Learned BRIEF – transferring the knowledge from hand-crafted to learning-based descriptors

Nina Žižakić and Aleksandra Pižurica
Group for Artificial Intelligence and Sparse Modelling (GAIM), TELIN
Ghent University
Ghent, Belgium
{nina.zizakic, aleksandra.pizurica}@ugent.be

***Abstract*—In this paper, we present a novel approach for designing local image descriptors that learn from data and from hand-crafted descriptors. In particular, we construct a learning model that first mimics the behaviour of a hand-crafted descriptor and then learns to improve upon it in an unsupervised manner. We demonstrate the use of this knowledge-transfer framework by constructing the *learned BRIEF* descriptor based on the well-known hand-crafted descriptor BRIEF. We implement our learned BRIEF with a convolutional autoencoder architecture. Evaluation on the HPatches benchmark for local image descriptors shows the effectiveness of the proposed approach in the tasks of patch retrieval, patch verification, and image matching.**

***Index Terms*—local image descriptors, autoencoders, unsupervised deep learning, knowledge transfer**

I. Introduction

Local feature descriptors play a crucial role in many image processing tasks, such as object tracking, object recognition, panorama stitching, and image retrieval. In the past decade, the approach to designing local feature descriptors has shifted from using hand-crafted features to the (deep) learning approach. While many advancements in the learning approach have lead to better performance on benchmarks [1], [2], hand-crafted descriptors still outperform the learned ones in many cases of practical interest and are often a descriptor of choice in practice [3].

According to recent studies [4], this is largely due to the fact that descriptor learning is typically approached as a standalone problem, without considering a broader image processing task where it should be integrated. Moreover, most of the descriptor learning approaches are supervised methods, relying on relatively many annotated examples. Such specific labeled datasets are often not available. Unsupervised methods, by definition, do not suffer from this problem.

Other reasons why hand-crafted descriptors are still preferred over the learned ones in practice include their interpretability and robustness to relevant distortions, which is a shortcoming of the current learned descriptors.

In this paper, we present a novel framework for constructing a learning model that first mimics the behaviour of a hand-crafted descriptor and then learns to improve upon it in an unsupervised manner. Two important assets of the proposed approach compared to most of the current descriptor learning methods are: (1) improved explainability, inherited from a chosen hand-crafted descriptor that it builds upon, and (2) no need for excessive labeled datasets. Compared to the hand-crafted descriptor, we achieve better performance and avoid the need for various ad-hoc choices of the parameters. Due to the unsupervised nature of our method, the descriptor can be fine-tuned for various applications without the specific labeled dataset for that application.

The framework that we present is a form of knowledge transfer. Knowledge transfer is a term used in machine learning community where a neural network is bootstrapped from an initial training based on, for example, a small or a private dataset [5]. In the case of our framework, this bootstrapping process is performed by training the network using a hand-crafted descriptor as a ground truth. The network is then further trained in an unsupervised manner as part of an autoencoder. This idea is partially motivated by the work from [6] where, in a different context, the authors discuss unrolling an iterative inverse-imaging problem into a convolutional neural network. They argue that for networks designed in this way, the performance of the properly trained network cannot be worse than that of the original algorithm which is its special case. In the same way, we claim that our learned autoencoder-based local image descriptor, with proper training, will be superior to the hand-crafted one that served as its initialisation. We support this claim by the results of a thorough evaluation on standard benchmarks for patch descriptors [1].

Our method is a general framework for transferring the knowledge from an arbitrary local image descriptor. In this paper, we use the framework to transfer the knowledge from the BRIEF descriptor [7], which enjoys popularity as a computationally simple and efficient descriptor. However, our method can be applied on other descriptors too, as we will explain in the Section III.

To summarise, our main contributions are twofold:

- We propose a framework for the transfer of knowledge from a hand-crafted descriptor to an unsupervised-learning–based descriptor. At the time of writing, no such framework has been reported, to the best of our knowledge.
- Within this framework, we propose an elegant method for implementing a learned BRIEF.

This research received funding from the Flemish Government (AI Research Program).

In the following section, we discuss the related work and preliminaries, and introduce the notation. In Section III we first introduce our framework for knowledge transfer from hand-crafted to learning-based descriptors, and then we apply this framework to implement a learned BRIEF. In Section IV we evaluate the performance of our learned BRIEF in comparison with the original BRIEF descriptor. Section V concludes the paper and discusses the future work.

II. Related work

The traditional approach to designing local image descriptors is using hand-crafted features. The most well-known hand-crafted descriptors include SIFT [8], HOG [9], SURF [10], BRIEF [7], ORB [11], which continue to be relevant to modern approaches with implementations being available in the computer vision libraries such as OpenCV. In recent years, the development of deep learning techniques has resulted in numerous learned descriptors [4], [12]–[17]. These descriptors are mostly learned in a supervised fashion with the labels on pairs of patches, indicating their similarity or dissimilarity.

While a few learned descriptors show high performance on benchmarks [1], [2], the established hand-crafted descriptors such as SIFT are consistently chosen over the learned descriptors in practical contexts [3]. One explanation for this preference could be that the literature typically approaches descriptor learning as a standalone problem and neglects the challenges of integrating that solution as a component of a broader image processing task [4].

Since supervised methods are dependent on labeled data, it is often unfeasible to create a descriptor for a specific image processing task. In contrast to supervised methods, unsupervised methods such as autoencoders are a viable option since they do not depend on labeled data.

The application of autoencoders to the problem of descriptor learning was first proposed by Chen et al. [18]. In our previous work [19], [20], we proposed autoencoder-based patch descriptors designed for applications with many patch comparisons within a single image.

While these autoencoder-learned descriptors showed promising results, they lacked explainability, which renders them less attractive for some applications, especially in high-stakes scenarios. Moreover, the computational complexity and especially memory requirements may be prohibitive for some real-time processing tasks with high resolution images.

In this paper, we present a novel method that enables the transfer of knowledge from a hand-crafted descriptor to an unsupervised-learning–based descriptor through the use of autoencoders. By transferring knowledge from a hand-crafted into a learning-based descriptor, we expect that the learning process will be biased in a way such that the final descriptor will be an optimised variant of the original hand-crafted descriptor which is already well understood. A descriptor generated through this learning approach has a further advantage in that it can be fine tuned for a specific task.

In the following, we design a learned BRIEF descriptor, which keeps the lightweightedness of BRIEF while having improved performance by training it as part of the autoencoder. Moreover, in contrast to the original BRIEF, the learned BRIEF achieves rotational and translational invariance.

A. BRIEF descriptor

BRIEF (Binary Robust Independent Elementary Features) is a binary local image descriptor that, for an input patch, calculates its binary code [7]. The code is calculated as follows (see Figure 1-a (top)). First, the input patch is smoothed in order to decrease the sensitivity to noise. The smoothing is done using an averaging kernel of size 9×9. Then, a binary feature vector is created, composed of the binary test responses. A binary test τ between pixels coordinates x and y on a patch $\mathbf{p}$ is defined by:

$$\tau(\mathbf{p}; x, y) = \begin{cases} 1 & \mathbf{p}(x) \geq \mathbf{p}(y) \\ 0 & \mathbf{p}(x) < \mathbf{p}(y) \end{cases} \tag{1}$$

The image patch is then encoded as the n_d-dimensional bit string:

$$f_{n_d}(\mathbf{p}) = \sum_{1 \leq i \leq n_d} 2^{i-1} \tau(\mathbf{p}; x_i, y_i). \tag{2}$$

The pairs of pixel coordinates (x_i, y_i) are drawn from a Gaussian distribution around the centre of the patch: $(\mathbf{X}, \mathbf{Y}) \sim$ i.i.d Gaussian$(0, \frac{1}{25}s^2)$, where $s \times s$ is the size of the patch. In the case of OpenCV implementation of BRIEF, these pairs of coordinates are then hard-coded and used across all configurations of the descriptor.

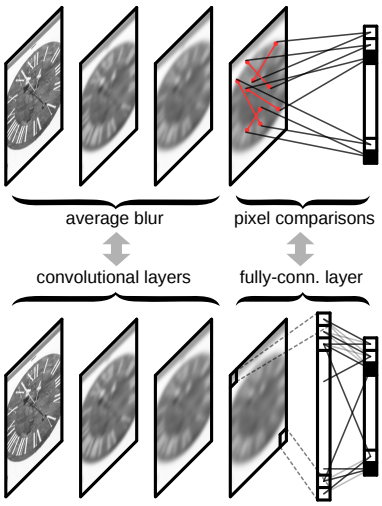
B. Autoencoders

Autoencoders [21] are unsupervised neural networks used for learning compact representations of data. An autoencoder consists of two parts, an encoder and a decoder, and is trained by setting the target output values to be equal to the input values, while imposing constraints on the middle layer. Formally, an autoencoder with encoder $\mathcal{E}$ and decoder $\mathcal{D}$ is trained to minimise the loss function J w.r.t. weights $\mathbf{w}_\mathcal{E}, \mathbf{w}_\mathcal{D}$ and biases $\mathbf{b}_\mathcal{E}, \mathbf{b}_\mathcal{D}$ of the encoder and decoder, respectively:

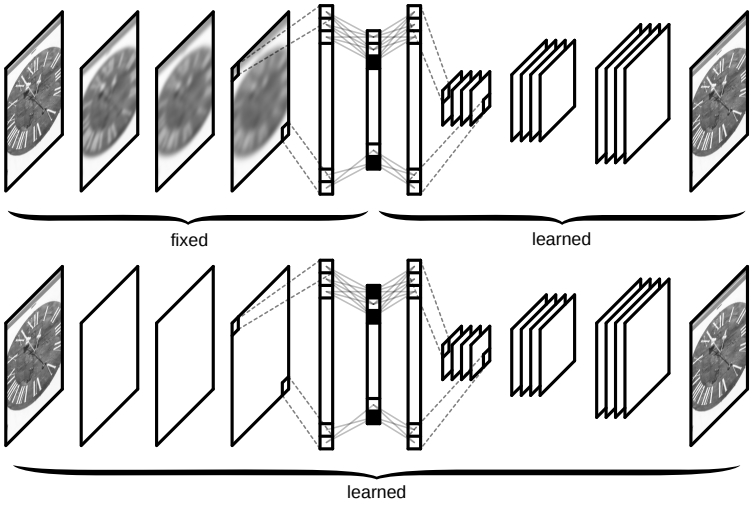
$$\min_{\mathbf{w}_\mathcal{E}, \mathbf{b}_\mathcal{E}, \mathbf{w}_\mathcal{D}, \mathbf{b}_\mathcal{D}} J(\mathcal{P}, \mathcal{E}_{\mathbf{w}_\mathcal{E}, \mathbf{b}_\mathcal{E}}, \mathcal{D}_{\mathbf{w}_\mathcal{D}, \mathbf{b}_\mathcal{D}}) = \tag{3}$$

$$\sum_{\mathbf{p} \in \mathcal{P}} \mathcal{L}(\mathbf{p}, \mathcal{D}_{\mathbf{w}_\mathcal{D}, \mathbf{b}_\mathcal{D}}(\mathcal{E}_{\mathbf{w}_\mathcal{E}, \mathbf{b}_\mathcal{E}}(\mathbf{p}))) \tag{4}$$

where $\mathbf{p} \in \mathcal{P}$ is a data sample (in our case an image patch) and $\mathcal{L}$ is some metric. Autoencoders working with image data usually consist of convolutional layers, with an optional fully-connected layer at the end of the encoder and the beginning of the decoder. The output neuron at location (i, j) in the k-th channel of the l-th convolutional layer is calculated as $x_{ij}^{(l,k)} = \sum_{c \in C} \sum_{u=1}^{s^{(l)}} \sum_{v=1}^{s^{(l)}} w_{uv}^{(l,k,c)} x_{(i+u)(j+v)}^{(l-1,c)} + b^{(l,k)}$, where C is the set of input channel indices, $w^{(l,k,c)}$ are the weights of the convolutional kernel for the l-th layer and k-th channel applied on the c-th input channel, $b^{(l,k)}$ is the bias for the l-th layer of the k-th channel, and $s^{(l)}$ is the size of the convolutional kernel for the l-th layer. The output of the i-th neuron of the l-th fully-connected layer is calculated as $x_i^l = \sum_{u=1}^{s(l)} (w_{iu}^l x_u^{l-1} + b_{iu}^l)$,



(a) Step 1 – implementing BRIEF as a CNN. Original BRIEF implementation (top) and our proposed as a CNN (bottom).



(b) Top: Step 2 – learning to invert a descriptor into its patch by training the decoder part of the autoencoder. The encoder is here set to be the network from step 1, implementing BRIEF. Bottom: Step 3 – training the whole autoencoder on a general dataset of patches.

Fig. 1. Visualisation of the proposed knowledge transfer and improvement framework from a BRIEF descriptor to an autoencoder-learned descriptor.

where w^l and b^l are the weight matrix and the bias vector of the l-th layer respectively, and $s(l)$ is the size of the l-th layer, i.e., the number of output neurons in the layer.

III. Proposed method

A. Framework for knowledge transfer from hand-crafted descriptors

The first contribution that we present in this paper is a framework for the transfer of knowledge from a hand-crafted descriptor to the unsupervised-learning–based descriptor. The framework we propose consists of four steps (steps 1–3 shown in Figure 1):

Step 1 – Construct and optimise a neural network $\mathcal{E}_{\mathbf{w}_\mathcal{E},\mathbf{b}_\mathcal{E}}$ that mimics patch encoding produced by the hand-crafted descriptor $\mathcal{H}$. For a set of patches $\mathcal{P}$ and loss function $\mathcal{L}$ where the loss function depends on the activation function chosen for the last layer, solve:

$$\min_{\mathbf{w}_\mathcal{E},\mathbf{b}_\mathcal{E}} J(\mathcal{P},\mathcal{E}_{\mathbf{w}_\mathcal{E},\mathbf{b}_\mathcal{E}}) = \sum_{\mathbf{p}\in\mathcal{P}} \mathcal{L}(\mathcal{H}(\mathbf{p}),\mathcal{E}_{\mathbf{w}_\mathcal{E},\mathbf{b}_\mathcal{E}}(\mathbf{p}))$$

Step 2 – Set $\mathcal{E}_{\mathbf{w}_\mathcal{E},\mathbf{b}_\mathcal{E}}$ to be the encoder $\mathcal{E}$ of an autoencoder, and train the decoder $\mathcal{D}_{\mathbf{w}_\mathcal{D},\mathbf{b}_\mathcal{D}}$, minimising the loss between its output and the input patch:

$$\min_{\mathbf{w}_\mathcal{D},\mathbf{b}_\mathcal{D}} J(\mathcal{P},\mathcal{D}_{\mathbf{w}_\mathcal{D},\mathbf{b}_\mathcal{D}}) = \sum_{\mathbf{p}\in\mathcal{P}} \mathcal{L}(\mathbf{p},\mathcal{D}_{\mathbf{w}_\mathcal{D},\mathbf{b}_\mathcal{D}}(\mathcal{E}(\mathbf{p})))$$

Step 3 – Train the whole autoencoder, consisting of the encoder $\mathcal{E}_{\mathbf{w}_\mathcal{E},\mathbf{b}_\mathcal{E}}$ and the decoder $\mathcal{D}_{\mathbf{w}_\mathcal{D},\mathbf{b}_\mathcal{D}}$:

$$\min_{\mathbf{w}_\mathcal{E},\mathbf{b}_\mathcal{E},\mathbf{w}_\mathcal{D},\mathbf{b}_\mathcal{D}} J(\mathcal{P},\mathcal{E}_{\mathbf{w}_\mathcal{E},\mathbf{b}_\mathcal{E}},\mathcal{D}_{\mathbf{w}_\mathcal{D},\mathbf{b}_\mathcal{D}}) = \sum_{\mathbf{p}\in\mathcal{P}} \mathcal{L}(\mathbf{p},\mathcal{D}_{\mathbf{w}_\mathcal{D},\mathbf{b}_\mathcal{D}}(\mathcal{E}_{\mathbf{w}_\mathcal{E},\mathbf{b}_\mathcal{E}}(\mathbf{p})))$$

Step 4 – Fine-tune the whole autoencoder on a dataset $\mathcal{P}_{Spec}$ specific to the desired application:

$$\min_{\mathbf{w}_\mathcal{E},\mathbf{b}_\mathcal{E},\mathbf{w}_\mathcal{D},\mathbf{b}_\mathcal{D}} J(\mathcal{P}_{Spec},\mathcal{E}_{\mathbf{w}_\mathcal{E},\mathbf{b}_\mathcal{E}},\mathcal{D}_{\mathbf{w}_\mathcal{D},\mathbf{b}_\mathcal{D}}) = \sum_{\mathbf{p}\in\mathcal{P}_{Spec}} \mathcal{L}(\mathbf{p},\mathcal{D}_{\mathbf{w}_\mathcal{D},\mathbf{b}_\mathcal{D}}(\mathcal{E}_{\mathbf{w}_\mathcal{E},\mathbf{b}_\mathcal{E}}(\mathbf{p})))$$

The first step is specific to the hand crafted descriptor that is chosen for knowledge transfer, and may involve creating specific neural network architectures that are functionally similar to the hand-crafted descriptor. In some cases, as we will show is the case with BRIEF in subsection III-B, it is possible to completely implement the hand-crafted descriptor using standard neural network layers (convolutional, fully-connected). In other cases, it may be necessary to make the network learn to output the descriptor using training. In this case, the network would be trained on a set of input patches until its output was sufficiently close to the output of the hand-crafted descriptor.

In the second step we train the decoder network to learn to reconstruct the patch from its encoding, similarly to what Mahendran et al. [22] did, but using a simple CNN without any regularisation. We found that, for reconstructing BRIEF, this setup achieves a good starting point for the training of the complete autoencoder, as we will discuss in subsection III-B.

Next, in the third step, we simultaneously train both the encoder and decoder parts of the autoencoder. We train the autoencoder on the same general dataset of patches used in both Step 1 and Step 2. In order for the learned descriptor to achieve rotation and translation invariance, it is possible to add geometric noise to the input patches while keeping the output patches the same. We further discuss the effect of the geometric noise adding in the Section IV.

In the final, fourth step, the autoencoder is fine-tuned from the previous step on an application-specific set of patches.

After the training is complete, the decoder part of the network can be discarded, and we are left with the encoder network, which is our local image descriptor.

In this paper, we demonstrate our framework using BRIEF as the hand-crafted descriptor from which we transfer the knowledge into a neural network.

B. Implementing learned BRIEF using the proposed framework

We now discuss how we implemented the learned BRIEF descriptor using the proposed framework.

We mostly focus on describing Step 1, since it is specific to the BRIEF descriptor. This step involves realising the original BRIEF descriptor as a convolutional neural network. In the case of BRIEF, in order to achieve a network that mimics the descriptor's output, this step does not necessitate training the network but simply setting the weights and biases of the network to match BRIEF's behaviour. The architecture of this neural network is comprised of several convolutional layers that implement BRIEF's average blurring, and one fully-connected layer that implements BRIEF's binary tests, i.e., the intensity comparisons between different pixels.

In the original implementation of BRIEF, the average blurring is performed with a 9×9 averaging kernel, i.e., in the deep learning terminology, a convolutional layer with one kernel of the size 9×9. In our implementation, we use four convolutional layers, each with one 3×3 averaging kernel, instead of a single layer with a 9×9 averaging kernel. This architecture is commonly used in modern CNNs and yields a sufficiently similar blurring to the blurring used in BRIEF. The activation functions of the convolution layers are rectifier linear units which have no effect at this point (since all the values of the kernel are greater than zero), but serve as non-linearity during the retraining of the network.

The binary pixel tests are implemented as a fully-connected layer (preceded by flattening the output from the previous convolutional layer in raster-scan fashion). Binary pixel test τ defined in (1) can be reformulated as:

$$[\sigma(\mathbf{p}(x) - \mathbf{p}(y))]\,,$$

where $\sigma(\cdot)$ is the sigmoid function and $[\cdot]$ denotes rounding to the nearest integer. The proof follows:

$$[\sigma(\mathbf{p}(x) - \mathbf{p}(y))] = \left[\frac{1}{1 + e^{-\mathbf{p}(x)+\mathbf{p}(y)}}\right] =$$
$$\begin{cases}[A], & A \in [0.5, 1) \quad \text{for} \quad \mathbf{p}(x) \geq \mathbf{p}(y) \\ [A], & A \in (0, 0.5) \quad \text{for} \quad \mathbf{p}(x) < \mathbf{p}(y)\end{cases} =$$
$$\begin{cases}1 & \mathbf{p}(x) \geq \mathbf{p}(y) \\ 0 & \mathbf{p}(x) < \mathbf{p}(y)\end{cases} = \tau(\mathbf{p}; x, y)$$

From this formulation, we can see that the binary test vector can be calculated with a fully-connected layer whose weight matrix $\mathbf{w}$ is built as follows. For all the binary tests $\tau(\mathbf{p}; x_i, y_i)$ indexed with $i, 1 \leq i \leq n_d$, we set the weights $w_{ix_i} = 1, w_{iy_i} = -1, w_{ij} = 0, j \neq x_i, j \neq y_i$. In other words, each neuron of the output of the fully-connected layer (i.e., each neuron of the binary code) is connected with exactly two neurons of the input of the fully-connected layer (two pixels on which the binary test will be applied), one with the weight $+1$, and the other with the weight -1, see Figure 1-a (bottom). The biases are set to 0. What is left is to set the activation function of this fully-connected layer to sigmoid, and the output of the whole network (when rounded to the nearest integer) will be the BRIEF descriptor of the input patch.

For Step 2, we use a decoder network consisting of a fully-connected layer (to mirror the network from Step 1), and several convolutional layers followed by upsampling until we reach the output of half of the size of the original patch. Learning to decode into a downsampled patch makes the autoencoder more resistant to Gaussian noise and, to some extent, to geometric noise. Step 3 involves training with artificial geometric noise added to the input patches while leaving the output patches the same. This should further increase descriptor's resilience to geometric noise such as rotation, translation, and shearing. Our results from the HPatches benchmark in the following section show that this step is quite effective in making our descriptor resistant to these kinds of geometric noise. The descriptor obtained from this step only differs in its weights from the BRIEF descriptor implemented as a CNN (from Step 1). In the final step, one would then apply further training with respect to a particular dataset, such as MRI images or multimodal macro photography. These applications, however, are beyond the scope of this paper and have been left to future work.

We set binary cross-entropy as the loss function $\mathcal{L}$. We use Adadelta optimiser for all neural networks in this paper. The networks are trained on a dataset of 80k 56×56 patches that were extracted from the images from the Imagenet dataset using FAST (Features from Accelerated Segment Test) algorithm for feature detection [23]. The ratio between training, validation, and test set is $8 : 1 : 1$. Our implementation is written in Keras and is publicly available.[1]

IV. Results

We evaluate the performance of our learned BRIEF in comparison with the original BRIEF descriptor, as implemented in the OpenCV library. The evaluation is performed on HPatches [1], a comprehensive benchmark for evaluating local image descriptors' performance on three different tasks (patch retrieval, image matching, and patch verification), each with varying difficulty levels ('easy', 'hard' and 'tough' – referring to the amount of geometric noise, as defined in [1]). We consider all tasks and difficulty levels. Table I summarises the results on all three tasks in terms of the mean Average Precision (mAP)[2], and in the following we discuss them more in detail.

[1]https://github.com/nimpy/learned-brief

[2]For a formal definition of mAP, we refer the reader to paper of Balntas et al. (author of the HPatches dataset) [1].

TABLE I
EVALUATION ON HPATCHES DATASET

	Original BRIEF	Learned BRIEF
Patch retrieval (mAP)	0.21	**0.29**
Image matching (mAP)	0.070	**0.081**
Patch verification (mAP)	**0.68**	**0.68**

The patch retrieval task tests how well a descriptor can match a query patch to a pool of patches extracted from many images, including many distractors. Figure 2 shows the performance of BRIEF and learned BRIEF for different sizes of test dataset (i.e., different sizes of the pools of patches in which the matching patches are searched for), comparing performances on all three difficulty levels. Our learned BRIEF shows improvement over the original BRIEF for each difficulty and for each size of pool of patches. Based on the results from the benchmark (as shown in Figure 2), we conclude that learned BRIEF outperforms BRIEF by a constant margin irrespective of the amount of geometric noise. The most influential factor is rather the size of the pool of patches – we observe the largest improvement on smaller pools that are typical in tasks such as panorama stitching, object tracking, and inpainting. Learned BRIEF remains superior to BRIEF also on large pools of patches and after a certain pool size (above $\sim$10000 patches) the improvement stabilises.

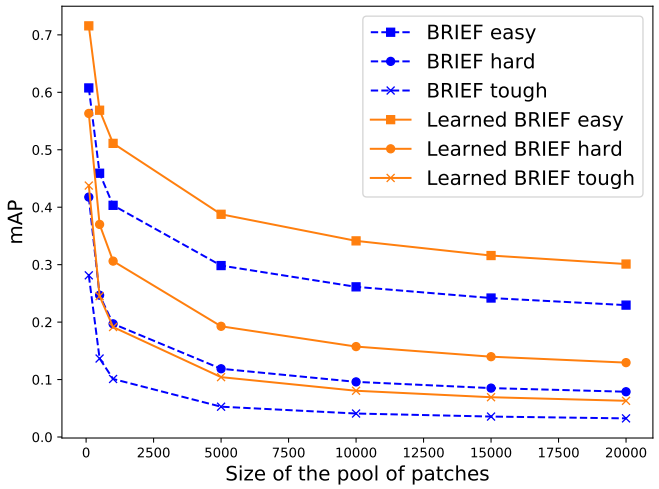


Fig. 2. Performance comparison on patch retrieval task between BRIEF and learned BRIEF.

The image matching task tests to what extent a descriptor can correctly identify correspondences in two images based on a pair of patches – one patch from each of the images. Our results for this task are shown in Figure 3. Similar to the previous task, the amount of geometric noise is varied. The results show that learned BRIEF also outperforms BRIEF on the image matching task. We observe again that the amount of geometric noise does not influence the improvement. In all cases, learned BRIEF outperformed the original BRIEF.

The patch verification task measures the ability of a descriptor to classify whether two patches match, i.e., whether

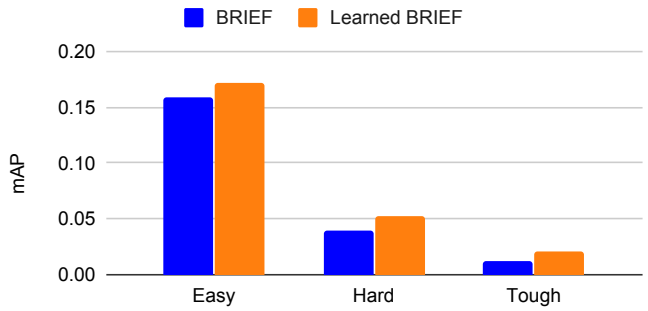


Fig. 3. Performance comparison on image matching task between BRIEF and learned BRIEF.

they are extracted from the same measurement, as defined in the benchmark. On this task, we observe equal performance of both descriptors.

We show in Figure 4 examples of retrieved patches, comparing learned BRIEF with BRIEF. These patches have been taken from a subset of the HPatches dataset. We compare the encoding of a query patch (showed larger in the figure), with the encodings of other patches, using both the original BRIEF descriptor and our learned BRIEF. We have observed that in most cases, the first patch retrieved by our descriptor is a much better match than the first patch retrieved by the original BRIEF. Furthermore, we have observed that the original implementation generally retrieves a visually dissimilar patch within the first five retrieved patches. This characteristic was not present in learned BRIEF.

V. CONCLUSION AND FUTURE WORK

In brief, we introduced in this paper a framework for transferring knowledge from a hand-crafted descriptor to an unsupervised-learning–based descriptor. We demonstrated the use of this framework by creating the learned BRIEF descriptor based on the BRIEF hand-crafted descriptor. Furthermore, we proposed an elegant implementation of BRIEF as a convolutional neural network. Using HPatches benchmark for evaluating local image descriptors, we showed that our learned BRIEF descriptor outperforms consistently the original BRIEF.

This paper acts as a proof of concept of our framework for knowledge transfer from hand-crafted descriptors, showing that a learned descriptor created in this way can outperform its hand-crafted counterpart. In the future, we will investigate this framework's applicability to other hand-crafted descriptors, including the widely used SIFT descriptor, which is still among the best performing ones.

REFERENCES

[1] V. Balntas, K. Lenc, A. Vedaldi, and K. Mikolajczyk, "HPatches: A benchmark and evaluation of handcrafted and learned local descriptors," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2017, pp. 5173–5182.

[2] P. Fischer, A. Dosovitskiy, and T. Brox, "Descriptor matching with convolutional neural networks: a comparison to SIFT," *arXiv preprint arXiv:1405.5769*, 2014.

[3] J. L. Schonberger, H. Hardmeier, T. Sattler, and M. Pollefeys, "Comparative evaluation of hand-crafted and learned local features," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2017, pp. 1482–1491.

[4] K. He, Y. Lu, and S. Sclaroff, "Local descriptors optimized for average precision," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2018.

[5] N. Papernot, M. Abadi, U. Erlingsson, I. Goodfellow, and K. Talwar, "Semi-supervised knowledge transfer for deep learning from private training data," *arXiv preprint arXiv:1610.05755*, 2016.

[6] M. T. McCann, K. H. Jin, and M. Unser, "Convolutional neural networks for inverse problems in imaging: A review," *IEEE Signal Processing Magazine*, vol. 34, no. 6, pp. 85–95, 2017.

[7] M. Calonder, V. Lepetit, C. Strecha, and P. Fua, "BRIEF: Binary robust independent elementary features," in *European conference on computer vision*. Springer, 2010, pp. 778–792.

[8] D. G. Lowe, "Object recognition from local scale-invariant features," in *ICCV*. IEEE, 1999, p. 1150.

[9] N. Dalal and B. Triggs, "Histograms of oriented gradients for human detection," in *2005 IEEE computer society conference on computer vision and pattern recognition (CVPR'05)*, vol. 1. IEEE, 2005, pp. 886–893.

[10] H. Bay, A. Ess, T. Tuytelaars, and L. Van Gool, "Speeded-up robust features (surf)," *Computer vision and image understanding*, vol. 110, no. 3, pp. 346–359, 2008.

[11] E. Rublee, V. Rabaud, K. Konolige, and G. R. Bradski, "ORB: An efficient alternative to SIFT or SURF." in *ICCV*. IEEE, 2011, pp. 2564–2571.

[12] S. Zagoruyko and N. Komodakis, "Learning to compare image patches via convolutional neural networks," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2015, pp. 4353–4361.

[13] E. Simo-Serra, E. Trulls, L. Ferraz, I. Kokkinos, P. Fua, and F. Moreno-Noguer, "Discriminative learning of deep convolutional feature point descriptors," in *Proceedings of the IEEE International Conference on Computer Vision*, 2015, pp. 118–126.

[14] V. Balntas, E. Riba, D. Ponsa, and K. Mikolajczyk, "Learning local feature descriptors with triplets and shallow convolutional neural networks." in *Proceedings of the British Machine Vision Conference (BMVC)*, 2016, pp. 119.1–119.11. [Online]. Available: https://dx.doi.org/10.5244/C.30.119

[15] Y. Ono, E. Trulls, P. Fua, and K. M. Yi, "LF-Net: learning local features from images," in *Advances in neural information processing systems*, 2018, pp. 6234–6244.

[16] Y. Tian, X. Yu, B. Fan, F. Wu, H. Heijnen, and V. Balntas, "SOSNet: Second order similarity regularization for local descriptor learning," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2019, pp. 11 016–11 025.

[17] Q. Wang, X. Zhou, B. Hariharan, and N. Snavely, "Learning feature descriptors using camera pose supervision," *arXiv preprint arXiv:2004.13324*, 2020.

[18] L. Chen, F. Rottensteiner, and C. Heipke, "Feature descriptor by convolution and pooling autoencoders," *International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences-ISPRS Archives 40 (2015), Nr. 3W2*, vol. 40, no. 3W2, pp. 31–38, 2015.

[19] N. Žižakić, I. Ito, and A. Pižurica, "Learning local image descriptors with autoencoders," in *Proc. IEICE Inform. and Commun. Technol. Forum ICTF 2019*, 2019.

[20] N. Žižakić, I. Ito, L. Meeus, and A. Pižurica, "Autoencoder-learned local image descriptor for image inpainting," in *BNAIC/BENELEARN 2019*, vol. 2491, 2019.

[21] G. E. Hinton and R. R. Salakhutdinov, "Reducing the dimensionality of data with neural networks," *science*, vol. 313, no. 5786, pp. 504–507, 2006.

[22] A. Mahendran and A. Vedaldi, "Understanding deep image representations by inverting them," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2015, pp. 5188–5196.

[23] E. Rosten and T. Drummond, "Machine learning for high-speed corner detection," in *European conference on computer vision*. Springer, 2006, pp. 430–443.

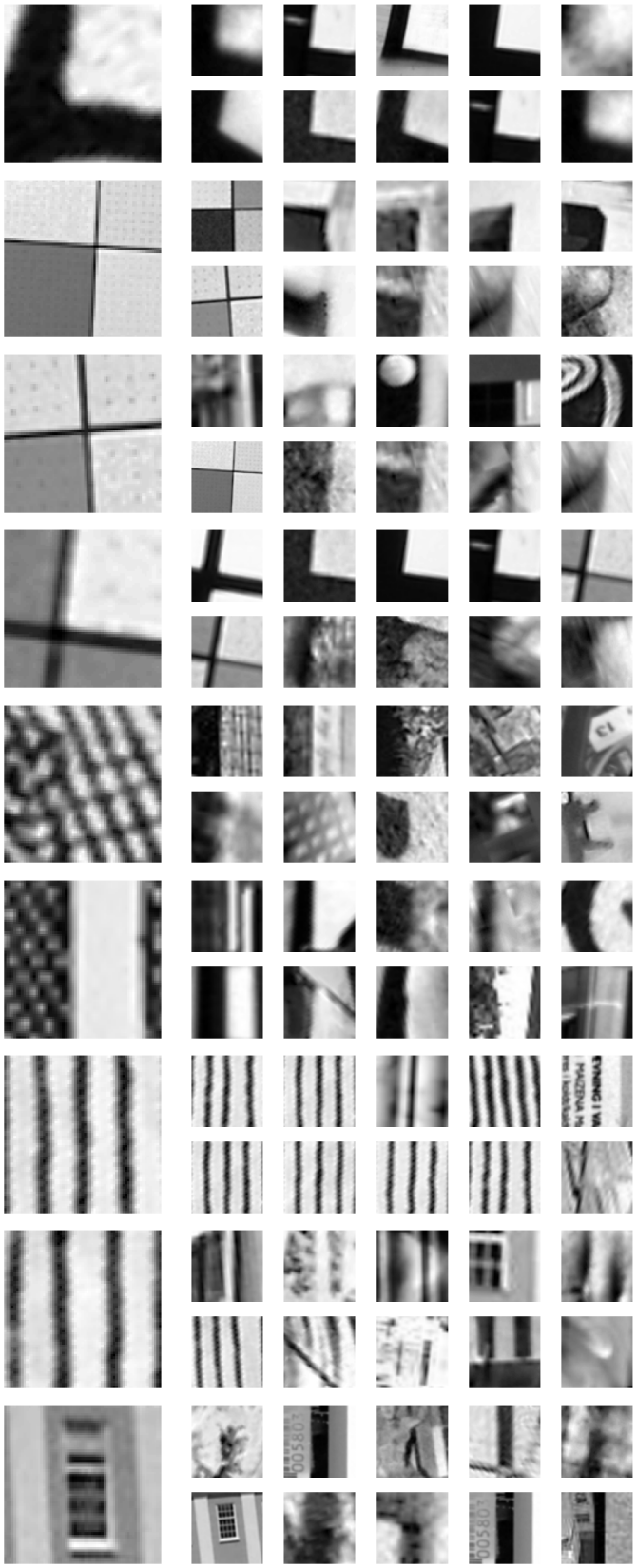


Fig. 4. Patch retrieval examples. Large patch is the query patch. Top rows: original BRIEF descriptor; bottom rows: learned BRIEF.