than DeepCompare [24], and 50 times faster than MatchNet [11] and DeepDesc [19]. In fact, when running on GPU, we reach speeds of $10\mu s$ per patch which is comparable with the CPU speeds of the fast binary descriptors[4]. This is a significant advantage over the previously proposed descriptors and makes CNN based descriptors applicable to practical problems with large datasets.

4.4 Conclusion

This work introduced a new approach to training CNN architecture for extracting local image descriptors in the context of patch matching. The results show that using triplets for training results in a better descriptor and faster learning. The networks can be simplified and extract features with a speed comparable to BRIEF. Also the dimensionality can be significantly reduced compared to other CNN based descriptors. We show that due to these properties the proposed network is less prone to over-fitting and has good generalisation properties. In addition, the high computational cost of hard negative mining has been successfully replaced by the very efficient triplet based loss.

We also demonstrate that ratio-loss based methods are more suitable for patch pair classification, and margin-loss based methods work better in nearest neighbour matching appli-

cations. This indicates that a good performance on patch classification does not necessarily generalise to a good performance in nearest neighbour based frameworks.

We provide all the learned models and the training code for all the variants at https://github.com/vbalnt/tfeat.

5 Acknowledgements

This work was supported by EPSRC project EP/N007743/1 and partially supported by the spanish project FireDMMI (TIN2014- 56919-C3-2-R).

References

[1] M. S. Aurelien Bellet, Amaury Habrard. A survey on metric learning for feature vectors and structured data. *Arxiv*, 2010.

[2] S. Sinha, J.M. Frahm, M. Pollefeys and Y. Genc. Workshop on Edge Computing Using New Commodity Architectures. 2006.

[3] L. Bottou. Stochastic gradient tricks. *Neural Networks, Tricks of the Trade, Reloaded*, pp 430–445, 2012.

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

[5] S. Chetlur, C. Woolley, P. Vandermersch, J. Cohen, J. Tran, B. Catanzaro, and E. Shelhamer. CUDNN: Efficient primitives for deep learning. *Arxiv*, 2014.

[6] R. Collobert, K. Kavukcuoglu, and C. Farabet. Torch7: A matlab-like environment for machine learning. In *BigLearn, NIPS Workshop*, 2011.

[7] J. Dong and S. Soatto. Domain-size pooling in local descriptors: DSP-SIFT. In *CVPR* , 2015.

[8] P. F. Felzenszwalb, R. B. Girshick, D. McAllester, and D. Ramanan. Object detection with discriminatively trained part based models. *PAMI*, 32(9):1627–1645, 2010.

[9] P. Fischer, A. Dosovitskiy, and T. Brox. Descriptor matching with convolutional neural networks: a comparison to sift. *Arxiv*, 2014.

[10] R. Hadsell, S. Chopra, and Y. LeCun. Dimensionality reduction by learning an invariant mapping. In *CVPR*, 2006.

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

[12] E. Hoffer and N. Ailon. Deep metric learning using triplet network. *Arxiv*, 2014.

[13] Y. LeCun, Y. Bengio, and G. Hinton. Deep learning. *Nature*, 521(7553):436–444, 2015.

[14] D. G. Lowe. Distinctive image features from scale-invariant keypoints. *IJCV*, 60(2):91–110, 2004.

[15] G. H. M. Brown and S. Winder. Discriminative learning of local image descriptors. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 33(1), pp.43-57, 2011.

[16] K. Mikolajczyk and C. Schmid. A performance evaluation of local descriptors. *PAMI*, pp 257–263, 2003.

[17] K. Mikolajczyk and C. Schmid. Scale and affine invariant interest point detectors. *IJCV*, 60(1):63–86, 2004.

[18] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy,

A. Khosla, M. Bernstein, A. C. Berg, and L. Fei-Fei. ImageNet Large Scale Visual Recognition Challenge. *IJCV*, pp 1–42, 2015.

[19] 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.

[20] K. Simonyan, A. Vedaldi, and A. Zisserman. Learning local feature descriptors using convex optimisation. *PAMI*, 36(8),pp 1573-1585, 2014.

[21] J. Wang, Y. Song, T. Leung, C. Rosenberg, J. Wang, J. Philbin, B. Chen, and Y. Wu. Learning fine grained image similarity with deep ranking. *CVPR*, 2014.

[22] S. Winder, G. Hua, and M. Brown. Picking the best daisy. In *CVPR*, 2009.

[23] P. Wohlhart and V. Lepetit. Learning Descriptors for Object Recognition and 3D Pose Estimation. In *CVPR*, 2015.

[24] S. Zagoruyko and N. Komodakis. Learning to compare image patches via convolutional neural networks. In *CVPR*, 2015.

[25] M. Jahrer, M. Grabner, and H. Bischof. Learned local descriptors for recognition and matching. In *Computer Vision Winter Workshop*, 2008

[26] C. Osendorfer, J. Bayer, S. Urban, and P. Van Der Smagt. Convolutional neural networks learn compact local image descriptors. In *ICONIP* 2013

[27] A. Vedaldi and B. Fulkerson VLFeat: An Open and Portable Library of Computer Vision Algorithms, 2008.