Deep learned compact binary descriptor with a lightweight network-in-network architecture for visual description

Ravimal Bandara[1] · Lochandaka Ranathunga[1] · Nor Aniza Abdullah[2]

Abstract

Binary descriptors have been widely used for real-time image retrieval and correspondence matching. However, most of the learned descriptors are obtained using a large deep neural network (DNN) with several million parameters, and the learned binary codes are generally not invariant to many geometrical variances which is crucial for accurate correspondence matching. To address this problem, we proposed a new learning approach using a lightweight DNN architecture via a stack of multiple multilayer perceptrons based on the network in network (NIN) architecture, and a restricted Boltzmann machine (RBM). The latter is used for mapping the features to binary codes, and carry out the geometrically invariant correspondence matching task. Our experimental results on several benchmark datasets (e.g., Brown, Oxford, Paris, INRIA Holidays, RomePatches, HPatches, and CIFAR-10) show that the proposed approach produces the learned binary descriptor that outperforms other baseline self-supervised binary descriptors in terms of correspondence matching despite the smaller size of its DNN. Most importantly, the proposed approach does not freeze the features that are obtained while pre-training the NIN model. Instead, it fine-tunes the features while learning the features needed for binary mapping through the RBM. Additionally, its lightweight architecture makes it suitable for resource-constrained devices.

Keywords Binary descriptor · Network-in-network · Restricted Boltzmann machine · Correspondence matching · Lightweight deep neural network

1 Introduction

Because visual data, such as images and videos, have a high density, their dimensionality must be reduced via feature extraction so that the abstract representation of the data can be formed, thus allowing computers to understand them [1–4]. Most computer vision tasks—including scene classification, object recognition, image stitching, and 3D reconstruction—use feature extraction as a preliminary step [5, 6]. The primary purpose of any feature descriptor is to extract the most essential and discriminative information from an image while being robust to various geometrical transformation such as rotation and scaling [4, 7]. Furthermore, recent studies have focused on highly efficient descriptors designed to enable visual data to be retrieved quickly from extensive archives of images and videos [4].

Throughout the past decade, many feature descriptors have been handcrafted or learned through various machine learning approaches in attempts to improve their discriminative power, robustness, and efficiency. Among these descriptors, the SIFT [8] and SURF [9] descriptors are the most widely explored. Convolutional neural networks (CNNs) were another milestone in the history of feature descriptors. CNNs learn optimal features effectively to classify the input among a given set of classes [10]. CNNs can accurately map a raw image to the most closely related labels available in the network [11]. However, the learning of intermediate representation of images that is also known as image descriptor, without going into the classification step, is still vital for some computer vision tasks, such as one-shot learning and correspondence matching [12, 13]. Hence, this study

✉ Ravimal Bandara
ravimalb@uom.lk; ravimal9@gmail.com

Lochandaka Ranathunga
lochandaka@uom.lk

Nor Aniza Abdullah
noraniza@um.edu.my

[1] University of Moratuwa, Moratuwa, Sri Lanka

[2] Faculty of Computer Science and Information Technology, University of Malaya, Kuala Lumpur, Wilayah Persekutuan, Kuala Lumpur, Malaysia

focuses on deriving an image descriptor for correspondence matching task with mainly concerning the efficiency and robustness.

Both SIFT and SURF suffer from high computational cost and high dimensionality, which negatively affect the speed of matching [14–16]. Moreover, image matching is a computationally expensive task due to the real-valued descriptors, which require high-precision floating-point operations [5, 15, 16]. The problems of dimensionality and computational expensiveness have been addressed by the proposition of several binary descriptors, such as BRIEF [16], ORB [15], BRISK [5], and FREAK [17]. Binary descriptors perform storing and matching tasks very efficiently due to their compactness and Hamming distance-based similarity measurement. Calculation of Hamming distance requires only a few bitwise operations; hence, binary descriptors are preferred for real-time image matching tasks [15]. However, most of these descriptors are sensitive to scale, orientation, illumination, and image noise due to the presence of handcrafted features that arise from the use of simple intensity comparison techniques. [18–21] propose several improvements to the binary description. The key component that is common to all these studies is the use of learning algorithms to learn encoding schemes to transform similar information to similar binary codes. The approaches used in [22–25] are based on feature learning through deep neural networks (DNNs), which typically involve the optimization of many parameters during the learning phase. DNNs also perform many arithmetic operations during the inferencing phase. Hence, computations involved in DNNs are expensive. Further, these descriptors are less rotationally invariant than other descriptors because they use linear convolutional filters and learn the encoding scheme based on pre-learned and frozen features that cannot be adopted for drastic variations in rotation [13, 26]. Although these descriptors are relatively easy to train, they are usually limited by features that are not optimal for binary encoding. The approaches in [18, 19, 27–29] focus on learning the optimal features for binary encoding. However, their use is limited due to supervised learning, which cannot be applied in the absence of annotated data.

In this paper, we propose a lightweight DNN architecture for learning compact binary descriptor to carry out image correspondence matching tasks. In the proposed DNN, a Gaussian-Bernoulli restricted Boltzmann machine (RBM) [30] is used to learn binary codes from the features extracted from a network-in-network (NIN) [31] DNN model. The stack of multiple multilayer perceptrons (MLPs), that is used in NIN, makes it extremely compact and efficient in feature extraction. RBM is a generative network that can learn a mapping between two distributions without supervision. RBMs are extremely small due to their single-layer architecture. A well-trained RBM can reconstruct the input (from the visible layer) that corresponds to a given output (to the hidden layer). In addition to the inherent compactness of the DNN architecture, a notable advantage of the proposed method is that it learns discriminative compact binary codes by optimizing both the "feature-to-binary mapping function" and "feature extraction" in a new learning approach. The objective of the new learning approach is to reduce the variation among the features extracted for different orientations of the same image patch. A conventional learning approach cannot be used to train this NIN–RBM hybrid network. This is mainly because of the incompatible learning approaches used in the different parts of the network. Therefore, in the proposed learning approach, first, the two parts learned individually then the deep layers of the NIN update their parameters by backpropagating the error between its output features and the reconstructed inputs of the RBM from an estimated target. This is in contrast to conventional learning approaches, which update parameters by backpropagating the error between the outputs and the provided targets. The efficiency and effectiveness of the proposed DNN model are demonstrated by employing it in correspondence matching tasks using several benchmark datasets. The next section describes related works that have been used to derive the proposed DNN architecture and for comparison purposes.

2 Related works

The primary purpose of a syntactic visual feature is to abstract visual data while keeping enough discriminative information for accurate matching. In the recent past, the main concern surrounding local visual feature descriptors has been their matching accuracy. Hence, such descriptors have been advanced from handcrafted feature descriptors (namely, HOG [32] and SIFT [8]) to deep learned features (namely, CKN-grad [33], MatchNet [25], GeoDesc [34], SOSNet [35], SKAR [36], and DeepCompare [22]) without consideration being given to their computational cost and efficiency.

Efficient descriptors The interest in efficient visual descriptors has risen with the achievement of several highly discriminative feature descriptors such as HOG [32] and SIFT [8]. Specifically, their inability to be used in real-time applications has received attention. There are several studies available regarding this inability and the history of improvements regarding efficiency. Such studies have used SURF [9], compact dither pattern code (CDPC) [4], and salient dither pattern feature (SDPF) [7, 37] descriptors. The common goal of these descriptors is to reduce the dimensionality of the descriptor and to speed up the feature extraction task to improve the efficiency of the overall process. However, real-valued descriptors suffer from a performance bottleneck during similarity matching. Hence, binary descriptors are more favorable for real-time applications.

Binary descriptors Several binary descriptors have been introduced to prevent the performance bottleneck that can occur in real-valued descriptor matching tasks. BRIEF [16], ORB [15], BRISK [5], and FREAK [17] are the most commonly explored binary descriptors and have been used in many real-time applications. These descriptors are efficient in the extraction and matching phases. However, due to the overly simplified pairwise pixel matching process used to construct binary codes, these descriptors are not very robust when dealing with various geometric transformations [13].

Several recent studies have shown that this problem can be solved by using machine learning instead of handcrafting the binary descriptors. For example, LDAHash [28] finds a projection that is designed to simultaneously minimize covariance within the class and maximize covariance across classes while performing additional work to finding an optimal threshold for binarizing the projection. D-BRIEF [27] uses BRIEF [16] on a highly discriminant subspace learned from the input image patches using linear discriminant analysis (LDA) [28], which is quite similar to LDAHash. The latter approach involves the construction of a projection method based on box filters calculated on integral images. Applying box filters to integral images is fast but not especially capable of dealing with rotation. RFD [19], local difference binary (LDB) [38], and BinBoost [39] use AdaBoost to learn a binary encoding scheme. LDB focuses on selecting the optimal sampling pairs, whereas BinBoost jointly optimizes the feature weighting and pooling strategy of each bit using AdaBoost to obtain highly compact and accurate binary descriptors. RFD learns a binary descriptor in a similar way to BinBoost, with a slight difference that the RFD selects receptive fields one by one for each bit, whereas BinBoost uses a linear combination of basic elements to form the whole descriptor. Although RFD and BinBoost have been associated with better performance than many existing binary descriptors, the large number of weak classifiers involved in the computing of each bit makes the computation time required for these descriptors highly unfavorable [39].

Unsupervised or self-supervised hashing algorithms, such as locality sensitive hashing (LSH) [40], semantic hashing (SH) [41], and DeepBit [13], learn binary representation by minimizing the error between the input and the reconstructed image from the learned binary codes. DeepBit learns binary codes based on the features extracted from a pre-trained VGG16 [42]. This learning process is completed by optimizing for three objectives to achieve invariance to transformation, an even distribution, and minimal quantization loss. SH learns binary hash codes for documents by using multilayer RBM, which acts as both a feature extractor and a method for the binary code mapping function. These descriptors are better when no annotated dataset is available, though they do not outperform state-of-the-art supervised methods.

Deep learning Deep learning enables a set of rich features to be learned based on an extensive collection of images to construct a classifier across many classes. CNN learns the weights of a set of convolution filters and the fully connected layers from a gradient descent algorithm. Many state-of-the-art binary descriptors, including DeepBit [13], DSH [43], CNNH [44], Binary L2-Net [29], and DNNH [45], are learned by replacing the last fully connected layers of some well-performed CNN models (e.g., VGG16) with a feature-to-binary code mapping layer. CNN is less capable of dealing with rotation, and it is hard to avoid overfitting when using CNN [13, 31].

NIN Unlike CNN, NIN uses multiple MLPs instead of linear filters to learn deep features. It also uses a global average instead of a fully connected layer for classification purpose. The study in [31] reports that the nonlinear function approximators in NIN can achieve the abstraction ability of the generalized linear models used in CNN while requiring fewer parameters [10]. The concept of having fully connected layers at the feature extraction segment of NIN has inspired the bottleneck design of ResNet [46] and the inception model [47]. Although the bottleneck design and inception model outperform NIN in image classification tasks, the compactness of all three models is comparable. There are fewer than 20 million FLOPs in the original implementation of NIN, whereas the baseline models of the other two contain more than 3 billion FLOPs [46]. Because of the compactness and the great abstraction ability of NIN, it is used as the feature extractor in our approach. The feature description is then achieved by using an RBM.

RBM RBM is a single-layer neural network which typically uses contrast divergence algorithm to learn a mapping between two Bernoulli distributions [30]. It has been later adapted to learn the mapping between Gaussian and Bernoulli distributions which enables representing a distribution of real values using a distribution of binary values. Moreover, RBM can be trained without supervision; hence, it has been chosen to obtain a binary descriptor in this study.

In summary, the existing local descriptors can be broadly classified into either real-valued or binary. Binary descriptors can be further classified into handcrafted, supervised, and unsupervised (or self-supervised) learned descriptors. The method proposed in the present work falls into the self-supervised learning-based binary descriptor category, which is advantageous when fast computation time is critical, and no annotated data are available. Using several public benchmark datasets, we will show that the proposed method, that is based on NIN and RBM, achieves comparable performance to state-of-the-art descriptors in correspondence matching tasks.

3 Proposed approach

In this section, we first explain the proposed DNN architecture and then describe the details of each section of the DNN as well as the novel learning approach. The proposed DNN has two main parts, namely the NIN and RBM. The NIN and RBM are used for learning invariant features and feature-to-binary code mapping, respectively. We propose a new learning approach that both enables the learning of invariant features and reduces the intraclass variance among the binary codes. First, the NIN is initialized with the weights obtained, while it is trained for general image classification purposes using backpropagation with stochastic gradient descent (SGD). Then, the classification layer of the NIN is replaced by a local average layer to transpose the features extracted from the NIN to the input layer of the RBM. Finally, the DNN is trained in three different sessions, as follows:

(1) Training the RBM to obtain a map between the NIN features of image patches to a set of binary codes using a contrast divergence (CD) algorithm,
(2) calculating a representative binary code from the binary codes obtained for the set of geometrically augmented inputs of a single patch, and
(3) fine-tuning the NIN with the geometrically augmented image patches using backpropagation with SGD. The fine-tuning of the NIN and the RBM is repeated alternatively.

Figure 1 shows the proposed DNN architecture and the overall learning approach. It consists of two sections that correspond to the training and fine-tuning processes. The training section shows how the RBM is trained with the outputs of the NIN, the method for generating binary codes, and the representative code. The fine-tuning section shows how the representative code is then used to fine-tune the features in the NIN. In the following sections, we first explain how the initial training of the NIN is executed for feature learning. Second, we describe the approach used to train the RBM. Third, we present the method used to generate the representative code. Finally, we explain how NIN is fine-tuned.

3.1 Training the NIN model

The NIN model differs from conventional CNN models in that the NIN model uses micro neural networks that are located inside each convolution layer, namely mlpconv. The study of NIN in [31] shows that a few layers of mlpconv can achieve a level of performance similar to that of a conventional CNN with significantly more model parameters. Nevertheless, [31] has mentioned that the NIN can still be trained by applying the generic backpropagation algorithm via an annotated image dataset.

Figure 2 shows the NIN model used in the proposed method. It contains only four convolutional layers, each of which consists of a multilayer perceptron and a max-pooling layer. This model is remarkably smaller than most conventional CNNs, including those mentioned in [10, 42]. This feature enables the NIN to learn an extremely compact set of features. The NIN model uses a global average layer at the end of the network to output the class label as given in [31] while in training, but this global average layer is later replaced by a local average pooling layer. Global average pooling is beneficial when obtaining class labels, but it is immensely destructive, making the detailed feature map inaccessible. Meanwhile, local average pooling is less destructive than global average pooling and reduces dimensionality by down sampling the feature map.

We use local average pooling instead of maximum pooling, which is typically recommended in conventional CNNs [10] because local average pooling preserves the non-maximal features that contribute to the final classification. However, as with traditional CNNs, the NIN model learns features using a set of convolutional layers, but each of these layers consists of MLP layers with rectified linear unit (ReLU) activation. Given a convolutional response x_{ij}

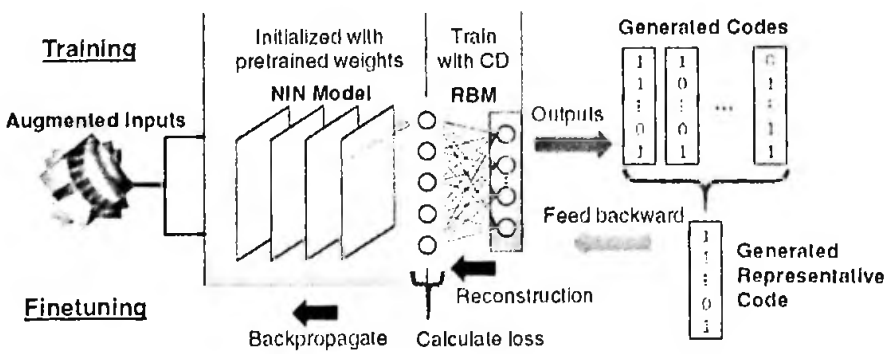

Fig. 1 The proposed DNN architecture with the learning approach

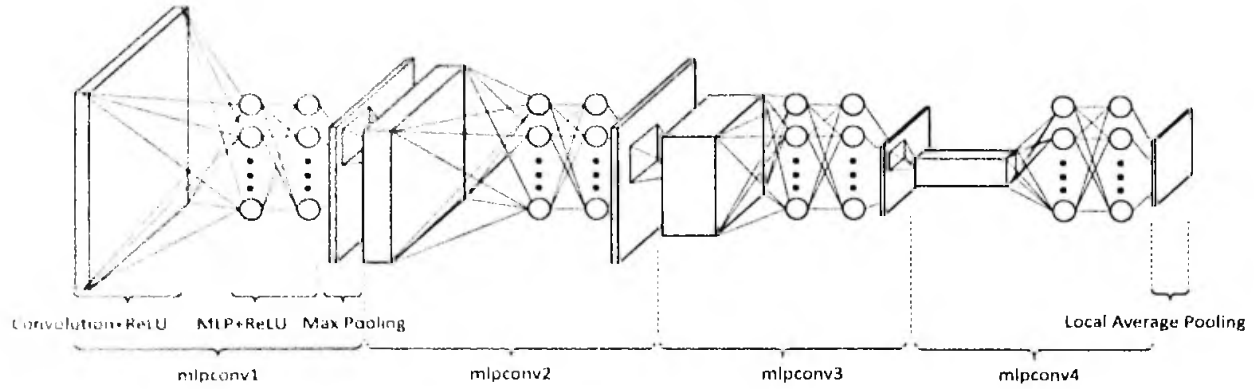

Fig. 2 Modified NIN architecture with the local average pooling layer, which replaces the global average pooling layer of the original network

of the input patch centered at location (i, j), the output can be obtained by using (1),

$$f^1_{i,j,k_1} = \max\left(w_k^T x_{i,j} + b_{k_1}, 0\right)$$

$$f^n_{i,j,k_n} = \max\left(w_k^T f^{n-1}_{i,j} + b_{k_n}, 0\right) \quad (1)$$

where n is the number of layers, w is the weight vector, b is the bias, and k is used to index the channels of the feature map. Then, the global average pooling is obtained from f^n_{i,j,k_n}. In the proposed work, the NIN model consists of four mlpconv layers, each of which has a convolution layer, followed by ReLU activated MLPs. The outputs of the first three mlpconv layers are subsampled by applying the max pool operation. The network is trained with a stochastic gradient descent algorithm (SGD) with cross-entropy loss as specified in [31]. We used a large image dataset with class labels to train the NIN network. Once the training was converged, the global average pooling layer of the NIN was replaced with the local average pooling layer. Then the output was directed to the input layer of the RBM. In the next phase of the training, we temporarily froze the NIN model and then used the output of the NIN to train the RBM.

3.2 Training the RBM

A Gaussian–Bernoulli RBM is a generative stochastic neural network that can map a Gaussian probability distribution to a Bernoulli distribution via unsupervised learning over a set of inputs. An RBM is used in the proposed work to obtain a compact binary code for the mlpconv extracted for an input image patch. The RBM used in the proposed approach takes the real-valued feature map from the NIN model as the input to calculate the values at the hidden unit using (2),

$$p(h_j = 1|x) = \text{sigm}\left(W_j.x + b_j\right) \quad (2)$$

where $p(h_j = 1|x)$ is the probability of being the jth hidden unit, 1, given the input feature map; x, W_j is the jth row of the weight matrix; b_j is the bias of the jth neuron in the hidden layer; and sigm is the sigmoid function. The reconstruction process can be accomplished by (3),

$$p\left(x_k|h\right) = W_k.h + c_k \quad (3)$$

where $p\left(x_k|h\right)$ is the probability of the kth visible unit given the value vector, h, at the hidden units; W_k is the kth column of the weight matrix; and c_k is the bias of the kth neuron in the visible layer. We employ stochastic gradient descent on contrastive divergence with a single step of Gibbs sampling (CD-1) in the initial training of the RBM to reduce the training time. The fine-tuning phase of the RBM uses five steps of Gibbs sampling (CD-5) to estimate the least biased negative samples at a much lower learning rate. The parameters of the RBM are updated according to (4), (5), and (6),

$$W = W + \alpha\left(h(x)x^T - h(\tilde{x})\tilde{x}^T\right) \quad (4)$$

$$b = b + \alpha(h(x) - h(\tilde{x})) \quad (5)$$

$$c = c + \alpha(x - \tilde{x}) \quad (6)$$

where α is the learning rate, x is an example training feature map, $\tilde{x}$ is the negative sample estimated using Gibbs sampling, and the function $h(x)$ is given in (7),

$$h(x) = \text{sigm}(W.x + b) \quad (7)$$

The training of the RBM uses the image patches in their upright position without using any augmentation. Note that the RBM learns feature-to-binary mapping without any supervision. Hence, different orientations and other visual variances of the same image patch may

be converted into a set of binary codes that might not be close enough in terms of Hamming distance to result in better matching. This property can cause the true positive rate to decrease, as our objective is to produce a set of binary codes with the lowest possible intraclass variance in hamming space despite the presence of geometric variations. This problem is overcome by calculating a representative code for the set of codes belonging to the same patch, which is then used to fine-tune the NIN.

3.3 Calculating the representative code

Existing deep learning-based binary descriptors proposed in [13, 26, 48] learn binary hashing on frozen convolutional features. Differently, the proposed method learns binary hashing on a Gaussian–Bernoulli RBM while periodically fine-tuning the nonlinear features in the NIN using backpropagation. The fine-tuning process involves the calculation of a representative binary code, which represents the "patch" itself in the correspondence matching task. This calculation begins by inputting a set of geometrically augmented versions of an image patch into the DNN model and collecting the resulting set of binary codes. Augmentation is performed by the rotation and random cropping of each of the input image patches. The representative code for the collection of binary codes is obtained by (8),

$$H_j^I = \begin{cases} 1 & \Sigma h_j^I \geq |I|/2 \\ 0 & \text{otherwise} \end{cases} \tag{8}$$

where H_j^I is the value of the jth position in the representative binary code of the image patch, I; h_j^I is the value of the jth position in a binary code obtained for an augmented sample of the patch, I; and $|I|$ is the number of augmented samples obtained for the image patch, I, within the batch. The result of H_j^I can be expressed as the mode of the all h_j^I values within the batch. Therefore, the meaning of H^I can be interpreted as the centroid of the distribution of the binary codes generated for different augmentations of the image patch, I, in the Hamming space. Note that the centroid binary code always provides the lowest sum of Hamming distances from the centroid itself to each of the binary codes in the distribution. Therefore, forcing the NIN model to produce a feature map that is expected to yield a binary code that is closer to the centroid should provide less error during the matching phase. This is done by fine-tuning the weights of the NIN to produce a feature map. This process is similar to the RBM reconstruction for the representative code. The next subsection explains the method used for fine-tuning the NIN model.

3.4 Fine-tuning the NIN model

The NIN model uses MLP, which is a universal function approximator used to learn features. Hence, the model is potent in terms of learning optimal features for binary coding. The features are fine-tuned as a means of reducing the variation among the binary codes within different orientations and the shifting of the same image patch. An optimal target from among the inputs should be calculated accordingly. For example, if we input several geometrically transformed variants of a single image patch, the outputs of the NIN should be similar. Also, they should generate almost similar binary codes through the RBM. The new target for the inputs can be generated by reconstructing the feature map f_k^I by giving the representative code H^I to the RBM using (9),

$$f_k^I = W_k.H^I + c_k \tag{9}$$

Note that $\tilde{x}$ in (4), (5), and (6) is obtained from (9) by using a generated binary code instead of H^I. This implies that the training of RBM converges if $\tilde{x} = x$. However, if a properly trained RBM is fed an input like the reconstructed input from a binary code, then it can be expected to generate a binary code similar to the one that is used for reconstruction.

The RBM is temporarily detached from the DNN model, and the parameters of the NIN are fine-tuned using f_k^I as the target of the augmented patches from the original image patch, I, within the training batch. The optimization process uses SGD with mean square error (L2 loss) at a very low learning rate. This fine-tuning process considers the different augmented inputs of a patch to produce a very similar feature map, which is then converted into very similar binary codes by the RBM.

Once the NIN has been fine-tuned, the RBM is integrated into the model and fine-tuned with contrast divergence as explained in Sect. 3.2 with five steps of Gibbs sampling (CD-5). The repetitious fine-tuning of the NIN and RBM parts of the model is computationally expensive. Hence, we empirically terminate the learning process once the CD-5 is converged. Note that, at this point, both the NIN and RBM parts of the network are trained and have undergone at least one fine-tuning process. The following section provides the details of the implementation and the parameters used in the DNN and the learning approach.

3.5 Implementation details

We implement the proposed approach using the TensorFlow [49] library. The proposed NIN model consists of four mlpconv layers, and the initial training process has been

completed using the ImageNet dataset. Images are normalized to the size of 36×36 and then randomly cropped while preparing the model for the correspondence matching task. The number of dimensions of the features obtained from the NIN model is reduced to 3072 using local average pooling. We use random mini batches of a size of 128 for this purpose. The initial training of the RBM uses a learning rate of 0.001 and a momentum of 0.95 via the CD-1 learning algorithm. The fine-tuning of the NIN model uses a learning rate of 0.00001 and a momentum of 0.9. The fine-tuning of the RBM uses a learning rate of 0.00001 and a momentum of 0.95 via the CD-5 learning algorithm. The fine-tuning phase of the RBM involves the preparation of 71 rotated, 10 randomly scaled, and 10 randomly shifted images for each image patch in the dataset. The rotation is completed in increments of 5 degrees, the scaling factor ranges from 0.8 to 1.2, and the shifting range is between -5 and 5 pixels. We select these parameters empirically, and the performance of the model may be improved by optimizing them for specific applications.

4 Experiment

An effective binary descriptor should have several essential characteristics, namely low dimensionality, accurate matching abilities, invariance to different geometrical variances, robustness, and high computational efficiency. Therefore, the proposed DNN architecture combined with the new learning approach is evaluated for visual correspondence matching under extreme variations in geometrical transformation and for computation efficiency. We use several public datasets that contain different challenging properties mentioned above. The task of correspondence matching is evaluated by directly matching the image patches provided in patch-based datasets and using the performance of instance retrieval, which is an application of correspondence matching. Patch similarity is obtained by using the Hamming distance between a pair of binary descriptors. We evaluate the robustness of the proposed DNN by applying it to both the grayscale and color image patches. The computational efficiency of the DNN is assessed based on two factors: the size of the model and the feature extraction time. The size of the model typically affects the initialization time, computation time, and memory requirements, whereas the feature extraction time significantly affects the speed of the overall matching process.

4.1 Datasets

We use the ImageNet ILSVRC2012 [50] dataset, which consists of several millions of images, for the initial training of the NIN model. The visual correspondence matching process is evaluated using Oxford [51], Paris [52], INRIA Holidays [53], Brown [54], RomePatches [33], and HPatches [55] datasets. Further, we use random images from the CIFAR-10 [56] dataset to evaluate computational efficiency. We briefly describe the datasets used for the experiments in this section.

ImageNet ILSVRC2012 [50] contains around 1.2 million images for training, 50,000 for validation, and 100,000 for testing in 1000 different object categories. The large number of hand-annotated images in this dataset enables the learning and generalizing features of the NIN.

The Brown [54] dataset contains a total of 1.2 million images that are collected from three landmarks: Liberty, Notre Dame, and Yosemite. Each of these three subsets includes significantly different views of the corresponding landmark. Each subset consists of more than 400,000 grayscale image patches, which are split into 200,000 training pairs and 100,000 test pairs. Note that exactly half of each set comprises matching pairs, while the other half makes up the set of non-matching pairs. This dataset has been widely used in evaluations of local descriptors. We use all combinations of cross-category training and testing configurations of patches from this dataset, thus enabling us to compare the generality of different binary descriptors. Generality is a significant concern in any self-supervised DNN, which makes the descriptor an ideal choice in a variety of application domains without requiring retraining.

The Brown dataset consists of only grayscale image patches and, thus, does not indicate a model's performance in the presence of colors. Hence, the RomePatches [33] dataset is also used. The RomePatches dataset consists of 20,000 local color image patches collected from several landmarks in Rome. Each of the training and test sets contains a set of 1000 patches, and each patch has 10 different views. This dataset is used for comparing models' utilization of color information for patch matching in the presence of slight geometric variance.

The Oxford [51] dataset contains a total of 5062 images of 11 Oxford landmarks, and the Paris [52] dataset includes a total of 6412 images of 11 Paris landmarks. The images within the classes of both datasets vary in terms of the scale and viewpoint. These two datasets are used to compare the performance of correspondence matching in the presence of drastic geometric variance. Comparisons are made when using the Oxford and Paris datasets by selecting 5 and 55 random query images from each landmark, respectively.

The INRIA Holidays [53] dataset contains a total of 1491 images from 500 categories, each of which represents a distinct scene or object. The images within each category vary by illumination, blurring, scale, and viewpoint. These variations allow comparisons to be made between models' correspondence matching abilities in complicated cases. The evaluation uses 500 query images. All these datasets are

used to evaluate instance retrieval performance by matching the local patches obtained from each of the images.

The HPatches dataset [55] consists of 1.5 million image patches. This dataset has been proposed for mitigating inconsistencies among different local descriptor evaluation protocols. It provides three benchmark tasks, each of which involves a precise evaluation protocol, on the data with three geometric noise levels: *easy*, *hard*, and *tough* [55]. Hence, the HPatches dataset has been used for comparing the performance of our method when faced with known levels of geometric noise.

CIFAR-10 [56] is used in many studies to evaluate computational performance due to its consistent aspect ratio, fixed image dimensions, and the presence of natural images comprising all color channels. It contains 60,000 color images from 10 object categories. In this study, we use CIFAR-10 to compare how quickly the proposed model encodes the image patches into their binary codes.

4.2 Evaluation metric

In this study, we evaluate the proposed method's ability to match the image patches provided by the aforementioned patch-based datasets. We also apply correspondence matching for the instance retrieval task.

The patch-based datasets provide image patches with their ground truths. Hence, the recall and precision metrics are used as the basis of the comparison. We use the 95% error rate, which is defined as the false positive rate at 0.95 of the true positive rate. The 95% error rate is identified by varying the Hamming radius used during similarity matching. This metric has been widely used for comparisons in related studies. We also use the mean average precision (mAP) percentage to evaluate the RomePatches dataset, as this is recommended as the standard metric for this dataset [33]. In addition to the 95% error rate, we utilize the full receiver operating characteristic (ROC) curve to compare the proposed method with methods tested in related works.

The objective of instance retrieval tasks is to retrieve the given instance of the query image from a set of images by matching the local patches, which are obtained using a keypoint detector. We use the mAP metric in the evaluation because it is the standard performance measurement metric used in previous studies. Additionally, we employ the method involving the three tasks provided by the HPatches benchmarking scheme [55] to compare our model with state-of-the-art local descriptors.

The size of a DNN model can be expressed using the number of parameters and their levels of precision, both of which affect storage requirements. We present the number of parameters in millions and the uncompressed model size in megabytes to compare the numeric precision of the parameters. We also report the encoding time of the proposed DNN in milliseconds.

4.3 Results and discussion

In this section, we describe the comparison of the proposed method with existing state-of-the-art local descriptors.

4.3.1 Patch matching task

We use the Brown dataset to evaluate the performance of the patch matching capability of our method. The dataset consists of image patches collected from three categories, namely Notre Dame, Liberty, and Yosemite [54]. The comparison is accomplished using several handcrafted and learned binary descriptors as well as handcrafted and learned real-valued descriptors. Table 1 shows the 95% error rate performance comparison with all combinations of cross-category training and testing configurations with the dimension of the descriptors.

Table 1 presents comparisons of different methods in several categories, such as *real-valued-handcrafted*, *real-valued-supervised*, *binary-valued-supervised*, *binary-valued-handcrafted*, and *binary-valued-self-supervised*. SIFT [8], which is in the first category, uses a histogram of oriented gradients, which contains 128 floating-point values. The floating-point precision leads to an exceedingly long computation time for patch similarity matching. In the second category, the MatchNet [25] uses a DNN with 4 convolution layers for learning the features, a bottleneck layer for dimensionality reduction, and 3 fully connected layers to output pairwise patch similarity. The final metric network in the MatchNet operates on an extremely high-dimensional feature vector with floating-point precision. Additionally, it uses a Siamese-like architecture [57] with two towers of the same network. Hence, supervision is mandatory during training. DeepCompare [22] uses an architecture similar to that of the MatchNet but with an additional decision network. Thus, the evaluation of patch similarity is done on a high-dimensional feature vector with floating-point precision. Although these DNN-based approaches are desirable in terms of patch matching, their use is limited due to a lack of generalizability, which is inherited from the supervised learning and the real-valued descriptors, thus making the patch matching task strikingly slower.

The methods in the third category require supervision during their training phases. Both the LDAHash [28] and D-BRIEF [27] methods show higher error rates than other methods of the same category. Both methods rely on the projection of handcrafted real-valued feature descriptors in Euclidean space to binary descriptors in Hamming space. The methods use sample images to learn the projection, but the results still inherit the performance bottleneck from

their original features in Euclidean space. Of the supervised methods, BinBoost, RFD, and Binary L2-Net [29] significantly outperform all self-supervised methods, including our proposed approach. The superiority of these descriptors is inherited from the domain dependent supervision received during the training. However, our method shows a favorable result in the absence of ground-truth data.

The fourth and fifth categories include all unsupervised methods in which BRISK [5], BRIEF [16], and ORB [15] are handcrafted, whereas DeepBit and DBD-MQ [26] are learned with self-supervision. BRISK, BRIEF, and ORB use simple comparisons of pairs of pixels to encode the image patches. DBD-MQ uses multiple autoencoders to learn features and a binarization function in which DeepBit [58] uses the rigid sign function blindly on the data distribution.

Our method outperforms all the handcrafted and self-supervised binary descriptors for most of the image category combinations. The only exception is that the DBD-MQ achieves a slightly better error rate for the Notre Dame category when the method is trained with Yosemite. We attribute the success of our approach to its ability to reduce the quantization error of the RBM through the contrastive divergence algorithm, thus preserving the discriminative power of the intermediate real-valued features. We also attribute the success of our model to its fine-tuning phase, which forces the model to learn more robust features even when dealing with different forms of geometric variance. Both the DBD-MQ and DeepBit are intended to learn geometrical transformation invariant encoding schemes in their optimization algorithm, but this optimization is applied only for the feature-to-binary encoding network; the VGG convolutional features are not optimized. The standard convolution features have limited rotational invariance [58]. Hence, we believe that unoptimized convolution features cause slightly higher error rates associated with the DBD-MQ and DeepBit when compared with the proposed model.

The results obtained from the Brown dataset are presented in Fig. 3. All pairs of image patches are similar when our method is used, though only the pairs from the first three columns are exact matches. This example shows that the proposed method is robust under blurring (first column), slight illumination variations (second column), and drastic variance in viewpoint (third column). The last column provides some examples of incorrectly matched patches for all three sets. However, the first incorrect pair shares the same content in extremely different orientations, though it is annotated as a mismatching pair. The second mismatching pair also shares some architectural similarities, albeit in different orientations.

The last pair is also somewhat visually similar, with a slight scale difference and around 90° of orientation difference, perhaps because the visual information is overly abstracted from being trained with heavily augmented image patches and because of the fine-tuning process introduced in this study.

Figure 4 shows the ROC curves of different binary descriptors for all combinations of cross-category training and testing configurations of the image patches from the Brown dataset. Our method shows consistent performance for almost all the cases by maintaining a balance between robustness and discriminative power. The Notre Dame and Liberty datasets consist of weak visual similarity over different patches, whereas Yosemite consists of different patches that have a strong visual resemblance, making it the most challenging of the three datasets. Hence, most of the approaches, including our method, show a higher error rate for Yosemite than for the other two datasets. Figure 4 shows that most of the learning-based methods, including our approach, suffer from a high false positive rate when the Hamming radius is increased.

In summary, the results shown in Table 1 and Fig. 4 reveal that the proposed DNN model outperforms all self-supervised binary descriptors using the 95% error rate for datasets that consist of drastic geometrical variance.

We use the RomePatches dataset to evaluate the correspondence matching ability of our method when color patches are considered. The results in Table 2 are obtained by using 1000 query feature points and 9000 target feature points as specified in [33]. We record the mAP percentage of our method and compare it with the mAP percentages of several other binary and real-valued descriptors.

The first three descriptors are real-valued, and both CKN-grad [33] and AlexNet-conv5 [10] require the features to be learned. The SIFT and CKN-grad perform better than any other descriptor, mainly due to the real-valued descriptors, which are capable of coding more discriminative information. CKN-grad uses convolution kernel features, which are somewhat like the convolutional features used in AlexNet-conv3. However, we believe that the CKN-grad performs much better than AlexNet-conv5 due to its much higher-dimensional descriptor, which allows it to encode more discriminative information. FREAK [17] and BRISK [5] are

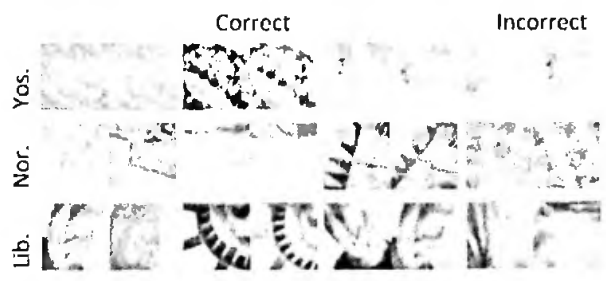

Fig. 3 Correctly matched patches and incorrect matches from the Brown dataset. The topmost row shows 4 sample pairs of matched patches from Yosemite category, the middle one from Notre Dame and the last row of samples is from liberty dataset

Table 1 Patch matching comparison of the proposed method with different descriptors in terms of 95% error rates of all combination of cross-category training and testing configurations of patches from Brown dataset

Method	Size	Supervised	NL	YL	NY	LY	YN	LN	Average	Remarks
SIFT [8]	128	No	36.27	36.27	29.15	29.15	28.09	28.09	31.17	Real-valued, handcrafted
MatchNet [25]	4096	Yes	6.9	10.77	8.39	10.88	5.67	3.87	7.74	Real-valued, use supervised learning, high-dimensional
DeepCompare [22]	256	Yes	4.85	7.20	4.10	5.00	2.11	1.90	4.19	
LDAHash [28]	128	Yes	49.66	49.66	52.95	52.95	51.58	51.58	51.40	Binary-valued, use supervised learning
D-BRIEF [27]	32	Yes	51.30	53.39	46.22	47.29	43.96	43.10	47.54	
BinBoost [39]	64	Yes	20.49	21.67	18.96	22.88	14.54	16.90	19.24	
RFD [19]	598	Yes	19.35	19.40	14.50	16.99	11.68	13.23	15.86	
Binary L2 Net [29]	256	Yes	4.01	6.65	4.04	5.61	2.51	1.90	4.12	
BIRSK [5]	512	No	79.36	79.36	73.21	73.21	74.88	74.88	75.81	Binary-valued, handcrafted
BRIEF [16]	256	No	59.15	59.15	54.96	54.96	54.57	54.57	56.23	
ORB [15]	256	No	59.15	59.15	54.96	54.96	54.57	54.57	56.23	
DeepBit [58]	256	No	32.06	34.41	63.68	57.61	29.60	26.66	40.67	Binary-valued, self-supervised: no ground truths needed
DBD-MQ [26]	256	No	31.10	33.11	57.24	57.15	27.20	25.78	38.59	
Ours	256	No	30.10	32.95	54.25	54.12	27.35	22.19	**36.83**	

The bold value signifies the best average error in the category relavent to the proposed work

N Notre Dame, *L* Liberty, *Y* Yosemite. Ex: *YN* trained on Yosemite and tested on Notre Dame

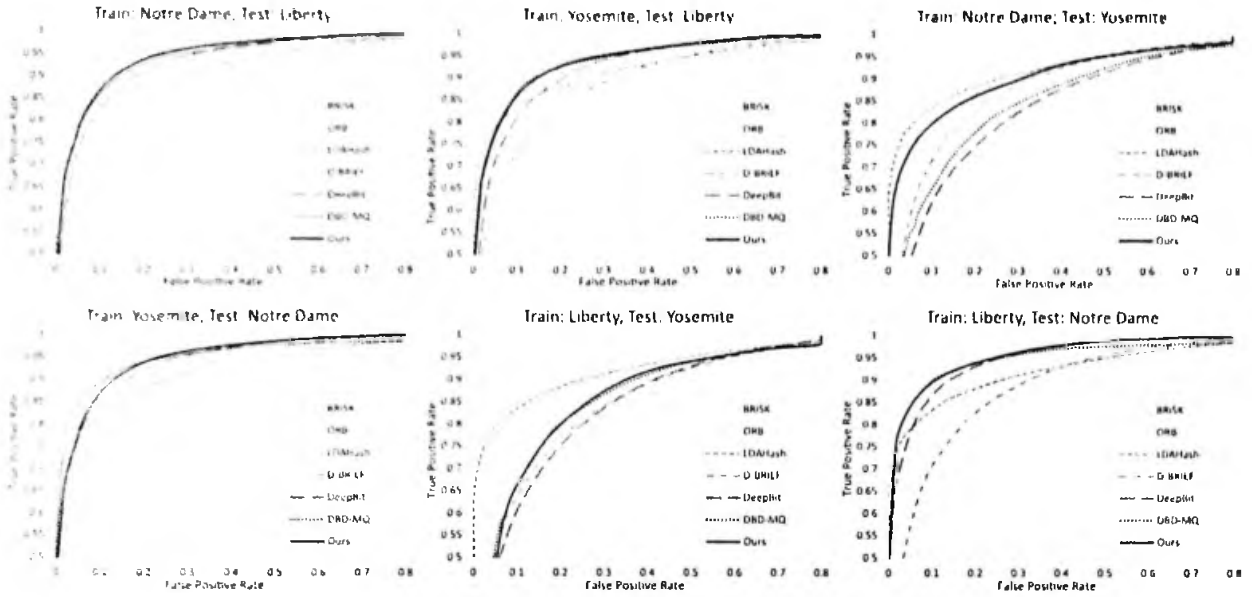

Fig. 4 ROC curves of different binary descriptors with of all combination of all cross-category training and testing configurations of image patches from Brown dataset

whanderafted binary descriptors, and they do not encode any color-related information. Hence, their performance may not be affected by the existence of color. Additionally, they do not handle geometrical invariance in the same way that ORB [15] does. The results show that ORB is better than any other handcrafted binary descriptor. We can attribute this observation to the geometrical invariance property of the ORB, as the patches in the RomePatches dataset also contain variation in terms of viewing angle. DeepBit performs better than any of the handcrafted binary descriptors but does not outperform any real-valued descriptor. DeepBit depends on the features optimized for image classification which is not the intended purpose. Hence, it may perform worse than our method, which optimizes both the encoding scheme and the features. The evaluation results obtained using the RomePatches dataset reveal that our method, which involves a 32-byte binary descriptor, significantly outperforms the other binary descriptors, which involve 32-byte and 64-byte code lengths. Our proposed model also achieves a slightly better mAP percentage than the AlexNet-conv5 real-valued descriptor. Our method is less effective than the remaining real-valued descriptors. However, it is still favorable for color image matching considering descriptor dimensionality.

Table 2 Performance of correspondence matching in terms of mAP (%) of different real-valued and binary descriptors on RomePatches dataset

Descriptor	Bytes	mAP	Remarks
CKN-grad [33]	1024	88.10	Real-valued and extremely high dimension
SIFT [8]	128	87.90	
AlexNet-conv5 [10]	256	49.60	
FREAK [17]	64	23.26	Binary-valued and very low dimension
BRISK [5]	64	31.95	
ORB [15]	32	43.83	
DeepBit [13]	32	46.97	
Ours	32	**50.50**	

The bold value signifies the best mAP in the category relavent to the proposed work

4.3.2 Instance retrieval tasks

In this subsection, we demonstrate the instance retrieval performance of our method by using Paris, Oxford, and Holidays datasets through local image patch matching. The images in these datasets contain much geometrical and illumination variance. Therefore, we apply our proposed method to the correspondence matching task. We first employ a FAST [59] keypoint detector on the images to detect salient patches at four different scales. We then obtain the binary codes to match with Hamming distance between pairs of images. The initial feature learning process is completed using the ImageNet ILSVRC2012 dataset, and the training of the RBM and fine-tuning of the NIN are done by using the Landmark dataset. Table 3 shows the comparisons among two SIFT-based real-valued descriptors, three deep learned binary descriptors, and several other deep learned real-valued descriptors.

Table 3 mAP (%) of instance retrieval performance on Paris, Oxford and Holidays datasets

Method	Paris	Oxford	Holidays	Remarks
SIFT-BOW [52]	46.0	36.4	54.0	Real-valued high-dimensional
SIFT-IFV [60]	–	41.8	62.6	
CNN + aug + ss [61]	79.5	68.0	84.3	Real-valued high-dimensional and larger models
DF.FC1 SL [62]	86.8	46.5	–	
ReDSL.FC1 [62]	94.7	78.3	–	
ITQ 256 [63, 64]	66.3	48.9	67.1	Binary-valued descriptors low-dimensional
ITQ 512 [63, 64]	66.8	50.8	3.9	
NC + PCA 256 [48]	–	55.7	78.9	
NC + PCA 512 [48]	–	55.7	78.9	
DeepBit 256 [13]	82.5	60.3	81.8	
DeepBit 512 [13]	82.9	62.7	82.7	
Ours + FAST 128	79.5	61.2	80.0	
Ours + FAST 256	**83.2**	64.5	**83.8**	
Ours + FAST 512	**83.3**	**64.7**	**83.8**	

The bold values signify the best mAPs in the category relavent to the proposed work

Both the SIFT-BOW [52] and SIFT-IFV [60] use SIFT keypoints during correspondence matching to retrieve the instances. SIFT-IFV uses a fisher model to prepare the descriptor. This model typically performs better than bag-of-word (BOW)-based descriptors [60] (see the first two rows of Table 3). The performance of several deep learned real-valued descriptors such as CNN with augmentation on spatial search technique (CNN+aug+ss) [61], DF.FC1 [62], and ReDSL.FC1 [62] is also mentioned in Table 3 for the comparison. The ITQ [63, 64], neural code (NC)-based PCA [48], and DeepBit are binary descriptors extracted from deep learned features. ITQ uses data-dependent optimization to reduce the quantization error in binarization, whereas NC with PCA uses PCA compression to obtain binary descriptors. Both methods depend on pre-trained CNNs to extract the input features similar to the DeepBit. Hence, the poor geometrical invariance properties of the convolutional features are inherited. Due to the use of mlpconv-based features in the fine-tuning approach, our approach outperforms the other three methods. Table 3 shows that our method, when combined with FAST keypoints, outperforms all deep learned feature-based binary descriptors with 256-bit codes. It also exceeds the mAP percentage of CNN features in the spatial search of the Paris dataset. Also, DF.FC1 and ReDSL.FC1 [62] significantly outperform all methods used for the Paris and Oxford datasets due to the dependency on the real-valued descriptor obtained from the fully connected (FC) layers of the network. However, the binary descriptors can perform matching tasks more quickly than the real-valued FC layer.

In summary, all baseline binary descriptors, including the DeepBit, NC, with PCA, and ITQ, focus on learning efficient binary codes on deep learned and frozen features, which limits the robustness of the codes during correspondence matching between image patches with a high degree of geometrical variance. Our method, in contrast, learns binary codes by fine-tuning the weights of filters to make the features more robust when drastic geometric variations are present. We also compare the 128-bit and 512-bit versions of our descriptor to the 256-bit version (Table 3). We find that the 256-bit version is optimal, as no significant improvement could be gained over the dimensionality extension.

4.3.3 Tasks from the HPatches benchmark

To assess the performance of the proposed method on more diverse data with diverse tasks, we followed the strict evaluation protocols given in the HPatches benchmark [55]. The proposed method is compared with several competing self-supervised and handcrafted binary descriptors as well as state-of-the-art real-valued descriptors on the three tasks as defined in [55]: matching, verification, and retrieval. The matching task is performed so that the descriptor's matches between target and reference images can be measured. The verification task assesses the discriminative power of the descriptor by separating positive pairs of patches from negative pairs of patches. This task is further divided into two experimental settings by taking the pair of images from the same patch sequence (intra-sequence verification) and different patch sequences (inter-sequence verification). The retrieval task measures the descriptors' ability to retrieve similar patches from an extensive collection. The performance of the descriptor on each of the tasks is measured in terms of mean average precision. None of the learned descriptors used in the experiment used patches from the HPatches dataset during training.

Figure 5 shows that our method is comparable with all the binary descriptors—namely the BRIEF [16], ORB [15], DeepBit [13], and DBD-MQ [26]—in all three tasks, and it even outperforms SIFT [8] in the patch verification task. The results show that the performance of all of the existing binary descriptors, including our own, worsens when extreme geometrical and illumination noise is present (i.e., for the *tough* category). However, our method performs better in the *hard* category in which the patches have a significant noise level. Further, SKAR [36], SOSNet [35], and GeoDesc [34], which are the best performing real-valued descriptors on HPatches dataset, show improvements over all three noise levels, whereas the binary descriptors improve only in the *easy* and *hard* categories. The results show that the performances of the

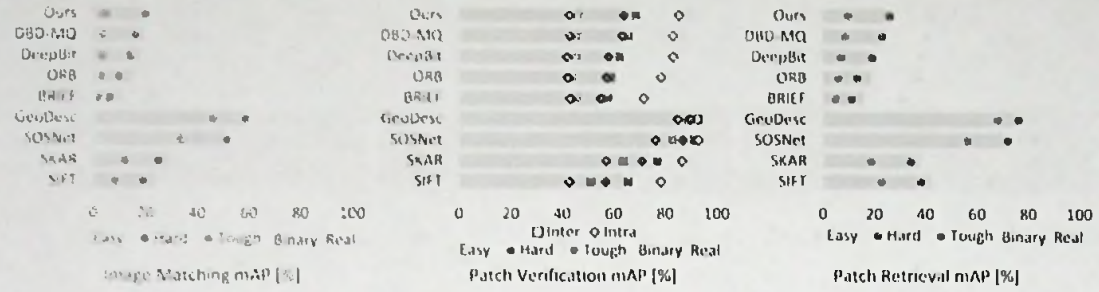

Fig. 5 Matching, verification, and retrieval results on HPatches dataset. Color of the marker indicates the noise levels, namely *easy*, *hard*, and *tough*. The marker corresponds to different experimental settings. The background color indicates the category of the descriptors, i.e., binary-valued or real-valued

Table 4 Comparison of the complexity of the feature extraction models used for learning binary descriptors in terms of the number of parameters and the model size

	AlexNet	VGG16	NIN
Model size	228 MB	537 MB	29 MB
Number of parameters	57 M	134 M	7.3 M

binary descriptors are comparable with the performances of real-valued descriptors in patch verification tasks regardless of whether the patches are taken from the same or different sequences.

4.3.4 Cost of the descriptor

We compare the cost of our method in terms of the number of parameters used, storage requirements, and encoding time. We do not compare the matching speed of our binary descriptor with the speed of other descriptors because the results in this regard do not depend on which method is used but instead on the length of the binary codes. Table 4 summarizes the complexity of the feature extraction models used in this study.

The NIN is exceptionally lightweight, and the performance results demonstrated in the above section reveal that it is well suited to generate a robust and discriminative binary descriptor. In our method, we use one fully connected layer in addition to the NIN model, and we do not utilize a global average pooling layer. Therefore, the addition to the complexity of the original NIN model from the complete binary hashing model in our approach is negligible. Most of the comparable methods used in the experiments described in the above subsection use VGG16 or AlexNet as the feature extractor. Hence, our approach is less expensive than the others when considering the cost of descriptors regardless of whether any nonsignificant performance gaps exist.

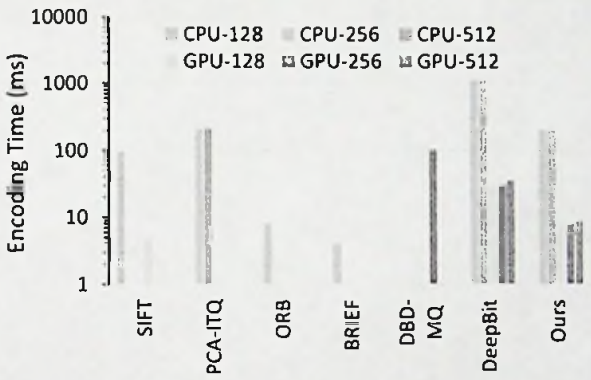

Fig. 6 Average encoding time of an image from CIFAR-10 for different methods

We further evaluate the encoding time of our descriptor (Fig. 6). The average feature extraction time is calculated by using 1000 random images from CIFAR-10 for both CPU and CUDA-enabled GPU. The experiments are carried out in a machine with an Intel Core i7 6700HQ processor with four physical cores and an NVIDIA GTX970M GPU. We use SIFT and a CUDA implementation of SIFT [65], ORB, BRIEF, PCA-ITQ on SIFT, DBD-MQ, and DeepBit on VGG16 features to make comparisons. The encoding time of our method is significantly faster than that of the DeepBit and PCA-ITQ on both CPU and GPU. The SIFT on GPU is faster than all other methods except for the BRIEF. Our method does not outperform the SIFT in any corresponding platforms. However, the speed is comparable when both run on GPU. Further speedup can be observed in descriptor matching tasks owing to the benefits associated with our model being a binary-valued descriptor. Comparison of binary-valued descriptors involve in bitwise operations which are much faster than floating point operations in typical CPU and GPU. DeepBit inherits the computational expensiveness of the VGG16 model. PCA-ITQ inherits the high computational

cost from the PCA projection and SIFT in this experiment. Our method, in contrast, consists of a four-layer NIN model with an RBM that consists of only one layer. Hence, the encoding time is much lower for our method than for the other methods that are compared. This result corroborates the number of parameters used in the models provided in Table 4. Further, the GPU version of our method is also comparable with the ORB on CPU.

4.3.5 Limitations

We found that it is difficult to train the proposed DNN because of the multiple steps involved that must be followed alternately. This difficulty involves in consuming more time in the training phase and selecting the hyperparameters.

Further, we stopped the training once the model had undergone at least one training and one fine-tuning session. This was done because the training process took a very long time. We accept this difficulty as a limitation of our work regardless of the performance exhibited by the empirically selected parameters and the termination criteria.

5 Conclusions

We have presented a lightweight deep neural network (DNN) architecture with a novel approach for learning a compact binary descriptor. Experiments on several public benchmark databases demonstrate that our method performs correspondence matching tasks better than state-of-the-art self-supervised binary descriptors. Further, our method achieved an encoding time comparable to that of state-of-the-art methods while being significantly smaller in model size due to the network in network (NIN) model and the restricted Boltzmann machine (RBM). The better invariant correspondence matching performance of our method can be attributed to the optimization of the both feature-to-binary code mapping and the feature extraction together. This procedure contrasts those of other methods, in which mapping is learned using frozen features. The proposed approach is more favorable than the other compared methods for practical applications due to its smaller size and fast encoding time. The lightweight model may permit the use of the method in resource-constrained devices for computer vision applications such as real-time object recognition. Future works can combine different learned features with the proposed compact binary code learning approach to study its use in different real-world applications.

Acknowledgements This work was carried out with the support of the Senate Research Council, University of Moratuwa, Sri Lanka (Grant No. SRC/16-1), and National Research Council, Sri Lanka (Grant No. 12-017).

Funding This study was funded by the Senate Research Council, University of Moratuwa, Sri Lanka (Grant No. SRC/16-1), and National Research Council, Sri Lanka (Grant No. 12-017).

Compliance with ethical standards

Conflict of Interest Authors, Ravimal Bandara, Lochandaka Ranathunga and Nor Aniza Abdullah, declare that they have no conflict of interest.

References

1. Chamasemani, F.F., Affendey, L.S., Mustapha, N., Khalid, F.: Video abstraction using density-based clustering algorithm. Vis. Comput. **34**, 1299–1314 (2018)
2. Kabbai, L., Abdellaoui, M., Douik, A.: Image classification by combining local and global features. Vis. Comput. **35**, 679–693 (2019)
3. Ali, M., Jones, M.W., Xie, X., Williams, M.: TimeCluster: dimension reduction applied to temporal data for visual analytics. Vis. Comput. **35**, 1013–1026 (2019)
4. Ranathunga, L., Zainuddin, R., Abdullah, N.A.: Performance evaluation of the combination of Compacted Dither Pattern Codes with Bhattacharyya classifier in video visual concept depiction. Multimed. Tools Appl. **54**, 263–289 (2011)
5. Leutenegger, S., Chli, M., Siegwart, R.Y.: BRISK: binary robust invariant scalable keypoints. In: 2011 International Conference on Computer Vision, pp. 2548–2555. IEEE (2011)
6. Kalpana, J., Krishnamoorthi, R.: Color image retrieval technique with local features based on orthogonal polynomials model and SIFT. Multimed. Tools Appl. **75**, 49–69 (2016)
7. Bandara, A., Ranathunga, L., Abdullah, N.: Invariant properties of a locally salient dither pattern with a spatial-chromatic histogram. In: 2013 8th IEEE International Conference on Industrial and Information Systems (ICIIS), pp. 304–308. IEEE (2013)
8. Lowe, D.G.: Distinctive image features from scale-invariant keypoints. IJCV **60**(20), 91–110 (2004)
9. Bay, H., Ess, A., Tuytelaars, T., Gool, L.V.: Speeded-up robust features (SURF). CVIU **110**(3), 346–359 (2008)
10. Krizhevsky, A., Sutskever, I., Hinton, G.E.: Imagenet classification with deep convolutional neural networks. In: Advances in Neural Information Processing Systems, pp. 1097–1105 (2012)
11. Chen, L., Wang, R., Yang, J., Xue, L., Hu, M.: Multi-label image classification with recurrently learning semantic dependencies. Vis. Comput. **35**, 1361–1371 (2019)
12. Vinyals, O., Blundell, C., Lillicrap, T., Wierstra, D., et al.: Matching networks for one shot learning. In: Advances in Neural Information Processing Systems, pp. 3630–3638 (2016)
13. Lin, K., Lu, J., Chen, C.-S., Zhou, J., Sun, M.-T.: Unsupervised deep learning of compact binary descriptors. IEEE Trans. Pattern Anal. Mach. Intell. **41**(6), 1501–1514 (2018)
14. Juan, L., Gwun, O.: A comparison of SIFT, PCA-SIFT and SURF. IJIP **3**, 143–152 (2009)
15. Rublee, E., Rabaud, V., Konolige, K., Bradski, G.: ORB: an efficient alternative to SIFT or SURF, pp. 2564–2571 (2011)

16. Calonder, M., Lepetit, V., Strecha, C., Fua, P.: Brief: binary robust independent elementary features. In: European conference on computer vision, pp. 778–792. Springer (2010)
17. Alahi, A., Ortiz, R., Vandergheynst, P.: Freak: fast retina keypoint. In: 2012 IEEE Conference on Computer Vision and Pattern Recognition, pp. 510–517. IEEE (2012)
18. Trzcinski, T., Christoudias, M., Lepetit, V.: Learning image descriptors with boosting. IEEE Trans. Pattern Anal. Mach. Intell. **37**, 597–610 (2015)
19. Lan, B., Kong, Q., Trzcinski, T., Wang, Z., Pan, C., Fua, P.: Receptive fields selection for binary feature description. IEEE Trans. Image Process. **23**, 2583–2595 (2014)
20. Zhang, S., Tian, Q., Huang, Q., Gao, W., Rui, Y.: USB: ultra-short binary descriptor for fast visual matching and retrieval. IEEE Trans. Image Process. **23**, 3671–3683 (2014)
21. Zheng, L., Wang, S., Tian, Q.: Coupled binary embedding for large-scale image retrieval. IEEE Trans. Image Process. **23**, 3368–3380 (2014)
22. Zagoruyko, S., Komodakis, N.: Learning to compare image patches via convolutional neural networks. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 4353–4361 (2015)
23. Zbontar, J., LeCun, Y.: Stereo matching by training a convolutional neural network to compare image patches. J. Mach. Learn. Res. **17**, 2 (2016)
24. Kumar, B., Carneiro, G., Reid, I., et al. Learning local image descriptors with deep siamese and triplet convolutional networks by minimising global loss functions. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 5385–5394 (2016)
25. Han, X., Leung, T., Jia, Y., Sukthankar, R., Berg, A.C.: Matchnet: unifying feature and metric learning for patch-based matching. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 3279–3286 (2015)
26. Duan, Y., Lu, J., Wang, Z., Feng, J., Zhou, J.: Learning deep binary descriptor with multi-quantization. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pp. 1183–1192 (2017)
27. Trzcinski, T., Lepetit, V.: Efficient discriminative projections for compact binary descriptors. In: European Conference on Computer Vision. Springer, pp. 228–242 (2012)
28. Strecha, C., Bronstein, A., Bronstein, M., Fua, P.: LDAHash: improved matching with smaller descriptors. IEEE Trans. Pattern Anal. Mach. Intell. **34**, 66–78 (2012)
29. Tian, Y., Fan, B., Wu, F.: L2-net: deep learning of discriminative patch descriptor in euclidean space. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 661–669 (2017)
30. Cho, K.H., Raiko, T., Ilin, A.: Gaussian-Bernoulli deep boltzmann machine. In: The 2013 International Joint Conference on Neural Networks (IJCNN) pp. 1–7. IEEE (2013)
31. Lin, M., Chen, Q., Yan, S.: Network in network (2013). arXiv :1312.4400
32. Sinha, A., Banerji, S., Liu, C.: New color GPHOG descriptors for object and scene image classification. Mach. Vis. Appl. **25**, 361–375 (2014)
33. Paulin, M., Douze, M., Harchaoui, Z., Mairal, J., Perronin, F., Schmid, C.: Local convolutional features with unsupervised training for image retrieval. In: Proceedings of the IEEE International Conference on Computer Vision, pp. 91–99 (2015)
34. Luo, Z., Shen, T., Zhou, L., Zhu, S., Zhang, R., Yao, Y., Fang, T., Quan, L.: Geodesc: learning local descriptors by integrating geometry constraints. In: Proceedings of the European Conference on Computer Vision (ECCV), pp. 168–183 (2018)
35. Tian, Y., Yu, X., Fan, B., Wu, F., Heijnen, H., Balntas, V.: SOSNet: second order similarity regularization for local descriptor learning. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 11016–11025 (2019)
36. Markuš, N., Pandžić, I., Ahlberg, J.: Learning local descriptors by optimizing the keypoint-correspondence criterion: applications to face matching, learning from unlabeled videos and 3D-shape retrieval. IEEE Trans. Image Process. **28**, 279–290 (2018)
37. Bandara, R., Ranathunga, L., Abdullah, N.A.: Nature inspired dimensional reduction technique for fast and invariant visual feature extraction. IJATCSE **8**, 696–706 (2019). https://doi.org/10.30534/ijatcse/2019/57832019
38. Yang, X., Cheng, K.-T.: Local difference binary for ultrafast and distinctive feature description. IEEE Trans. Pattern Anal. Mach. Intell. **36**, 188–194 (2014)
39. Trzcinski, T., Christoudias, M., Fua, P., Lepetit, V.: Boosting binary keypoint descriptors. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 2874–2881 (2013)
40. Andoni, A., Indyk, P.: Near-optimal hashing algorithms for approximate nearest neighbor in high dimensions. In: 47th Annual IEEE Symposium on Foundations of Computer Science, 2006, FOCS'06, pp. 459–468. IEEE (2006)
41. Salakhutdinov, R., Hinton, G.: Semantic hashing. Int. J. Approx. Reason. **50**, 969–978 (2009)
42. Simonyan, K., Zisserman, A.: Very deep convolutional networks for large-scale image recognition (2014). arXiv:1409.1556
43. Liu, H., Wang, R., Shan, S., Chen, X.: Deep supervised hashing for fast image retrieval. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 2064–2072 (2016)
44. Xia, R., Pan, Y., Lai, H., Liu, C., Yan, S.: Supervised hashing for image retrieval via image representation learning. In: Twenty-eighth AAAI conference on artificial intelligence. Association for the Advancement of Artificial Intelligence (AAAI) (2014)
45. Lai, H., Pan, Y., Liu, Y., Yan, S.: Simultaneous feature learning and hash coding with deep neural networks. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 3270–3278 (2015)
46. He, K., Zhang, X., Ren, S., Sun, J.: Deep residual learning for image recognition. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 770–778 (2016)
47. Szegedy, C., Vanhoucke, V., Ioffe, S., Shlens, J., Wojna, Z.: Rethinking the inception architecture for computer vision. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 2818–2826 (2016)
48. Babenko, A., Slesarev, A., Chigorin, A., Lempitsky, V.: Neural codes for image retrieval. In: European Conference on Computer Vision, pp. 584–599. Springer (2014)
49. Abadi, M., Barham, P., Chen, J., Chen, Z., Davis, A., Dean, J., Devin, M., Ghemawat, S., Irving, G., Isard, M., et al.: Tensorflow: a system for large-scale machine learning. In: OSDI'16: Proceedings of the 12th USENIX conference on Operating Systems Design and Implementation, pp. 265–283. USENIX Association (2016)
50. Russakovsky, O., Deng, J., Su, H., Krause, J., Satheesh, S., Ma, S., Huang, Z., Karpathy, A., Khosla, A., Bernstein, M., et al.: Imagenet large scale visual recognition challenge. Int. J. Comput. Vis. **115**, 211–252 (2015)
51. Philbin, J., Chum, O., Isard, M., Sivic, J., Zisserman, A.: Object retrieval with large vocabularies and fast spatial matching. In: IEEE Conference on Computer Vision and Pattern Recognition, 2007, CVPR'07, pp. 1–8. IEEE (2007)
52. Philbin, J., Chum, O., Isard, M., Sivic, J., Zisserman, A.: Lost in quantization: improving particular object retrieval in large scale image databases. In: 2008 IEEE Conference on Computer Vision and Pattern Recognition, pp. 1–8. IEEE (2008)

53. Jegou, H., Douze, M., Schmid, C.: Hamming embedding and weak geometric consistency for large scale image search. In: European Conference on Computer Vision, pp. 304–317. Springer (2008)
54. Brown, M., Hua, G., Winder, S.: Discriminative learning of local image descriptors. IEEE Trans. Pattern Anal. Mach. Intell. **33**, 43–57 (2011)
55. Balntas, V., Lenc, K., Vedaldi, A., Mikolajczyk, K.: HPatches: a benchmark and evaluation of handcrafted and learned local descriptors. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 5173–5182 (2017)
56. Krizhevsky, A., Hinton, G.: Learning multiple layers of features from tiny images, vol. 1. Citeseer (2009)
57. Liu, J., Rosin, P.L., Sun, X., Xiao, J., Liao, Z.: Image-driven unsupervised 3D model co-segmentation. Vis. Comput. **35**, 909–920 (2019)
58. Lin, K., Lu, J., Chen, C.-S., Zhou, J.: Learning compact binary descriptors with unsupervised deep neural networks. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 1183–1192. (2016)
59. Rosten, E., Drummond, T.: Machine learning for high-speed corner detection. In: European Conference on Computer Vision, pp. 430–443. Springer (2006)
60. Jegou, H., Perronnin, F., Douze, M., Sánchez, J., Perez, P., Schmid, C.: Aggregating local image descriptors into compact codes. IEEE Trans. Pattern Anal. Mach. Intell. **34**, 1704–1716 (2012)
61. Sharif Razavian, A., Azizpour, H., Sullivan, J., Carlsson, S.: (2014) CNN features off-the-shelf: an astounding baseline for recognition. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops, pp. 806–813
62. Wan, J., Wang, D., Hoi, S.C.H., Wu, P., Zhu, J., Zhang, Y., Li, J.: Deep learning for content-based image retrieval: a comprehensive study. In: Proceedings of the 22nd ACM international conference on multimedia, pp. 157–166. ACM (2014)
63. Yunchao, G., Lazebnik, S., Gordo, A., Perronnin, F.: Iterative quantization: a procrustean approach to learning binary codes for large-scale image retrieval. IEEE Trans. Pattern Anal. Mach. Intell. **35**, 2916–2929 (2013)
64. Reddy Mopuri, K., Venkatesh Babu, R.: Object level deep feature pooling for compact image representation. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops, pp. 62–70 (2015)
65. Björkman, M., Bergström, N., Kragic, D.: Detecting, segmenting and tracking unknown objects using multi-label MRF inference. Comput. Vis. Image Underst. **118**, 111–127 (2014)