Learnable Descriptors for Visual Search

Andrea Migliorati, Attilio Fiandrotti, Gianluca Francini, Riccardo Leonardi

Learnable Descriptors for Visual Search

Andrea Migliorati, *Member, IEEE*, Attilio Fiandrotti, *Senior Member, IEEE*, Gianluca Francini, and Riccardo Leonardi, *Fellow, IEEE*

***Abstract*— This work proposes LDVS, a learnable binary local descriptor devised for matching natural images within the MPEG CDVS framework. LDVS descriptors are learned so that they can be sign-quantized and compared using the Hamming distance. The underlying convolutional architecture enjoys a moderate parameters count for operations on mobile devices. Our experiments show that LDVS descriptors perform favorably over comparable learned binary descriptors at patch matching on two different datasets. A complete pair-wise image matching pipeline is then designed around LDVS descriptors, integrating them in the reference CDVS evaluation framework. Experiments show that LDVS descriptors outperform the compressed CDVS SIFT-like descriptors at pair-wise image matching over the challenging CDVS image dataset.**

***Index Terms*— Pair-wise image matching, patch matching, binary descriptors, convolutional neural networks, fully convolutional neural networks.**

I. Introduction

THE MPEG CDVS (Compact Descriptors for Visual Search) standard [1] aims at low-complexity, bitrate-efficient, large scale image matching and retrieval over mobile devices. Preliminarily, SIFT descriptors [2], [3] are extracted from the image and compressed as binary *local descriptors*. Each image is represented as a set of compressed descriptors, the number and size depending on the target descriptor bitrate, i.e. the *encoding mode*. Then, compressed descriptors are combined into a *global descriptor*. The ensemble of local descriptors and global descriptor extracted from an image forms the image *CDVS descriptor*. Images can be matched comparing the relative CDVS descriptors in multiple ways, such as matching pairs of local descriptors comparing Hamming distances followed by a geometric consistency check.

Manuscript received December 4, 2019; revised July 20, 2020; accepted September 26, 2020. Date of publication October 23, 2020; date of current version November 19, 2020. This work was supported by a fellowship from TIM S.p.A. (Telecom Italia). The associate editor coordinating the review of this manuscript and approving it for publication was Dr. Marta Mrak. *(Corresponding author: Andrea Migliorati.)*

Andrea Migliorati is with the Dipartimento di Elettronica e Telecomunicazioni, Politecnico di Torino, 10138 Turin, Italy, and also with the Department of Information Engineering, Università degli Studi di Brescia, 25123 Brescia, Italy (e-mail: andrea.migliorati@polito.it).

Attilio Fiandrotti is with the Dipartimento di Informatica, Università di Torino, 10149 Torino, Italy, and also with the LTCI, Télécom Paris, Institut Polytechnique de Paris, 91120 Palaiseau, France (e-mail: attilio.fiandrotti@unito.it).

Gianluca Francini is with Telecom Italia, Joint Open Lab, 10128 Turin, Italy (e-mail: gianluca.francini@telecomitalia.it).

Riccardo Leonardi is with the Department of Information Engineering, Università degli Studi di Brescia, 25123 Brescia, Italy (e-mail: riccardo.leonardi@unibs.it).

Digital Object Identifier 10.1109/TIP.2020.3031216

Recent research investigated the potential of deep convolutional architectures to learn global descriptors. For example, the approach in [4] consists of learning a global image hash code via a convolutional neural network. In [5] it is proposed to learn a global descriptor via a trainable neural network functions which aggregate the local descriptors into a global representation. In [6] the compression of a global CDVS descriptor is formulated as a resource-constrained optimization problem. Recently, MPEG has standardized the CDVA (Compact Descriptors for Video Analysis) technology, a successor to CDVS tackling image retrieval in video sequences. In the CDVA standard [7], CDVS local and global descriptors are complemented with a global descriptor called Nested Invariance Pooling (NIP) learned using a convolutional architecture [8].

Far less attention has been devoted to the challenging problem of how to learn discriminative, compact, local descriptors, and how to possibly integrate them in the MPEG CDVS framework. Existing research has either focused on GPU-accelerating SIFT descriptors extraction [9] or on learning SIFT-like descriptors in a supervised framework [10]. SIFT-like learned descriptors retain desirable properties such as invariance to rotations, illumination, and perspective changes [11]. Interesting results are reported when pairs or triplets of patches are jointly encoded and a decision network is trained to learn an appropriate distance metric [12], [13]. While complete image matching pipelines based on a deep learning framework such as [14] go beyond the scope of the present work, it shows the potentials of image matching architectures based on deep learning frameworks.

Therefore, it is not clear how a learned local descriptor would perform when compared to an efficient standardized solution such as MPEG CDVS. Namely, existing learnable descriptors are usually designed with few constraints. On the contrary, the CDVS standard mandates the use of binary descriptors that shall be compared via Hamming distance within a SIFT-tailored geometry consistency framework. In detail, it is not clear how such approaches would perform when a geometric consistency check framework is introduced in the processing chain. Finally, little if no attention has been posed to the problem of keeping the descriptor complexity under control, as done in the CDVS standard.

This work[1] proposes a pair-wise image matching architecture designed around learnable local binary descriptors

[1]Trained models to reproduce results are available at https://bit.ly/LDVS-repository

(*LDVS* descriptors in the following). We introduce a fully convolutional architecture for learning binary local descriptors in a supervised way. Our design specifically allows for the learning of real-valued descriptors that can be sign-quantized at deployment time, resulting in binary descriptors that lean to the same performance at matching as the float ones, while allowing for comparison via Hamming distance as required by the CDVS standard. Furthermore, fewer parameters make computing LDVS descriptors simpler over memory-constrained devices as per CDVS design goals. Then, we design a complete pipeline for pair-wise image matching around LDVS descriptors that retains the key components of the MPEG CDVS standard, effectively integrating the proposed learned descriptors in the CDVS standard. This pipeline includes a geometric consistency check and allows us to compare with CDVS on an identical descriptor matching context.

Patch matching experiments over two distinct datasets show favorable performance over comparable learned binary descriptors despite the design of the latter is not standard-constrained as in our case. Concerning image matching, LDVS descriptors improve over CDVS ones according to multiple performance metrics mainly in reason of the fewer false matches. To the best of our knowledge, this work is the first to accurately evaluate the benefits of learned local descriptors when dropped inside the MPEG CDVS reference pipeline.

II. Related Works

Recently, advances in deep learning showed one can train a neural network to learn and compare images via local descriptors in an automated way without necessarily relying on handcrafted techniques. A considerable amount of learned descriptors such as LIFT [14], MatchNet [15], HardNet [16], and many others [11]–[13], [17]. However, these designs have not gained momentum in practical applications. Hence, handcrafted local descriptors such as SIFT and its variants are as relevant as in the previous years, as shown in recent literature such as [18]. [19], [20] report that handcrafted local descriptors significantly outperform learned ones in the field of small-scale retrieval and pair-wise image matching, and 3D reconstruction respectively. Further, by cross-checking a huge number of references, authors in [19], report several ambiguities and inconsistencies when comparing local descriptors' performance.

Ultimately, it is not clear which approach is best between handcrafted and learned descriptors, especially because of the lack of an unequivocal, meaningful evaluation framework in which the two methods can be compared within the same matching pipeline. In this scenario, an increasing number of binary descriptors have been proposed. In particular, binary descriptors allow for fast comparison via Hamming distance, and deployment in limited-resources applications thanks to their improved rate and storage efficiency. A list of seminal works would include early descriptors such as BRIEF [21], BRISK [22], FREAK [23], and ORB [24].

Learned binary descriptors became ever more relevant thanks to works such as BinBoost [25], in which performance at patch matching is boosted by learning a set of projection matrices, and Supervised Discrete Hashing (SDH) [26], in which binary codes are learned to minimize the classification loss of a linear classifier. More recently, other deep binary descriptor learning approaches have been devised [27]–[33], achieving state-of-the-art performance. A relevant approach is the one by [34], in which two asymmetric and complementary descriptors are extracted from the convolutional domain and fused to constitute the local descriptor for each patch, however at the expense of a feature extraction pipeline with doubled complexity. Other recent works also include the unsupervised methods DeepBit [35], GraphBit [36], and BinGAN [37], respectively introducing a set of non-linear projection functions, a regularization method for Generative Adversarial Networks, and a directed acyclic graph. Although unsupervised approaches generalize well over different application domains, with this work we focus on supervised methods as they ensure stable results when matching natural images.

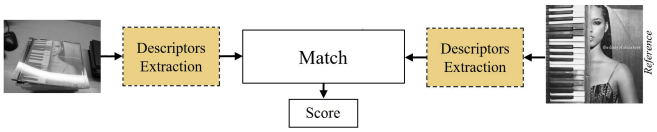

Fig. 1. MPEG CDVS pair-wise image matching procedure. Dashed boxed represent normative parts of the standard.

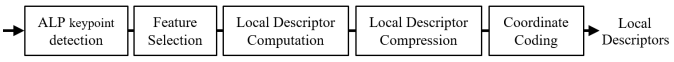

Fig. 2. Normative process for extracting local descriptors from a single image in the MPEG CDVS standard.

III. The MPEG CDVS Standard

The MPEG Compact Descriptors for Visual Search (CDVS) standard regulates methods for efficient compression of SIFT descriptors. As this work focuses on learning local descriptors, we will not consider the global descriptor defined in the CDVS standard. While we refer the reader to [38], [39] for a complete overview of the standard, hereafter we focus on pair-wise image matching via local descriptors only. We recall that pair-wise matching consists of determining whether a *query* image depicts the same objects or scene of a *reference* image as illustrated in Fig. 1. Notice that as for other MPEG standards, only the descriptors extraction procedures are normative (dashed boxes), whereas the matching procedures are meant for reference.

1) Descriptors Extraction: The *Descriptors Extraction* process is illustrated in Fig. 2 for query and reference images. First, robust scale-invariant keypoints are located via the ALP method [1]. Next, detected keypoints are ranked on a relevance basis according to a statistical model accounting for parameters such as scale (σ), orientation (θ), position (x, y), and the number of selected interest points [40]. The number of retained keypoints depends on the target descriptor bitrate or *CDVS encoding mode*, where higher encoding modes entail more retained keypoints.

Then, a SIFT descriptor is computed over a squared image *patch* centered around each selected keypoint. Next, each

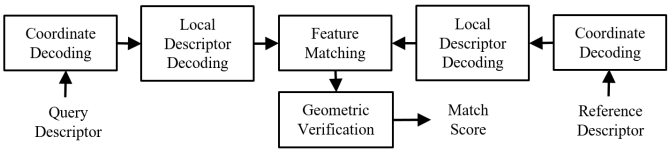

Fig. 3. MPEG CDVS reference procedure for pair-wise image matching using the compressed binary local descriptors.

SIFT descriptor undergoes a set of transformations based on linear combinations of the bin values, followed by a scalar quantization of the obtained values. The quantized values are then compressed via variable-length coding while the keypoint coordinates are quantized and arithmetically encoded. In the following, we will use the term *CDVS descriptor* to indicate a set of compressed SIFT descriptors, i.e. to indicate a set of compressed binary *local* descriptors. According to the *encoding mode*, i.e. the target CDVS descriptor bitrate, the number and the bitrate of the compressed SIFT descriptor in the CDVS descriptor changes accordingly.

2) Image Matching: Two images I_1 and I_2 (e.g., query image and reference image) are matched comparing the relative CDVS descriptors as explained in Fig. 3. First, each compressed local descriptor is decoded along with the relative keypoint coordinates. Then, decoded descriptors from the query and reference images are matched with a two-way approach. The local descriptors are compared via the Hamming distance: the ratio between the closest distance and the next closest distance, denoted as r, is used as a criterion for distinctiveness to determine keypoint correspondences. If r exceeds a given threshold Λ_{CDVS} (*ratio test threshold* [2]), then the two keypoints are considered a matching pair. For each matching pair, a score is computed as $\beta \triangleq \cos(\pi\ r/2)$ as suggested in the informative part of the standard. Then, based on the overlapping set of matching pairs that are detected using a cross-referencing approach (query to reference and reference to query), a matching score s is computed. Finally, whether the images are a matching pair will be decided based on the cumulative effect of matching pairs of keypoints. For matching keypoints that are present in both directions, s is computed as

$$s \triangleq \left(\frac{\beta_{1\to 2} + \beta_{2\to 1}}{2}\right). \tag{1}$$

A geometric consistency check follows to separate *inlier* from *outlier* pairs of matching keypoints. Inliers have matched descriptors that refer to the same physical objects. Outliers instead correspond to matching descriptors that are not geometrically consistent between the images. The higher the number of inliers for a pair of matching images, the stronger and stable the match. The method should also be able to handle complex cases where the same object appears skewed, distorted, or under different illumination conditions. Similarly, the lower the number of outliers that survive the geometric consistency check, the more reliable is the prediction on whether the image pair is a match or not. In detail, MPEG CDVS adopts DISTRAT [41], a technique based on the histogram of logarithmic distance ratios (LDR) for pairs of matches. The inliers matching total score S between the two images is computed accumulating each *s-th* score of the local feature matches found to be inlier as:

$$S \triangleq \sum_{inliers} s. \tag{2}$$

In particular, one can apply a threshold on the S score to determine if the query and the reference images are matching, hence they refer to the same physical object.

IV. Proposed Architecture

This section describes how to train a neural network to generate pairs of variable length suitable to determine the correspondence between image patches. The method for matching pairs of images via binary descriptors that is our ultimate goal will be instead described in Sec. V.

We propose a network architecture for patch matching that relies on a *Siamese* topology, commonly used in problems where a distance metric function between fixed-size representations of equally-sized signals [42] is employed. In the simplest case, a Siamese network is composed of two identical *feature extraction subnetworks* with shared learnable parameters to reduce the network complexity. Each subnetwork generates a representation of the input signal through a fixed-size vector of features as output. Due to parameter sharing, the two subnetworks represent the same transfer function from the input signal to the output feature space. Typically, a (learnable) *metric function* compares the two feature vectors and outputs a distance between the vectors, representing the (dis)similarity between the input signals. Thanks to the target (dis)similarity metric, Siamese networks are trained end-to-end in a fully supervised way to learn representations of the input signals (descriptors, in the following) that maximize or minimize some (learnable) feature distance function.

A. Network Architecture

Fig. 4 illustrates our network architecture for encoding image patches as local descriptors. We will describe our network in terms of *convolutional subnetwork for feature and descriptor extraction* and an appropriate distance metric function for training such a network.

1) Descriptor Extraction Architecture: The two feature extraction subnetworks receive as input a pair of identically sized image patches (P_1, P_2). Each subnetwork is composed by M *convolutional modules*, where each k-th module m_k ($k \in 1, \ldots, M$) consists of one convolutional layer, a hyperbolic tangent (TanH) as activation function, and a 2×2 max-pooling layer. Concerning the size of the filters, m_1 is composed of 5×5 filters (kernels), while the following modules m_{i+1} to m_M include 3×3 filters. Convolutions are all of the wide type with one-pixel stride and zero-padding at the border. Let f_k indicate the number of filters in the convolutional layer of the k-th module, i.e. the number of feature maps output by the k-th convolutional layer. The number of filters doubles at every convolutional layer as in $f_{k+1} = 2\ f_k$, so the number of *feature maps* produced in output by each convolutional layer doubles at each module.

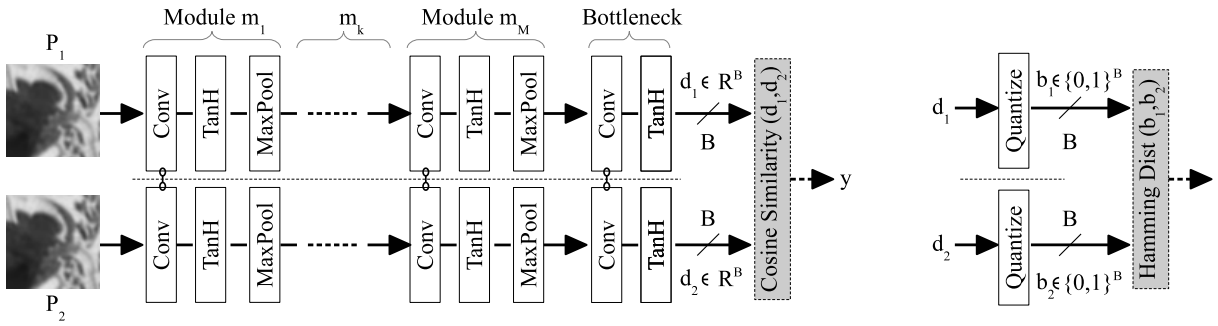

Fig. 4. Left: the proposed neural network architecture for matching pairs of image patches (P_1, P_2). The network is trained at generating real-valued descriptors (d_1, d_2) that separate pairs of matching patches from non-matching pairs by minimizing a cosine similarity metric between the network output y and the ground truth t. The cosine similarity is only instrumental for training. Right: descriptors (d_1, d_2) are sign-quantized at the time of deploying the trained network producing the sought binary descriptors (b_1, b_2) that can be compared using the Hamming distance as per MPEG CDVS standard.

Let us now indicate as FO_k the number of features output by each module, computed as $f_k \times w_k \times h_k$, where w_k and h_k are the width and the height of the feature maps output by the module. In the following, we will refer to FMR as feature map resolution such that $FMR = w_k \times h_k$.

Since the pooling layer reduces both w_k and h_k at each k-th module, the number of features output at each modules decreases by a two-factor, i.e. $FO_{k+1} = \frac{FO_k}{2}$. That is, each module projects the input patch over a smaller space of latent features which produces a compact patch descriptor. Notice that by controlling the total number of convolutional modules M, different trade-offs between semantic level and spatial detail of the feature maps are possible, as experimentally shown later on.

Each subnetwork includes one *bottleneck* module which reduces the dimensionality of the extracted features to B elements. The bottleneck module is indicated as the $(M+1)$-th convolutional module in the figure since it is implemented as a convolutional layer with $f_{M+1} = B$ filters sized 3×3 followed by a hyperbolic tangent activation function. Implementing the bottleneck layer as a convolutional rather than a fully connected layer reduces the network complexity. The hyperbolic tangent activation function limits the bottleneck module outputs between -1 and 1. This is consistent with the choice of the distance metric function used to train the network illustrated later on. Finally, the output of the two Siamese subnetworks in the figure is a pair of B-elements real-valued patch descriptors (d_1, d_2), where the number of filters in the bottleneck layer B controls the *rate* of the descriptors. The scheme for binarizing the (d_1, d_2) real-valued descriptors will be described in the following.

2) Training Distance Metric: We recall that our goal is to design a network able to generate patch descriptors (d_1, d_2) that can be trained end-to-end to match pairs of image patches. Towards this end, we need a function suitable to account for the distance between a pair of real-valued descriptors (d_1, d_2) in the descriptors space. Works such as [12], for example, rely on one or more fully connected layers to learn an appropriate distance metric function. That is, a large number of extra learnable parameters are required for learning the distance metric. Thus, we propose to evaluate the similarity between d_1 and d_2 via the function

$$C(d_1, d_2) \triangleq \frac{\langle d_1, d_2 \rangle}{\|d_1\| \cdot \|d_2\|} = \frac{\sum_{j=1}^{B} d_{1,j} d_{2,j}}{\sqrt{\sum_{j=1}^{B} d_{1,j}^2} \sqrt{\sum_{j=1}^{B} d_{2,j}^2}} \quad (3)$$

referred to as *cosine similarity* and whose range lies in $[-1, 1]$. Such function enjoys several useful properties such as the fact it is always continuous and differentiable thus allowing a fully supervised, end-to-end training of the network. Most important, the affinity of the cosine similarity with the Hamming distance [43] enables descriptor binarization and comparison via Hamming function, as mandated by the CDVS standard. Concluding, given a pair of patches (P_1, P_2), the network is trained to output the cosine similarity $y = C(d_1, d_2)$ according to the procedure described in the following. The real-valued descriptors (d_1, d_2) learned as above are binarized at inference time and compared using the Hamming distance as mandated by the CDVS standard as described in the following.

B. Training Procedure

The architecture described in the previous sections is trained to minimize the quadratic error between the cosine similarity y computed over a pair of patches and the corresponding ground truth label t (i.e. matching or non-matching pair). Let us define the i-th pair of identically sized patches $x_i = (P_1^i, P_2^i)$ as the i-th *training sample*. P_1^i and P_2^i are a pair of *matching* patches or, equivalently, x^i is a matching sample if P_1^i and P_2^i represent the same detail; P_1^i and P_2^i are a pair of *non-matching* patches or, equivalently, x^i is a non-matching sample otherwise. Let us define $y^i = C(d_1^i, d_2^i)$ as in Eq. 3 and t^i respectively as the network output and the corresponding *target* output for x^i. Our goal is to learn the network parameters w enabling the network to generate the descriptors pair (d_1^i, d_2^i) such that $y^i = t^i$. To this end, we decide to encode the target network output t^i as follows. In the case x^i is a matching sample, the network shall generate (d_1^i, d_2^i) so that $C(d_1^i, d_2^i) \approx 1$: therefore, we impose $t^i = 1$ for matching samples. Concerning non-matching samples, we impose $t^i = 0$ since non-matching patches (P_1^i, P_2^i) is equivalent to measuring the cosine similarity between random i.i.d. vectors (d_1^i, d_2^i). So, our learned (d_1^i, d_2^i) will be *spread-out* descriptors [44], such that the probability of having

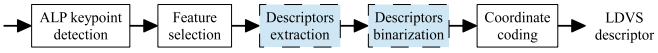

Fig. 5. The procedure for extracting LDVS descriptors from an image.

almost orthogonal descriptors by randomly sampling pairs of non-matching samples is close to one. For each i-th training sample, we optimize the network parameters to minimize a $L(l_i, y_i)$ loss function which is defined as the quadratic error between target t^i and actual network output y^i as

$$L(t^i, y^i) = (t^i - C(d_1^i, d_2^i))^2 \quad (4)$$

recasting our classification between matching and non-matching patches as a l_2 regularized regression problem. As a side remark, we also experimented with a Hinge loss function which offers the possibility to optimize with a margin m. However, our practical experiments showed no appreciable performance gain, whereas the choice of a suitable m value proved to be an additional problem per se.

Considering the practical aspects of the training procedure, the gradient of the loss function is minimized via Stochastic gradient descent and is computed via error gradients back-propagation for all hidden layers. We observed that preliminary normalizing the input patches over their own l_2 norm increases the network robustness and generalization capacity with respect to changes in illumination conditions. Notice that as a preliminary step, we normalize the input patches with respect to the mean pixel intensity and standard deviation values computed over the entire training set, as commonly practiced in the state-of-art approaches. We also experimentally observed that Spatial Batch Normalization [45] improves both performances as well as convergence speed. Finally, we found that applying spatial Dropout with probability $p = 0.5$ before each M-th convolutional layer improves the network generalization ability. In our experiments in Sec. VI, we trained the network from scratch for each of the considered B values in order to be able to find optimal performance.

V. Image Matching Pipeline

In this section, we describe an image matching pipeline designed around the proposed binary patch descriptors. First, we detail the process of extracting an LDVS descriptor from an arbitrary image. Then, we will detail the procedure to match two images by comparing the relative LDVS descriptors.

A. LDVS Binary Descriptors Extraction

Fig. 5 illustrates our proposed pipeline for extracting LDVS descriptors from a natural image (dashed boxes represent procedures differing from the standard CDVS counterpart shown in 2). As per the CDVS standard, robust scale-invariant keypoints are located in the image using the ALP algorithm and are ranked on a relevance basis selecting a subset thereof depending on the descriptor mode as described in Sec. III. Next, we extract a 64×64 image patch around each retained keypoint as follows. For a given keypoint, let σ be the relative scale, (x, y) its position inside the image, and θ its dominant orientation. A grayscale (8 bit) patch image centered around (x, y), with a rotation angle equal to θ, inscribed into a circle with radius set to $3.96 \cdot 2 \cdot \sqrt{2} \cdot \sigma$ (where σ is the scale of the keypoint) is sampled with a step of 1/64th of the patch size, so that each patch has 64×64 pixels. Bilinear interpolation is used to find the pixel values after alignment with the patch rotation angle. The pixel content inside each patch is extracted from a Gaussian filtered version of the image as done in CDVS during the compact descriptor computation steps. Next, each 64×64 patch is provided in input to the trained network illustrated in Fig. 4.

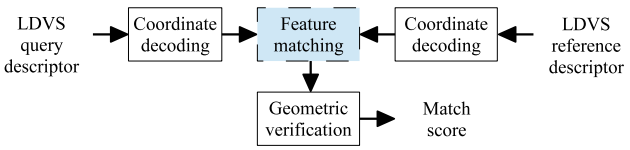

Fig. 6. Procedure for matching two images comparing the respective LDVS descriptors.

We recall that the two Siamese subnetworks share the same parameters, thus providing a patch into any of the convolutional subnetworks yields in output a B-elements descriptor. However, the descriptor produced by the subnetwork is a vector of real-valued elements, whereas binary descriptors are needed to achieve descriptor rates comparable to CDVS. Under the hypothesis (experimentally verified) that the network generates descriptors with zero-mean elements, each of the B-elements real-valued descriptor d is quantized to its sign value over a single bit producing a B-bits descriptor b. Finally, the coordinates of each keypoint are compressed according to the CDVS standard as discussed in Sec. III-.1. For the sake of clarity and borrowing from the CDVS terminology, we refer to each pair of a learnable binary descriptor of B bits and relative compressed coordinates as a *learnable feature*.

Summarizing, image I is encoded as a set of learnable features, one feature for each detected and selected keypoint. Each feature includes the proposed learnable binary descriptor and its compressed coordinates. The set of features describing image I is refereed to as the LDVS descriptor of the image. The actual number of features in each LDVS descriptor and the size of each binary descriptor in each feature will depend on the actual coding mode, as in MPEG CDVS.

B. Image Matching

We detail here the procedure to match two images I_1, I_2 via the relative LDVS descriptors as depicted in Fig. 6. Preliminary, for each feature in the LDVS descriptor, keypoint coordinates are decoded recovering the (x, y) parameters associated with each binary descriptor, as per CDVS standard. At this point, each image is represented as a set of features, i.e. learned binary descriptors with relative coordinates.

The feature matching block compares query and reference images features following the principles of the two-way CDVS matching scheme recalled in Sec. III-.2. Namely, for each binary descriptor b_1 in the query LDVS descriptor, we calculate its Hamming distance with respect to each descriptor

b_2 in the reference LDVS descriptor and vice-versa as

$$\bar{H}(b_1, b_2) = \frac{1}{B} H(b_1, b_2). \quad (5)$$

such distance will range from 1 for non-matching pairs, i.e. for patches representing different details, to 0 for perfectly identical pairs of patches, i.e. patches representing most likely the same image detail. As per CDVS standard, if the ratio r between first closest neighbor distance and second closest neighbor exceeds the threshold Λ_{LDVS}, then b_1 and b_2 are considered a match.

We recall that the network in Fig. 4 has been trained to discriminate between pairs of matching and non-matching real-valued descriptors (d_1, d_2) according to a cosine similarity function. The choice of the cosine similarity at training time is what allows us to compare binary descriptors via Hamming distance at deployment time. In fact, the cosine similarity between d_1, and d_2, defined as in 3, can be expressed as a function of the L2-normalized inner product between the sign-quantized descriptors b_1 and b_2, as in the following. Given the definition of Hamming distance

$$H(b_1, b_2) \triangleq \|b_1\| + \|b_2\| - 2\langle b_1, b_2 \rangle, \quad (6)$$

where $\langle \cdot, \cdot \rangle$ refers to the inner product between vectors, and $\| \cdot \|$ indicates the bit count of the sequence, we can conclude that the cosine similarity between d_1 and d_2 (up to a scaling factor and an additive term) is similar to the Hamming distance between b_1 and b_2 [43]. The distance between binary learned descriptors can be measured as the relative Hamming distance, which can be efficiently computed via simple bitwise XOR and $POPCNT$ operations as in the MPEG standard.

The rest of the feature matching procedure follows the MPEG CDVS specifications recalled in Sec. III-.2. After each descriptor in the query image is matched with its nearest descriptor in the reference image that survived the ratio test using a two-way approach, a score s is returned for each pair of matching descriptors. The following geometric verification block finally attempts to separate true from false matches by performing a geometric consistency check using the same approach [41] used in MPEG CDVS. As a side note, experiments with the RANSAC-based geometry consistency [46] showed no appreciable performance gain despite increased computational complexity. Finally, as per the MPEG CDVS reference image matching procedure, a matching score S for each pair of images to be compared is returned.

VI. Experimental Evaluation

In this section, we first experimentally find the hyper-parameters that maximize the performance of LDVS descriptors over the Brown *et al.* dataset [47]. Then, we evaluate and benchmark LDVS descriptors at patch matching over the Brown *et al.* and HPatches datasets [19] Finally, the pairwise image matching pipeline described in Sec. V is experimentally evaluated in an image matching task comparing the performance of our LDVS descriptors against the standard MPEG CDVS pipeline based on compressed SIFT descriptors.

Fig. 7. Example of patches extracted from the Liberty (L), Notre-dame (N) and Yosemite (Y) datasets.

A. Preliminary Analysis

1) Experimental Setup: The patch matching performance of the proposed framework is first evaluated over three datasets composed of 64×64 patches extracted from 3D reconstructions of the *Liberty* statue (L), the *Notre-dame* cathedral (N) and the *Yosemite* mountains (Y) [47]. The patches are centered upon Difference of Gaussians (DoG) key points with canonical scale and orientation. A few examples are shown in Fig. 7.

Following the approach of [47], the network is trained three times, one for each L, N, Y dataset. For each training set, the network performance is evaluated on the other two datasets, for a total of six different training/testing *setups* per experiment (e.g., in the tables below the N/Y setup refers to the case in which we train on Notre-dame and test on Yosemite). For each setup, training takes place over 250k pairs of matching patches and 250k pairs of non-matching patches; testing takes place over 25k matching pairs and 25k non-matching pairs of patches. The proposed neural network architecture is implemented using the *Torch7* framework on an NVIDIA GeForce GTX 1080 GPU.

The training relies on gradient descent with adaptive optimization (AdaGrad) [48]) over batches of 100 matching samples and 100 non-matching samples. Throughout the training phase and for each experiment, an initial learning rate of 10^{-3}, a learning rate decay of 5×10^{-5} and a weight decay of 10^{-4} are used. Training ends when the error on the test has not decreased for 10 consecutive epochs or after 400 epochs. Following the approach of [47], as a first step ROC curves are computed thresholding the normalized Hamming distance in (5) between pairs of binary descriptors of B bits each. For each setup, the False Positive Rate (FPR) for a True Positive Rate (TPR) of 95% is measured and in the following is reported as the percentual *patch classification error*.

2) Effects of Quantization: As a preliminary experiment, we investigate the effect of sign-quantization over the learned descriptors for $M = 3$ convolutional modules and descriptors of $B = 128$ elements. Fig. 8 shows the cosine similarity (*top*) in Eq. 3 and the normalized Hamming distance distributions (*bottom*) in Eq. 5 of the matching and non-matching patches over the entire test set, computed over pairs of real-valued (d_1, d_2) and sign-quantized (b_1, b_2) descriptors, respectively. The top figure shows that the average cosine similarity between pairs of matching patches is close to 1, while the average value of the similarity between non-matching patches is around zero. This confirms that our neural network has been trained so to generate descriptors with zero cosine similarity, i.e. orthogonal, for pairs of non-matching image patches.

3) Optimal Network Depth: As a first step, we experiment at finding the depth of the architecture in Fig. 4 yielding

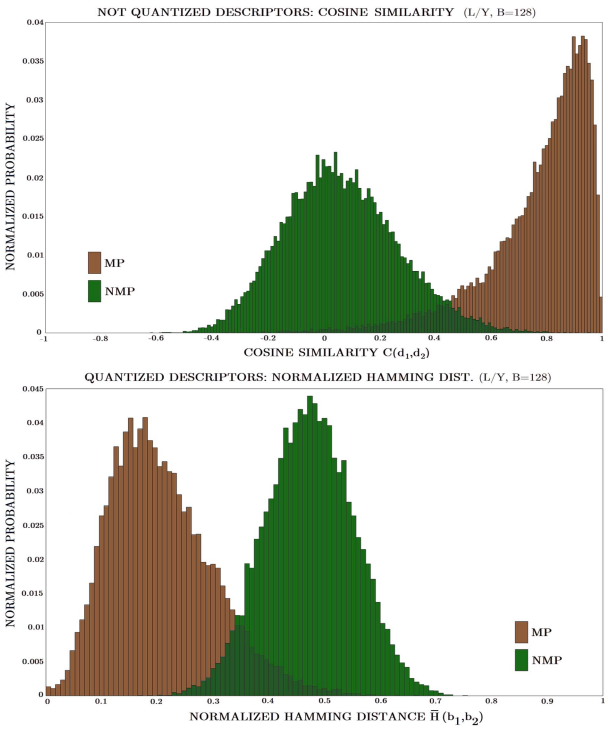

Fig. 8. Cosine similarity (*top*) and normalized Hamming distance (*bottom*) distributions of the matching and non-matching patches over the entire test set, computed over real-valued descriptors (top) and quantized binary descriptors (bottom).

TABLE I

PATCH CLASSIFICATION ERROR AS A FUNCTION OF THE NETWORK DEPTH M FOR DESCRIPTORS OF $B = 128$ BITS

M	f_M	FMR	N/L	Y/L	L/N	Y/N	L/Y	N/Y	AVG
2	128	16x16	9.53	13.24	5.88	7.47	9.60	9.17	9.15
3	256	8x8	6.94	9.73	4.39	5.05	9.53	9.09	**7.46**
4	512	4x4	7.72	11.17	5.03	5.64	10.83	8.66	8.18

the best patch matching performance on the Brown *et al.* dataset for descriptors of $B = 128$ bits. Namely, our goal is to experimentally find the number of convolutional layers M that maximizes the network performance at patch matching. To this end, we vary the number of convolutional modules in Fig. 4 such as $M \in \{2, 3, 4\}$. As M increases, the number of *featuremaps* produced by each layer doubles, while their resolution is halved. We follow the common design pattern where the number of filters doubles at each convolutional layer, starting from $f_1 = 64$ filters in the first layer. For example, in the case $M = 4$, we would obtain $(f_1, f_2, f_3, f_4 = 64, 128, 256, 512)$.

Table I shows the test set performance of the three different network configurations in which M varies within $\{2, 3, 4\}$. The f_M and $FMR = w_M \times h_M$ columns respectively report the featuremaps number and resolution (where $f_M \times FMR = FO_M$). The configuration where featuremaps have 8×8 resolution yields the best results. As a possible explanation, we hypothesize that a shallow architecture with larger featuremaps ($M = 2$) does not capture the high-level semantic information of the patches. On the contrary, a deeper architecture with low-resolution featuremaps ($M = 4$) is not able to preserve the spatial detail of the textures. From now on, in our experiments, we will consider the architecture with $M = 3$ convolutional blocks since it did yield the best results in this experiment.

4) Performance-Complexity Tradeoff: Next, we investigate the performance-complexity trade-off of the LDVS architecture as a function of the numbers of filters f_1, f_2, f_3 in each convolutional layer ($M = 3$). We consider three different configurations to which we refer to as $0.5X, 1X, 1.5X$, in which the number of filters doubles with m (i.e. module index of each convolutional layer, going from $M = 1$ to $M = 3$). Let us indicate as *1X* the network architecture considered up to this point where $(f_1, f_2, f_3 = 64, 128, 256)$. Now, *0.5X* indicates the case where $(f_1, f_2, f_3 = 32, 64, 128)$ (half the number of filters), and *1.5X* to $(f_1, f_2, f_3 = 96, 192, 384)$ (50% more filters), respectively. For each subnetwork, the $M+1$-th convolutional layer is replaced by a Fully Connected layer with B hidden units which takes at input $FO_M = f_3 \times FMR$ features and has B outputs, one for each of the two subnetworks the Siamese structure is composed of (*(FC)* scheme). Notice that only the number of learnable parameters is affected, while the total number of neurons in the network does not vary. Also, we recall that the proposed architecture includes 5×5 filters in the first convolutional layer and 3×3 filters in the following layers. Moreover, we compare the fully convolutional architecture proposed in Fig. 4 with an architecture similar to [43] and comparable to [12], [17], [34], where the bottleneck layer has a fully connected topology.

Table II reports the memory and computational complexity as the number of learnable parameters and multiply-add (MADD) operations. The *LDSV-FC* architecture memory complexity is dominated by the number of parameters in the fully connected layer. The number of parameters of LDVS is between three and four times less than its LDVS-FC counterpart, showing the benefits of a convolutional bottleneck layer. Concerning the number of MADD operations, our architecture has lower complexity than [12] and [34] and about the same order of complexity of [15] in its no-bottleneck version.

Table III then shows that the proposed fully convolutional (*Conv.*) approach has also better performance than the (*FC*) variant for almost all configurations despite the lower complexity. In detail, the *Conv. 1.5X* configuration offers the smallest average error over the six setups. Hence, in the following, we will identify our LDVS descriptor with the *Conv. 1.5x* configuration.

We also experimented at increasing the number of the filters in each layer beyond *1.5X*, up to *2X* and *3X*. The experiments showed that while the error over the training set further decreased, the error over the test set that represents the network ability to generalize over previously unseen data increased. We attribute such an effect to the tendency of networks with large learning capacities to overfit to the training data, especially in the *FC* case where the bottleneck layer has a fully connected topology. Notice that techniques such as $L2$ regularization and probabilistic dropout showed to be ineffective towards this end. In the following, we will refer to the $M = 3$, *Conv. 1.5X* architecture as *proposed* due to its favorable complexity-performance trade-off.

TABLE II

NUMBER OF LEARNABLE PARAMETERS AND MULTIPLY-ADD (MADD) OPERATIONS AS A FUNCTION OF THE NUMBERS OF FILTERS f_1, f_2, f_3 FOR THE PROPOSED FULLY CONVOLUTIONAL ARCHITECTURE (*conv.*) AND THE REFERENCE WHERE THE LAST LAYER IS FULLY CONNECTED (*fc*) ($M = 3$, $B = 128$) AND FOR SOME COMPARABLE CONVOLUTIONAL ARCHITECTURES

Proposed Approach								**Other Approaches**		
	Number of Filters			**Number of Parameters**		**Number of MADD**			**Number of**	
	f_1	f_2	f_3	**Convolutional**	**FC**	**Convolutional**	**FC**		**Parameters**	**MADD**
0.5x	32	64	128	240 k	1,1 M	8,4 M	9,2 M	**MatchNet [15]**	2.6 M	19.4 M
1x	64	128	256	665 k	2,4 M	30,4 M	32,3 M	**Zagoruyko *et al.* [12]**	13 M	101 M
1.5x	96	192	384	1,2 M	3,9 M	66 M	68,5 M	**DeepCD [34]**	25 M	135 M

TABLE III

PATCH CLASSIFICATION ERROR OVER BROWN *et al.* AS A FUNCTION OF THE NUMBER OF FILTERS (f_1, f_2, f_3) FOR THE FULLY CONVOLUTIONAL LDVS (*conv.*) AND WHERE THE LAST LAYER IS FULLY CONNECTED (*fc*) ($m = 3$, $b = 128$)

	f_1, f_2, f_3	**N/L**	**Y/L**	**L/N**	**Y/N**	**L/Y**	**N/Y**	**AVG**
Conv.	**0.5X**	9.60	13.50	5.89	8.26	9.48	9.34	9.35
	1X	6.94	9.73	4.39	5.05	9.53	9.09	7.46
	1.5X	6.73	9.71	4.08	4.97	8.80	7.63	6.99
FC	**0.5X**	10.28	14.23	5.68	8.07	9.75	10.20	9.70
	1X	6.90	9.56	3.89	5.04	9.93	9.16	7.41
	1.5X	7.17	9.81	4.41	4.99	9.42	7.70	7.25

TABLE IV

PATCH CLASSIFICATION ERROR OVER THE BROWN *et al.* DATASET FOR LDVS AND OTHER BINARY DESCRIPTORS

	B (bits)	**N/L**	**Y/L**	**L/N**	**Y/N**	**L/Y**	**N/Y**	**AVG**
BinBoost [25]	64	16.90	22.88	20.49	18.97	21.67	14.54	19.24
DELFT [49]	64	19.11	20.43	13.59	12.34	19.25	17.36	17.01
	128	16.27	18.32	9.31	10.56	15.77	13.94	14.03
Cov.Opt [50]	1024	8.25	14.84	12.16	8.50	14.8	7.09	11.00
DeepCD [34]	192	8.31	16.29	14.45	13.62	18.38	8.97	13.34
	768	3.73	9.97	7.82	7.67	11.75	4.35	7.55
L2-Net[33]	128	10.3	11.71	6.37	6.76	13.5	11.57	10.03
L2-Net+	128	7.44	10.29	3.81	4.31	8.81	7.45	7.01
CS L2-Net	256	2.55	4.24	0.87	1.39	3.81	2.84	2.61
CS L2-Net+	256	1.71	3.87	0.56	1.09	2.07	1.3	1.76
LDVS	64	9.66	12.38	6.18	6.76	11.88	10.41	9.55
	128	6.73	9.71	4.08	4.97	8.80	7.63	6.99
	192	6.69	9.43	3.99	4.36	8.32	7.10	6.65
	256	6.08	8.66	3.69	4.09	7.51	6.57	6.10

B. Pair-Wise Patch Matching

1) Experiments on Brown et al. Dataset: Table IV compares our LDVS with several state-of-the-art references. LDVS outperform or perform favorably with respect to the competitors on an equal bitrate basis. Only for the N/L, Y/L, and N/Y setups *DeepCD* [34] performs better, but at the expense of a 3 times larger descriptor (768 bits with respect to B = 256 bits for the proposed scheme). We attribute such gains to the optimal tradeoff between semantic level and resolution of the feature maps extracted from the patches together with a better network generalization ability in reason of the fully convolutional design that is less prone to yield to overfitting issues. Finally, CS L2Net outperforms CDVS since it relies on a different and more complex central-surround architecture.

TABLE V

AVERAGE MAP (%) FOR LDVS OVER THE HPATCHES [19] DATASET FOR VERIFICATION, MATCHING AND RETRIEVAL TASKS

Method	**Size**	**Verification**	**Matching**	**Retrieval**
BRIEF [21]	256	68.82	9.21	24.96
ORB [24]	256	69.10	13.86	28.27
BinBoost [25]	256	74.73	13.19	32.64
DeepBit [29]	256	61.27	13.05	20.61
GraphBit [36]	256	65.19	14.22	25.19
LDVS	128	**80.24**	**14.23**	**37.05**

2) Experiments on HPatches Dataset: Furthermore, we experiment over the HPatches [19], a large dataset of patches (2.5M pairs) with predefined evaluation protocol for the tasks of patch matching, retrieval, and classification. The protocol is composed of 116 image sequences each of which is provided with six other images for homographies under different levels of geometric noise and illumination variations. We trained our LDVS descriptor for different descriptor rates over a training set of 250*k* matching patches and 250*k* non-matching patches and tested it on a previously unseen set of 25*k* matching patches and 25*k* non-matching patches. Then, we computed the performance at the three considered tasks using the [19] evaluation framework reporting the results in Tab. V. For this experiment, increasing the descriptor length to 192 and 256 did not lead to significant performance improvements. For this reason, Tab. V reports performance on the three HPatches tasks obtained with LDVS 128 bits descriptors, showing that the proposed binary descriptor is competitive with respect to others, especially at Verification and Retrieval, even at lower descriptors rates.

C. Pair-Wise Image Matching Experiments

Next, we evaluate our LDVS descriptors and the relative pair-wise image matching pipeline described in Sec. V against the standard MPEG CDVS descriptors and relative image matching pipeline described in Sec. III-.2.

1) Experimental Setup: All our following experiments are performed over the reference MPEG CDVS dataset, which consists of about 17k matching pairs and 17k non-matching pairs of images, hence a considerably large and complex collection of images. Such dataset is a collection of different annotated datasets including the Stanford Mobile Visual Search Dataset [51], the Stanford Streaming Augmented Reality Dataset [52], and the Zurich Building Image Database [53]. Images are annotated for pairwise image matching, i.e. a list of

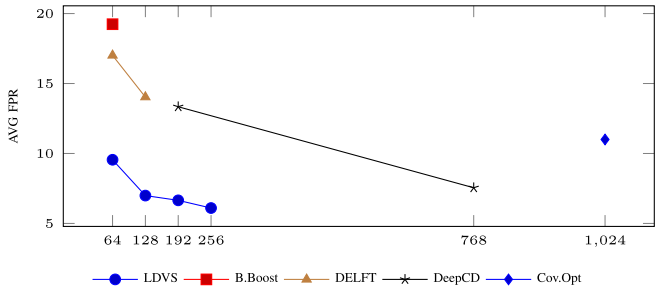

Fig. 9. Patch classification error (FPR) over Brown *et al.* as a function of the descriptor length averaged across all setups (rightmost column of Tab. IV).

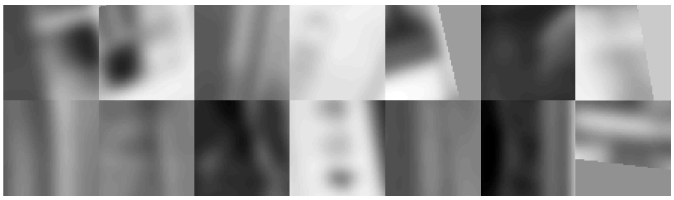

Fig. 10. Examples of patches sampled at random from the CDVS dataset.

all images depicting the same object (e.g., a specific building) is provided.

The data required to train the network in Fig. 4 (with $M = 3$ convolutional blocks) from scratch are prepared as follows. First, we extract a set of about 1.6 million pairs of 64×64 *matching* patches using the standard CDVS extraction and matching pipeline. Second, we generate an equally sized set of *non-matching* pairs of patches by randomly drawing 64×64 patches from non-matching pairs of images. While such a choice is not expected to be optimal towards maximizing the performance of the proposed LDVS descriptors, it is meant to guarantee a fair comparison with CDVS descriptors. The above sets of pairs of patches are randomly subdivided into a training set of 250k pairs of matching patches and 250k pairs of non-matching patches, whereas the remaining patches are used for testing purposes. Overall, the training set is composed of about 1 M pairs of image patches, where matching pairs of patches are labeled as $t = 1$ and non-matching pairs as $t = 0$ as explained in Sec. IV-B as exemplified in Fig. 10. To ensure the network is tested on patches not seen at training time, we exclude from the test set patches extracted from images that contributed at least one patch to the train set. Notice that all the results reported in the following refer to patches extracted from the test images.

2) Comparison With MPEG CDVS: We recall that the CDVS standard considers different bitrates or *modes* for representing an image as a collection of compressed SIFT descriptors. Each CDVS *mode* yields different average local descriptors length, and a fixed (maximum) amount of descriptors that can be retained for each image, as done in the *Feature Matching* stage in Fig. 5. To ensure a fair comparison, we first extracted all the CDVS descriptors for all modes, noting for each mode the average compressed local descriptor size and the average number of local descriptors per CDVS descriptor. Then, concerning the proposed LDVS descriptors,

TABLE VI

COMPARISON OF THE AVERAGE DESCRIPTOR LENGTH FOR OUR LDVS DESCRIPTORS AND STANDARD MPEG CDVS DESCRIPTORS

	Mode 2	**Mode 3**	**Mode 4**	**Mode 5**
Max. Selected Keypoints	250	250	300	500
Stand. CDVS	33 bit	67 bit	107 bit	133 bit
Prop. LDVS	32 bit	64 bit	112 bit	128 bit

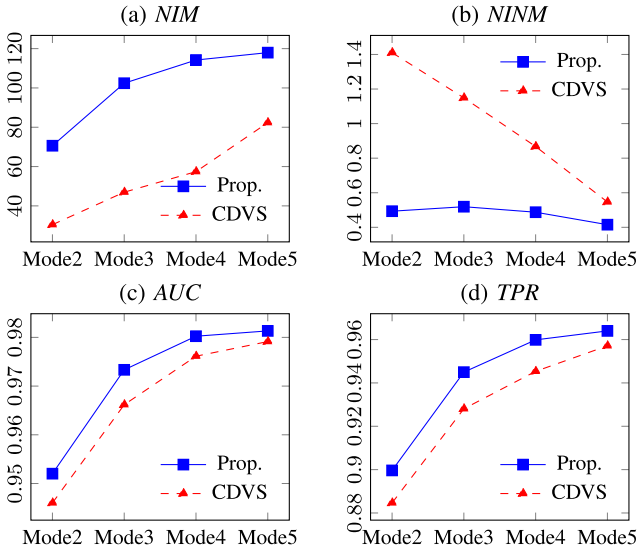

Fig. 11. Pair-wise image matching performance comparison between the proposed LDVS and standard CDVS descriptors as a function of the descriptor mode. Top-left: number of inliers per matching image (NIM). Top-right: number of matches wrongly labeled as inliers per non-matching image (NINM). Bottom-left: area under the curve (AUC). Bottom-right: true positive rate (TPR) when FPR= 1%.

for each mode, we only retained the CDVS average number of descriptors and limited the descriptor size B to the CDVS average equivalent number. Table VI summarizes the resulting average number of local descriptors and the average descriptor length in bits for our LDVS descriptors and standard CDVS descriptors.

The performance of our LDVS descriptors and standard CDVS descriptors are compared according to the following metrics:

- Number of Inliers per Matching Image (*NIM*): average number of true matching inliers for each pair of matching images.
- Number of matches wrongly labeled as inliers per Non-matching Image (*NINM*): average number of false matching inliers for each pair of non-matching images.
- Area Under the Curve (*AUC*): area of the graph under the S scores ROC curves
- True Positive Rate (*TPR*): true positives rate for a fixed 1% False Positive Rate as in the official CDVS evaluation framework.

AUC and *TPR* are computed by thresholding the S scores (see 2) for each pair of annotated query/reference image. For each metric, we report the average value computed over the annotated CDVS image dataset, i.e. about 17k matching and

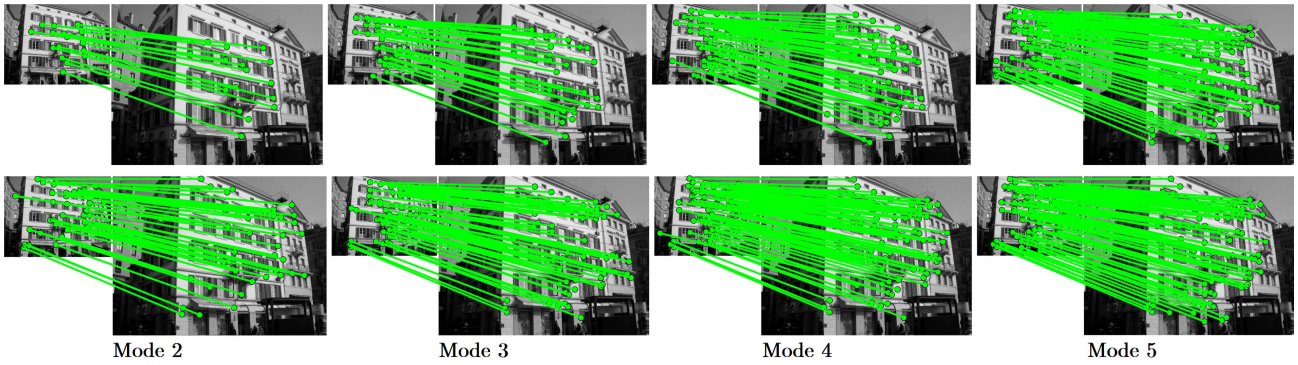

Fig. 12. Example of inliers as a function of the descriptor mode (or bitrate) for MPEG CDVS (top) and LDVS (below) descriptors. The maximum selected number of keypoints for *Mode 2*, *Mode 3*, *Mode 4*, and *Mode 5*, as in Tab. VI, is 250, 250, 300, and 500, respectively. The proposed LDVS descriptor yields more inliers especially at lower bitrates, enabling robust geometric verification and better performance at image matching.

17k non-matching images. While the *ratio test threshold* value for CDVS (Λ_{CDVS}) is set at 0.85, we verified that optimal performance with LDVS is obtained by setting $\Lambda_{LDVS} = 0.90$.

Fig. 11 shows that for each of the four considered metrics, the proposed LDVS descriptors outperform the CDVS reference for every mode (i.e., at any bitrate). Concerning the number of true inliers *NIM*, i.e. the number of correct descriptors matches surviving geometry verification, the proposed approach almost doubles such number with respect to CDVS. We recall that the proposed approach and CDVS rely on the same number of descriptors per image since they share the same *keypoint detection* and *feature selection* steps in Fig. 5). More inliers are key to improve the robustness of the overall image matching process, especially at low bitrates where the available descriptors are few. At the same time, the number of false inliers *NINM* is significantly lower for our approach, especially at low bitrates (CDVS yields about three times more false inliers at *Mode2*). That is, in the case of non-matching images, the average number of matched keypoints that survives the *geometric verification* phase and end up wrongly being considered inliers is considerably lower than CDVS. As a result of more inliers for true matching pairs and fewer inliers for nonmatching pairs, our approach scores better than CDVS in terms of *AUC* and *TPR* metrics at all rates. Concluding, this experiment shows that the proposed LDVS descriptors allow more robust image matching thanks to the ability to better discriminate matching images from non-matching ones.

Fig. 12 depicts inliers for CDVS (top) and our LDVS (bottom) descriptor for a sample image. The picture shows how our LDVS descriptor yields more inliers than CDVS at any bitrate. This is particularly evident in *Mode 2* and *Mode 3*, i.e. in the case where the number of descriptors available for image matching is lower. In conclusion, our LDVS descriptors outperform CDVS descriptors even when fitted within a CDVS-optimized pair-wise image matching pipeline, especially in the case of limited computational, storage and bandwidth resources (i.e. at lower bitrates).

VII. Conclusion

The proposed LDVS descriptors are learnable, binary local descriptors designed for image matching within the MPEG CDVS international standard. LDVS descriptors are learned so that they can be sign-quantized and compared using the Hamming distance as required by the standard. The fully convolutional architecture exhibits a low parameters count with respect to comparable architectures. LDVS performs favorably at patch matching over competing learned binary descriptor technologies on two challenging datasets Brown *et al.* and HPatches. Then, a complete pair-wise image matching pipeline is designed around the introduced LDVS descriptors. We integrate them into the reference CDVS evaluation framework that employs a geometric consistency test. Experiments show that LDVS descriptors outperform same-rate compressed CDVS descriptors at pair-wise image matching, especially at low descriptor lengths. We attribute such gains to the LDVS descriptors ability to generate a greater number of matches capable to survive geometric consistency checks.

References

[1] G. Francini, M. Balestri, and S. Lepsoy, *CDVS: Telecom Italia Response to CE1-Interest Point Detection*, Standard ISO/IEC JTC1/SC29/WG11 31369, 2013.

[2] D. G. Lowe, "Distinctive image features from scale-invariant keypoints," *Int. J. Comput. Vis.*, vol. 60, no. 2, pp. 91–110, Nov. 2004.

[3] S. Zhang, Q. Tian, K. Lu, Q. Huang, and W. Gao, "Edge-SIFT: Discriminative binary descriptor for scalable partial-duplicate mobile search," *IEEE Trans. Image Process.*, vol. 22, no. 7, pp. 2889–2902, Jul. 2013.

[4] H. Zhu, M. Long, J. Wang, and Y. Cao, "Deep hashing network for efficient similarity retrieval," in *Proc. 13th AAAI Conf. Artif. Intell.*, 2016, pp. 2415–2421.

[5] J. Qian, X. Lin, H. Liu, Y. Deng, and R. Ji, "Towards compact visual descriptor via deep Fisher network with binary embedding," in *Proc. IEEE Int. Conf. Multimedia Expo (ICME)*, Jul. 2018, pp. 1–6.

[6] Y. Wu *et al.*, "Codebook-free compact descriptor for scalable visual search," *IEEE Trans. Multimedia*, vol. 21, no. 2, pp. 388–401, Feb. 2019.

[7] *Information Technology–Multimedia Content Description Interface—Part 15: Compact Descriptors for Video Analysis*, document ISO/IEC 15938-15:2019, 2019.

[8] Y. Lou *et al.*, "Compact deep invariant descriptors for video retrieval," in *Proc. Data Compress. Conf. (DCC)*, Apr. 2017, pp. 420–429.

[9] L.-Y. Duan *et al.*, "Fast MPEG-CDVS encoder with GPU-CPU hybrid computing," *IEEE Trans. Image Process.*, vol. 27, no. 5, pp. 2201–2216, May 2018.

[10] L. Zheng, Y. Yang, and Q. Tian, "SIFT meets CNN: A decade survey of instance retrieval," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 40, no. 5, pp. 1224–1244, May 2018.

[11] E. Simo-Serra, E. Trulls, L. Ferraz, I. Kokkinos, P. Fua, and F. Moreno-Noguer, "Discriminative learning of deep convolutional feature point descriptors," in *Proc. IEEE Int. Conf. Comput. Vis. (ICCV)*, Dec. 2015, pp. 118–126.

[12] S. Zagoruyko and N. Komodakis, "Learning to compare image patches via convolutional neural networks," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2015, pp. 4353–4361.

[13] W. Luo, A. G. Schwing, and R. Urtasun, "Efficient deep learning for stereo matching," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2016, pp. 5695–5703.

[14] K. M. Yi, E. Trulls, V. Lepetit, and P. Fua, "LIFT: Learned invariant feature transform," in *Proc. Eur. Conf. Comput. Vis.* Cham, Switzerland: Springer, 2016, pp. 467–483.

[15] X. Han, T. Leung, Y. Jia, R. Sukthankar, and A. C. Berg, "MatchNet: Unifying feature and metric learning for patch-based matching," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2015, pp. 3279–3286.

[16] A. Mishchuk, D. Mishkin, F. Radenovic, and J. Matas, "Working hard to know your neighbor's margins: Local descriptor learning loss," in *Proc. Adv. Neural Inf. Process. Syst.*, 2017, pp. 4829–4840.

[17] J. Žbontar and Y. LeCun, "Stereo matching by training a convolutional neural network to compare image patches," *J. Mach. Learn. Res.*, vol. 17, nos. 1–32, p. 2, 2016.

[18] S. Zhang, Q. Tian, Q. Huang, W. Gao, and Y. Rui, "USB: Ultrashort binary descriptor for fast visual matching and retrieval," *IEEE Trans. Image Process.*, vol. 23, no. 8, pp. 3671–3683, Aug. 2014.

[19] V. Balntas, K. Lenc, A. Vedaldi, and K. Mikolajczyk, "HPatches: A benchmark and evaluation of handcrafted and learned local descriptors," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jul. 2017, pp. 5173–5182.

[20] J. L. Schonberger, H. Hardmeier, T. Sattler, and M. Pollefeys, "Comparative evaluation of hand-crafted and learned local features," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jul. 2017, pp. 6959–6968.

[21] M. Calonder, V. Lepetit, M. Ozuysal, T. Trzcinski, C. Strecha, and P. Fua, "BRIEF: Computing a local binary descriptor very fast," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 34, no. 7, pp. 1281–1298, Jul. 2012.

[22] S. Leutenegger, M. Chli, and R. Y. Siegwart, "BRISK: Binary robust invariant scalable keypoints," in *Proc. Int. Conf. Comput. Vis.*, Nov. 2011, pp. 2548–2555.

[23] A. Alahi, R. Ortiz, and P. Vandergheynst, "FREAK: Fast retina keypoint," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jun. 2012, pp. 510–517.

[24] E. Rublee, V. Rabaud, K. Konolige, and G. Bradski, "ORB: An efficient alternative to SIFT or SURF," in *Proc. Int. Conf. Comput. Vis.*, Nov. 2011, pp. 2564–2571.

[25] T. Trzcinski, M. Christoudias, P. Fua, and V. Lepetit, "Boosting binary keypoint descriptors," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jun. 2013, pp. 2874–2881.

[26] F. Shen, C. Shen, W. Liu, and H. T. Shen, "Supervised discrete hashing," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2015, pp. 37–45.

[27] V. E. Liong, J. Lu, G. Wang, P. Moulin, and J. Zhou, "Deep hashing for compact binary codes learning," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2015, pp. 2475–2483.

[28] Y. Duan, J. Lu, J. Feng, and J. Zhou, "Learning rotation-invariant local binary descriptor," *IEEE Trans. Image Process.*, vol. 26, no. 8, pp. 3636–3651, Aug. 2017.

[29] K. Lin, J. Lu, C.-S. Chen, and J. Zhou, "Learning compact binary descriptors with unsupervised deep neural networks," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2016, pp. 1183–1192.

[30] H. Jain, J. Zepeda, P. Perez, and R. Gribonval, "SuBiC: A supervised, structured binary code for image search," in *Proc. IEEE Int. Conf. Comput. Vis. (ICCV)*, Oct. 2017, pp. 833–842.

[31] H. Liu, R. Wang, S. Shan, and X. Chen, "Learning multifunctional binary codes for both category and attribute oriented retrieval tasks," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jul. 2017, pp. 3901–3910.

[32] Y. Shen, L. Liu, L. Shao, and J. Song, "Deep binaries: Encoding semantic-rich cues for efficient textual-visual cross retrieval," in *Proc. IEEE Int. Conf. Comput. Vis. (ICCV)*, Oct. 2017, pp. 4097–4106.

[33] Y. Tian, B. Fan, and F. Wu, "L2-net: Deep learning of discriminative patch descriptor in Euclidean space," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jul. 2017, pp. 661–669.

[34] T.-Y. Yang, J.-H. Hsu, Y.-Y. Lin, and Y.-Y. Chuang, "DeepCD: Learning deep complementary descriptors for patch representations," in *Proc. IEEE Int. Conf. Comput. Vis. (ICCV)*, Oct. 2017, pp. 3314–3322.

[35] K. Lin, J. Lu, C.-S. Chen, J. Zhou, and M.-T. Sun, "Unsupervised deep learning of compact binary descriptors," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 41, no. 6, pp. 1501–1514, Jun. 2019.

[36] Y. Duan, Z. Wang, J. Lu, X. Lin, and J. Zhou, "GraphBit: Bitwise interaction mining via deep reinforcement learning," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, Jun. 2018, pp. 8270–8279.

[37] M. Zieba, P. Semberecki, T. El-Gaaly, and T. Trzcinski, "BinGAN: Learning compact binary descriptors with a regularized GAN," in *Proc. Adv. Neural Inf. Process. Syst.*, 2018, pp. 3608–3618.

[38] L.-Y. Duan, J. Lin, J. Chen, T. Huang, and W. Gao, "Compact descriptors for visual search," *IEEE MultimediaMag.*, vol. 21, no. 3, pp. 30–40, Jul. 2014.

[39] L.-Y. Duan *et al.*, "Overview of the MPEG-CDVS standard," *IEEE Trans. Image Process.*, vol. 25, no. 1, pp. 179–194, Jan. 2016.

[40] G. Francini, S. Lepsøy, and M. Balestri, "Selection of local features for visual search," *Signal Process., Image Commun.*, vol. 28, no. 4, pp. 311–322, Apr. 2013.

[41] S. Lepsoy, G. Francini, G. Cordara, and P. P. B. de Gusmao, "Statistical modelling of outliers for fast visual search," in *Proc. IEEE Int. Conf. Multimedia Expo*, Jul. 2011, pp. 1–6.

[42] J. Bromley, I. Guyon, Y. LeCun, E. Säckinger, and R. Shah, "Signature verification using a 'Siamese' time delay neural network," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 1994, pp. 737–744.

[43] A. Migliorati, A. Fiandrotti, G. Francini, S. Lepsoy, and R. Leonardi, "Feature fusion for robust patch matching with compact binary descriptors," in *Proc. IEEE 20th Int. Workshop Multimedia Signal Process. (MMSP)*, Aug. 2018, pp. 1–6.

[44] X. Zhang, F. X. Yu, S. Kumar, and S.-F. Chang, "Learning spread-out local feature descriptors," 2017, *arXiv:1708.06320*. [Online]. Available: http://arxiv.org/abs/1708.06320

[45] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," in *Proc. Int. Conf. Mach. Learn.*, 2015, pp. 448–456.

[46] M. A. Fischler and R. Bolles, "Random sample consensus: A paradigm for model fitting with applications to image analysis and automated cartography," *Commun. ACM*, vol. 24, no. 6, pp. 381–395, 1981.

[47] M. Brown, G. Hua, and S. Winder, "Discriminative learning of local image descriptors," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 33, no. 1, pp. 43–57, Jan. 2011.

[48] J. Duchi, E. Hazan, and Y. Singer, "Adaptive subgradient methods for online learning and stochastic optimization," *J. Mach. Learn. Res.*, vol. 12, pp. 2121–2159, Feb. 2011.

[49] Z. Liu, Z. Li, J. Zhang, and L. Liu, "Euclidean and Hamming embedding for image patch description with convolutional networks," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. Workshops (CVPRW)*, Jun. 2016, pp. 72–78.

[50] K. Simonyan, A. Vedaldi, and A. Zisserman, "Learning local feature descriptors using convex optimisation," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 36, no. 8, pp. 1573–1585, Aug. 2014.

[51] V. R. Chandrasekhar *et al.*, "The stanford mobile visual search data set," in *Proc. 2nd Annu. ACM Conf. Multimedia Syst.*, Feb. 2011, pp. 117–122.

[52] D. M. Chen, S. S. Tsai, R. Vedantham, R. Grzeszczuk, and B. Girod, "Streaming mobile augmented reality on mobile phones," in *Proc. 8th IEEE Int. Symp. Mixed Augmented Reality*, Oct. 2009, pp. 181–182.

[53] H. Shao, T. Svoboda, and L. V. Gool, "Zubud-Zurich buildings database for image based recognition," Comput. Vis. Lab, Swiss Federal Inst. Technol., Zürich, Switzerland, Tech. Rep. 260, 2003, p. 20.

Andrea Migliorati (Member, IEEE) received the master's degree in communication technologies and multimedia from the Università degli Studi di Brescia in 2015 and the Ph.D. degree in telecommunication engineering after a three year collaboration between the Università degli Studi di Brescia and Telecom Italia's JOL-Joint Open Cognitive Lab, Turin, Italy, in 2019. He is currently a ResearchFellow of the Department of Electronics and Telecommunications, Politecnico di Torino, Turin, Italy. His current research interests deep learning techniques for image analysis, adversarial robustness, and biometric verification, as well as compression and encryption algorithms for EO on-board imaging.

Attilio Fiandrotti (Senior Member, IEEE) received the Ph.D. degree in computer science from the Politecnico di Torino in 2010. He is currently an Assistant Professor with the Dipartimento di Informatica, Università di Torino, Italy, and holds a position as a maıtre de conférénces with the Multimedia Group at LTCI, Télécom Paris, Institut Polytechnique de Paris. His current research interests include deep learning techniques for image and video analysis, compression, and synthesis, and the distribution of multimedia contents over packet networks.

Gianluca Francini received the master's degree *(cum laude)* in computer science from the University of Torino, Italy. In 1996, he joined the Multimedia Research Group of CSELT (the former research center of Telecom Italia), working on three-dimensional video conference systems. He is a senior researcher actively involved in computer vision, image and video retrieval, 3D reconstruction, recommendation systems, and self-organizing networks. He is currently leading the Joint Open Cognitive Lab, a laboratory opened by Telecom Italia at the Polytechnic of Turin in 2014, with the objective of developing technologies for visual search and deep learning. He is a co-inventor of 20 patents in the fields of image and video analysis.

Riccardo Leonardi (Fellow, IEEE) received the Diploma and Ph.D. degrees in electrical engineering from the Swiss Federal Institute of Technology, Lausanne, Switzerland, in 1984 and 1987, respectively. He has been a Researcher with UC Santa Barbara and Bell Laboratories from 1987 to 1991. Since 1992, he has been with the Università degli Studi di Brescia, Italy, where he established the Signal, Imaging, Networking, and Communications (SINC) Group. He conducts research in signal and image representation for visual communications and visual content protection. He is also an expert in machine learning tools with applications to multimedia content analysis and medical imaging. He is as an Expert Evaluator of the European Commission and is currently the President of the Italian Telecommunication and Information Technology Association (GTTI).