Received November 25, 2021, accepted December 21, 2021, date of publication December 23, 2021,
date of current version January 4, 2022.

Digital Object Identifier 10.1109/ACCESS.2021.3138168

Efficient Local Image Descriptors Learned With Autoencoders

NINA ŽIŽAKIĆ, (Member, IEEE), AND ALEKSANDRA PIŽURICA, (Senior Member, IEEE)
Department of Telecommunications and Information Processing, TELIN–GAIM, Ghent University, 9000 Ghent, Belgium

Corresponding author: Nina Žižakić (nina.zizakic@ugent.be)

This work was supported by the Flemish Government (AI Research Program).

ABSTRACT Local image descriptors play a crucial role in many image processing tasks, such as object tracking, object recognition, panorama stitching, and image retrieval. In this paper, we focus on learning local image descriptors in an unsupervised way, using autoencoders and variational autoencoders. We perform a thorough comparative analysis of these two approaches along with an in-depth analysis of the most relevant hyperparameters to guide their optimal selection. In addition to this analysis, we give insights into the difficulties and the importance of selecting right evaluation techniques during the unsupervised learning of the local image descriptors. We explore the extent to which a simple perceptual metric during training can predict the performance on tasks such as patch matching, retrieval and verification. Finally, we propose an improvement to the encoder architecture that yields significant savings in memory complexity, especially in single-image tasks. As a proof of concept, we integrate our descriptor into an inpainting algorithm and illustrate its results when applied to the virtual restoration of master paintings. The source code required to reproduce the presented results has been made available as a repository on GitHub (https://github.com/nimpy/local-img-descr-ae).

INDEX TERMS Local image descriptor, autoencoder, variational autoencoder, inpainting, unsupervised deep learning.

I. INTRODUCTION

Finding a compact representation of a small patch in an image, i.e., finding a local image descriptor, is a crucial building block of various image processing tasks. Image inpainting, denoising, stitching, object tracking, motion estimation, and saliency detection are all examples of tasks where local image descriptors are used. For example, in object tracking and motion estimation, local image descriptors are used to identify corresponding objects in subsequent frames and to associate those objects between frames. Image inpainting relies on using parts of an image to fill in areas that may be missing or corrupted. By identifying the parts of the image that are the most similar to the parts of the image bordering the missing or corrupted areas, local image descriptors can significantly improve the performance of an inpainting algorithm.

Traditionally, the approach to designing local image descriptors has involved using hand-crafted features, with SIFT [1] and its variants (SURF [2], GLOH [3], PCA-SIFT [4]) being arguably among the most popular. Recent advancements in the field of deep learning, however, have caused a shift from traditional approaches to learning-based approaches for designing descriptors. Various learning-based methods have emerged in the past couple of years [5]–[10] and have demonstrated performance superior to the hand-crafted descriptors on different benchmarks [11], [12].

Nonetheless, despite the apparent improved performance on benchmarks, comparative evaluation studies such as [13] still suggest that traditional hand-crafted descriptors such as SIFT can outperform learned descriptors when being part of large image processing tasks. Reportedly, SIFT is still predominantly a descriptor of choice in practical applications. This has been attributed to the fact that the learned descriptors were trained too generally and thus often underperform on a specific image processing task compared to the pre-designed ones [7]. Furthermore, since the majority of learning-based approaches are supervised, they require labelled data, which is often unavailable when creating descriptors for specific

The associate editor coordinating the review of this manuscript and approving it for publication was Chun-Wei Tsai.

imaging processing tasks. An alternative is to turn to unsupervised models such as autoencoders (AEs) [14] and variational autoencoders (VAEs) [15]. Here we take this approach.

Autoencoders are neural networks where an encoder and decoder are simultaneously trained to encode data into and decode data from a compact encoding. Since recently, there has been a surge of AE-based and VAE-based architectures in various computer vision tasks [15]–[18]. However, they have been less studied so far for learning local image descriptors. The work of Chen *et al.* [19] reported an AE-based descriptors showing promising results, however, the methods they used are no longer considered state of the art. We proposed in our previous work an approach for learning local image descriptors based on convolutional autoencoders [20], [21] and variational autoencoders [22]. In our experience, both AEs and VAEs have shown promising results for learning local image descriptors, however, a thorough comparative analysis is still missing. On one hand, VAEs facilitate learning a smoother latent space that is easy for interpolation – a property that seems useful for descriptors. VAEs are, however, more complex and more difficult to work with. They introduce new hyperparameters (such as β – the weight of the Kullback–Leibler divergence term in the loss function), take slightly longer on average to train, and are not deterministic. To the best of our knowledge, a comprehensive comparison between the two methods does not exist in the literature. In this article, we compare the performance of AE-based and VAE-based descriptors and consider the advantages and disadvantages of the two approaches. Furthermore, we investigate some novel ways of optimising the performance of the two descriptor types. For example, we explore the use of a perceptual loss function and how different hyperparameters, such as activation functions and the level of data augmentation, impact the performance of learned descriptors.

Another issue we encountered during developing a local image descriptor using a (variational) autoencoder is not being able to tell how well it will perform as a descriptor once it is trained. The autoencoder is trained by minimising the loss function which measures some difference (e.g. mean squared error or binary cross-entropy) between the input and the output of the network. However, we have noticed that having the lowest such difference does not necessarily lead to the best performing descriptors. It is only after evaluating the descriptor on a benchmark (such as HPatches [11]), which takes a very long time (sometimes as long as the training itself), that we know how good the descriptor is. In this paper, we try to provide some insights into the descriptor evaluation process and look at what metrics (that measure the difference between the input and the output of a (V)AE) are the best "proxies" for how well the descriptor will perform.

We also propose modifying the architecture of the autoencoder in order to yield a more efficient descriptor design for applications with many patch comparisons within a single image. Our specific network architecture produces a special image representation that we refer to as the *intermediate representation* (IR). The IR is a compact way of storing the descriptors of all the patches of an image because the descriptors of overlapping patches overlap themselves. Extracting a descriptor from the IR is done fast using only one max-pooling operation.

We apply this architectural change to the descriptor that was yielded as the best from our thorough analysis. We show that this change does not degrade the performance of the descriptor as measured by HPatches benchmark, but offers improvements in terms of computational memory in the applications with many patch comparisons within a single image, such as inpainting.

As a proof of concept for this, we integrate this descriptor into an existing inpainting algorithm [23] to show these improvements. We hypothesise that the improved inpainting results come from the fine-tuning of the descriptor on types of images to be used in the inpainting, made possible due to the unsupervised nature of our descriptor. To achieve such fine-tuning with other (supervised) descriptors, it would be necessary to have a labelled set for the type of images that need to be inpainted, which is unrealistic in most cases. As a case study in this paper, we used high-resolution photographs of the panels of Ghent Altarpiece [24], [25], on which we fine-tuned the descriptor and tested our improved inpainting algorithm.

Parts of this work have been accepted for presentation at conferences [20]–[22], but these only relate to some aspects of learning descriptors with autoencoders, and do not compare different methods nor present insights into their evaluation. The goal of this paper is to provide a united, thorough analysis of all aspects of learning local image descriptors using (variational) autoencoders, implement the proposed IR architectural change on the best model, and, as a proof of concept, show the descriptor's benefits by integrating it into an inpainting algorithm. The main contributions of the paper can be summarised as follows:

1) We present a thorough comparison between autoencoder-based and variational-autoencoder–based approach for learning local image descriptors and analysis of hyperparameter importance for obtaining a successful descriptor. To our knowledge, no comparison between AEs and VAEs in this context has been carried out.
2) We propose using some new techniques that have not been used before for learning descriptors. In particular, we incorporate perceptual loss, which proved to significantly increase the performance of the descriptor. We perform a thorough hyperparameter analysis to establish which hyperparameters are the most important when learning a descriptor using a (variational) autoencoder.
3) We provide important insights into evaluation metrics for the learned local image descriptors and propose a

rapid approximate evaluation method for descriptors learned with autoencoders that shows high correlation with the established benchmarks such as HPatches.

4) An important technical novelty is a specific autoencoder architecture, which enables the acquirement of the *intermediate representation* (IR) structure. This approach gives the most benefits in the applications where a single image is used (such as inpainting).
5) As a proof of concept, we incorporate these novelties into an inpainting algorithm which led to its acceleration and to improved inpainting results.

The rest of this paper is organized as follows. In the following section, we discuss the related work on local image descriptors and introduce the preliminaries (autoencoders and variational autoencoders). In Section III, we compare the autoencoder-based and variational-autoencoder–based approach to learning descriptors, and carry out a hyperparameter study for descriptors learned in this way. In Section IV we study the evaluation of descriptors and propose an approach for fast approximate evaluation of the trained models. We propose a modification to the encoder architecture of the autoencoders in Section V, and conduct an empirical comparison of our architecture to the traditional encoder architectures. In Section VI, we integrate our proposed descriptor into an inpainting algorithm. We conclude the work in Section VII.

II. RELATED WORK

A. LOCAL IMAGE DESCRIPTORS

Traditionally, local image descriptors were designed to use hand-crafted features. The most prominent kind of hand-crafted descriptors used to be the distribution-based descriptors, which are using distributions of image properties such as gradients to represent patches. They are sometimes referred to as the SIFT-based descriptors [11], after the paramount work by Lowe [1], one of the most cited papers in computer science. Other prominent hand-crafted descriptors include HOG [26], SURF [2], GLOH [3], PCA-SIFT [4], RSIFT [27] and DAISY [28], to name a few. Some hand-crafted descriptors produce encodings in Hamming space – they are most commonly referred to as binary descriptors. Examples of such descriptors are BRIEF [29], ORB [30], BRISK [31], FREAK [32] and LDAHash [33].

In recent years, a popular approach to solving many image processing challenges, including the design of local image descriptors, has shifted towards the use of deep learning methods. The first learned local image descriptors arose from the necessity to find a way of choosing parameters in the hand-crafted methods that does not involve hand-tuning [34], [35]. As machine learning, and in particular, deep learning methods developed further, a growing number of learning-based descriptors started emerging. Nowadays, learned descriptors are mostly supervised methods, learning useful features on pairs of similar and dissimilar patches, striving for a high correlation between the similarity of the patches and the similarity of their descriptors. Most recent methods involve the use of convolutional neural networks [8], [10], [12], [36], some of which are siamese networks [5], [37], or triplets [6], [9].

Learned descriptors have been shown to outperform some hand-crafted ones on benchmarks [11], [12]. However, despite many advancements in the learning approach and their superior performance on benchmarks, hand-crafted descriptors still perform comparably or better than the learned descriptors in practical context [13]. Partly, this can be attributed to the selection of the training set (e.g. descriptors learned on general datasets such as ImageNet may not necessarily work well on some specific medical imaging datasets).

Unsupervised learning methods such as autoencoders do not suffer from dependence on labelled data. Chen *et al.* were the first to apply autoencoders to learn local image descriptors [19]. Their method shows promising results, however, the techniques they are using are today no longer modern (e.g. both encoder and decoder of their AE consist of only one (fully-connected) layer, no convolutional layers are used).

In our previous works we have proposed using autoencoders [20], [21] and variational autoencoders [22] to learn local image descriptors that can be specifically tailored for particular applications. Both classical and variational autoencoders have shown promising results, however, it remains unclear which ones are more suitable for learning descriptors. To the best of our knowledge, a comprehensive comparison between the two methods does not exist in the literature, neither specifically for learning local image descriptors, nor a general study. In this paper, we offer such a thorough comparison between the autoencoders and variational autoencoders approach for learning local image descriptors. Furthermore, we examine other aspects of (variational) autoencoder architecture and provide an analysis of what are the most important hyperparameters for optimising descriptors. Encouraged by the recent result from [38], who demonstrated benefits from using a perceptual loss function with the autoencoders that learn embeddings for downstream prediction tasks, we include perceptual loss into our analysis of AE-based local image descriptors. This proved to lead to an improved performance of the descriptors. To the best of our knowledge, we are the first to use a perceptual loss when training autoencoders to learn a local image descriptor.

B. AUTOENCODERS

Autoencoders (AEs) [14] are unsupervised neural networks used for learning compact representations of data. An autoencoder consists of two parts, an encoder and a decoder, and is trained by setting the target output values to be equal to the input values, while imposing constraints on the middle layer. Formally, an autoencoder consists of an encoder $\mathcal{E}$ and a decoder $\mathcal{D}$ which are neural networks parametrised with their weights $\mathbf{w}_{\mathcal{E}}, \mathbf{w}_{\mathcal{D}}$ and biases $\mathbf{b}_{\mathcal{E}}, \mathbf{b}_{\mathcal{D}}$, respectively. The autoencoder is trained to minimise the loss

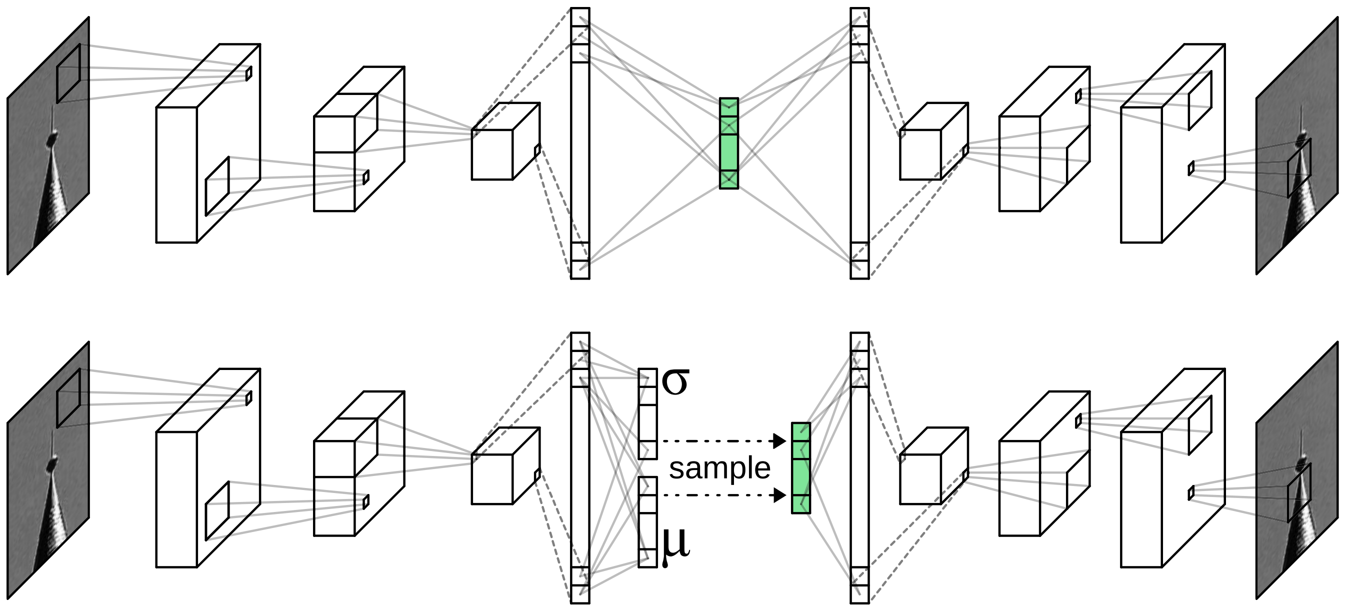

FIGURE 1. **Architecture of an autoencoder (top) and a variational autoencoder (bottom). The bottleneck layer (which is the patch encoding produced by the descriptor) is shown in green.**

function J w.r.t. $\mathbf{w}_{\mathcal{E}}, \mathbf{b}_{\mathcal{E}}, \mathbf{w}_{\mathcal{D}}, \mathbf{b}_{\mathcal{D}}$:

$$\min_{\mathbf{w}_{\mathcal{E}},\mathbf{b}_{\mathcal{E}},\mathbf{w}_{\mathcal{D}},\mathbf{b}_{\mathcal{D}}} J(\mathcal{P}, \mathcal{E}_{\mathbf{w}_{\mathcal{E}},\mathbf{b}_{\mathcal{E}}}, \mathcal{D}_{\mathbf{w}_{\mathcal{D}},\mathbf{b}_{\mathcal{D}}}) = \sum_{\mathbf{p}\in\mathcal{P}} \mathcal{L}(\mathbf{p}, \mathcal{D}_{\mathbf{w}_{\mathcal{D}},\mathbf{b}_{\mathcal{D}}}(\mathcal{E}_{\mathbf{w}_{\mathcal{E}},\mathbf{b}_{\mathcal{E}}}(\mathbf{p}))) \quad (1)$$

where $\mathbf{p} \in \mathcal{P}$ is a data sample (in our case an image patch) and $\mathcal{L}$ is some loss function, typically binary cross-entropy $\mathcal{L}_{BCE}(\mathbf{p}, \mathbf{p}') = -\frac{1}{N}\sum_{i=1}^{N} p'_i \log p_i + (1 - p'_i)\log(1 - p_i)$. Autoencoders working with image data usually consist of convolutional layers, with an optional fully-connected layer at the end of the encoder and the beginning of the decoder.

A shortcoming of classical autoencoders is that they have no way of enforcing the continuity of the latent space and are thus unable to guarantee that the learned encodings are useful, i.e., that they possess the similarity preserving property – an important property for local image descriptors.

C. VARIATIONAL AUTOENCODERS

In contrast to classical autoencoders variational autoencoders (VAEs) [15] are probabilistic models that assume a prior distribution of the latent space, giving significant control over how we want to model the latent distribution. The data x has a likelihood $p(x|z)$ (the decoder distribution) that is conditioned on latent variables z. The posterior (typically Gaussian) is approximated with a family of distributions $q(z|x)$ (the encoder distribution). Apart from minimising the reconstruction loss, VAEs also minimise the Kullback–Leibler (KL) divergence between the true posterior $p(z)$ and its approximation $q(z|x)$. Given a dataset $X = \{x_1, x_2, \ldots, x_n\}$, the goal of a VAE is to minimise the negative log-likelihood lower bound:

$$\mathcal{L}(\theta, \phi; x) = -\mathbb{E}_{q_\phi(z|x)}[\log p_\theta(x|z)] + D_{KL}[q_\phi(z|x)||p_\theta(z)], \quad (2)$$

where the encoder and decoder distributions are parametrised by ϕ and θ, respectively. The first term promotes a good reconstruction of the input data samples, while the second term enforces that the distribution of the latent space is as close as possible to the multivariate Gaussian distribution.

A generalisation of a variational autoencoder named β-VAE [17] extends the loss function from (2) with a weight on the second term, to allow a trade-off between the reconstruction quality and the smoothness of the latent space:

$$\mathcal{L}(\theta, \phi; x) = -\mathbb{E}_{q_\phi(z|x)}[\log p_\theta(x|z)] + \beta D_{KL}[q_\phi(z|x)||p_\theta(z)].$$

A smoother latent space results in more meaningful distances between the encodings, thus increasing their similarity-preserving property.

In the next section, we will show a comparison between AE and VAE approach for learning local image descriptors, discuss the choice of these two types of autoencoders including their advantages and disadvantages, and discuss how to select the β parameter.

III. AE VERSUS VAE AND A HYPERPARAMETER STUDY

Here we present an in-depth study on learning local image descriptors using (variational) autoencoders. First, we describe the selected hyperparameters (Section III-A), then the experimental setup (Section III-B) and, finally, the empirical evaluation (Section III-C).

Why the focus on AEs versus VAEs? Classical autoencoders serve as a baseline for compacting the essential information from the input into a lower-dimensional representation. However, their lack of ability to ensure the continuity of the latent space may be a drawback when learning local image descriptors. Variational autoencoders are designed to ensure the smoothness of latent space, and are therefore a viable alternative to AEs in this task. We shall not consider sparse autoencoders as they are overcomplete, meaning that the encoding is larger in dimensionality than the input. Encoding patches into higher-dimensional, albeit sparse, vectors is not a desired property of a descriptor and would require these encodings to be further compressed. Denoising autoencoders add artificial noise to the input and are trained to denoise it, thus also learning to encode useful properties of data samples. We emulate this behaviour using data augmentation, which is taken as a separate hyperparameter, as we discuss later in this section. Contractive autoencoders have been shown to be related to the denoising ones [39], therefore, we do not consider them separately.

A. OVERVIEW OF THE HYPERPARAMETERS

In choosing which hyperparameters to optimise and how, we need to consider two aspects: (i) how well the autoencoder learned its primary task – to encode the input into a low-dimensional representation and to recover a close replica of the input from that encoding, and (ii) how well the generated encoding performs as local image descriptor.

When considering the first objective, the main goal of hyperparameter search is to adjust the effective capacity of the neural network to match the complexity of its task at hand [40]. The way hyperparameters can influence this is by influencing the actual capacity of the network, the ability to successfully minimise the cost function, or the degree of regularisation [40]. The literature on how different hyperparameters influence these different aspects is plentiful.

The second objective is more elusive and less explored. While it is straightforward how to optimise and evaluate autoencoders' performance on reconstructing the input (which is what they are trained to do), it is less straightforward how to optimise the objective "learn the best possible descriptor". For example, data augmentation may lead to worse reconstruction performance due to sometimes outputting blurred patches, but may in fact result in a better performance as a descriptor. There exists no literature on which hyperparameters influence the performance of (V)AEs as local image descriptors. This is what we explore in this section.

Table 1 lists the hyperparameters that we address, with the concrete choices that we include in our analysis and the main empirical findings. Here we explain briefly the analysed hyperparameters and the empirical findings will be detailed in the next section.

An important parameter that we consider is the loss function used to calculate the differences between the input images and the output (reconstructed) images. The choice of loss function has been shown to strongly influence the performance of neural networks, be it an autoencoder [38], or other types of neural networks [41]. The default choice for the loss function of AEs (and the default for the reconstruction term in VAEs) is the binary cross-entropy, $\mathrm{BCE}(x, y) = -\sum_{i=1}^{N}\sum_{j=1}^{N} y_{ij}\log x_{ij} + (1-y_{ij})\log(1-x_{ij})$, where x and y are two images of size $N \times N$. Recently, a perceptual loss, multiscale structural similarity (MS-SSIM), has been shown to give significant improvements when training autoencoders [38], but has not been used before for learning descriptors. MS-SSIM is a multiscale version of the SSIM, which is defined as

$$\mathrm{SSIM}(x, y) = \frac{(2\mu_x\mu_y + c_1)(2\sigma_{xy} + c_2)}{(\mu_x^2 + \mu_y^2 + c_1)(\sigma_x^2 + \sigma_y^2 + c_2)},$$

where x and y are two windows, μ_x and μ_y are the average and σ_x^2 and σ_y^2 are the variance of x and y, respectively, σ_{xy} is the covariance between x and y, and c_1 and c_2 are the variables to stabilize the division with weak denominator. For more details on MS-SSIM, refer to [42].

The activation function, in our experience, has a significant influence on the training and overall performance of the autoencoder. The Exponential Linear Unit (ELU) and Rectified Linear Unit (ReLU) have shown to be the best-performing ones, which is why we include these two in the analysis. Exponential Linear Unit is defined as follows:

$$\mathrm{ELU}(x) = \begin{cases} x & \text{if } x \geq 0 \\ \alpha(e^x - 1)) & \text{if } x < 0 \end{cases}$$

and Rectified Linear Unit as $\mathrm{ReLU}(x) = \max(0, x)$. We do not change the activation of the last layer, where we use sigmoid function as it is common when the output is in image format [40].

Data augmentation acts as a form of regularisation in the context of learning descriptors. If we add some geometrical noise to the input, e.g. rotation by 10°, and expect the output to be the original image, then the autoencoder will learn to ignore these minor variations in the rotation. It is similar for other types of geometric noise (translation, scaling, shearing) which we want our descriptor to be (to an extent) invariant to. We empirically explore this hypothesis in Section III-C.

Beta parameter is a parameter specific to VAEs (i.e. to their generalisation, β-VAEs) and is used in the loss function to regulate the trade-off between the reconstruction term and the term that enforces the latent space distribution to be as close as possible to the prior distribution. It is a very important parameter, as it influences the trade-off between VAE learning to reconstruct well and its latent space having a distribution close to the prior distribution. The correct choice of β has been shown to induce disentanglement of facial features on the datasets of images of faces [17], while also resulting in blurred reconstructions of those images. We hypothesise that better disentanglement is related to the descriptor producing more informative encodings, and in this work we perform an analysis of the influences of β parameter on the descriptor's performance.

We do not study the optimisation of the learning rate – the studies on it are plentiful. As it controls the first objective – effective capacity of the network, and there are many algorithms that automatically optimise it (such as RMSProp, Adam, Adadelta), its tuning is beyond the scope of this work. We adopt one of the standard optimisation

TABLE 1. Summary of the analysed hyperparameters and our findings regarding their influence on the quality of local image descriptors learned with (variational) autoencoders.

Hyperparameter	Values	Empirical findings
Loss function	BCE, MS-SSIM	MS-SSIM consistently outperforms BCE on all tasks and regardless of other hyperparameters. Loss function seems to be the most important parameter.
Activation functions	ReLU, ELU	ReLU is used by the best performing descriptors. When using MS-SSIM loss function, ReLU outperforms ELU, when using BCE loss, it depends on the data augmentation level – ReLU is better with less data augmentation and ELU is better with more data augmentation.
Data augmentation level	$\{0, 1, 2, 3\}$	Best performing models overall use no data augmentation, but when breaking down into HPatches tasks, this is not always the case. For the matching task data augmentation levels 1 and 2 are the best, for retrieval 1, and for verification 0. Using data augmentation higher than 2 seems to degrade the performance of the descriptor across all tasks.
The extent to which the latent space distribution is enforced to be as close as possible to the prior distribution	Variational autoencoder with $\beta \in \{$1e-05, 1e-04, 1e-03$\}$, classical autoencoder	For all three HPatches tasks and overall, variational autoencoders with β values on the lower end (1e-05 or 1e-04) and classical autoencoders show similar (best) results. Classical autoencoders are simpler, deterministic, and more trivial to train (do not require tuning β parameter), however, for performance-sensitive applications VAEs are a marginally better option. When BCE loss is used, a VAE with a high β value will perform poorly.

algorithms (Adam) with default settings. Same holds for the weight decay coefficient.

Similarly, hyperparameters such as the number of hidden units in the layers and the number of layers in the network, have been researched elsewhere since they optimise the effective capacity of the network. Ultimately, they depend on the size of input patches. Standard regularisations such as dropout have also been explored in a general sense, they also optimise the first objective and are not specific to our problem, therefore, we do not explore them either.

We use grid search over the selected parameters – the reason for this approach (as opposed to a more heuristic-based search) is that we can get more insights into how they interact with each other when we have the results for the grid than if we had it randomly distributed.

B. THE EXPERIMENTAL SETUP

Our neural network models are built using PyTorch library for deep learning. We perform grid search for our hyperparameters using Weights & Biases Python library.

We use standard autoencoder architecture in these experiments. The encoder consists of three convolutional layers with zero padding and kernel size 3×3, each followed by a max-pooling layer. The last layer of the encoder is a fully-connected layer. The decoder mirrors the encoder, with a fully-connected layer followed by three transposed convolutional layers with stride 2 and kernel size 2×2.

The architecture of the variational autoencoder is the same as that of the classical autoencoder, except that one fully connected layer at the end of the encoder is replaced by two parallel ones (having the same input) – for the mean and variance of the Gaussian distribution. Using the mean and variance, we can sample from the Gaussian distribution in order to obtain the bottleneck layer – the encoding of our descriptor.

All the convolutional layers in the models have 32 filter maps as input and output 32 filter maps, except for the first and the last one – the first one has 1 filter map as input (because the input is a one-channel (grayscale) image), and the last one outputs 1 filter map in order to match the input of the model. The autoencoders are trained on grayscale patches so that they can be assessed with HPatches benchmark, but they can be easily extended to RGB patches by changing the number of input and output layers to 3.

For a data augmentation level a (where $a \in \{0, 1, 2, 3\}$), we perform the following transforms on the input patch to the (variational) autoencoder:

- rotation by a_r degrees, where $a_r \sim \mathcal{U}(-10a, 10a)$
- translation by a_t% of the input patch size, where $a_t \sim \mathcal{U}(-10a, 10a)$
- scaling of the input patch to a_{sc}% of the original, where $a_{sc} \sim \mathcal{U}(100 - 10a, 100 + 10a)$
- shearing transform where the new y axis forms an angle of a_{sc} to the original $y axis$, where $a_{sc} \sim \mathcal{U}(-10a, 10a)$

We use Adam optimiser for all neural networks in this paper. The networks are trained on a dataset of 125k 65×65 patches that were extracted from the images from ImageNet [43], KonIQ [44] and Visual Genome [45] datasets. The patches were extracted using FAST (Features from Accelerated Segment Test) algorithm for feature detection [46]. The patch size has been chosen because HPatches benchmark expects this patch size. The size of the encodings is 32. The ratio between training, validation and test set is $8 : 1 : 1$.

For the hyperparameter search we have created a full pipeline that allows training the models and their subsequent evaluation on HPatches (as well as using other metrics

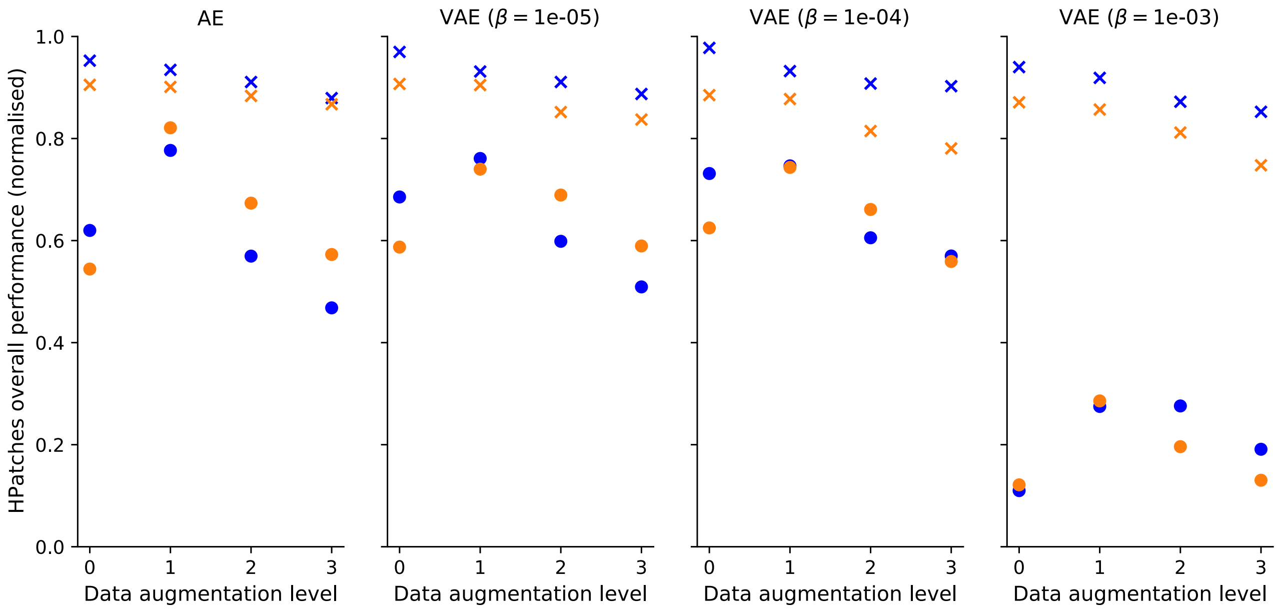

FIGURE 2. HPatches performance (normalised), showing the effect of different hyperparameter choices. β value differs across the graphs, increasing from left to right, in the left-most graph showing classical AE (essentially a deterministic VAE with $\beta = 0$), and the other three graphs VAEs with β values 1e-05, 1e-04, 1e-03, respectively. Data augmentation level is shown on x-axis of the graphs. The choice of activation function is indicated in different colours (ReLU, ELU) and the loss function with different markers (× MS-SSIM, ● BCE).

between the input and the output of the test set images). It was, therefore, enough to start one script in order to execute the whole hyperparameter grid search and evaluation. The code for the experiments in this paper is open-source and can be found in our GitHub repository.

C. EMPIRICAL RESULTS FOR HYPERPARAMETER SELECTION

We now evaluate the performance of descriptors learned with (variational) autoencoders, analysing jointly the influence of the selected hyperparameters. For each set of hyperparameters, we evaluated the learned descriptor on HPatches benchmark [11] for all three tasks: matching, retrieval and verification. Since the performance on each of these tasks can yield different conclusions regarding the optimal values of the hyperparameters, we also create one general metric. This 'overall' performance metric is the average of the normalised (between 0 and 1) outputs from the three provided tasks.

The results can be found in Figure 2 (overall performance) and Figure 3 (performance on the three individual tasks). These results allow us to draw important conclusions about the choice of hyperparameters and the choice between AEs and VAEs. Table 1 presents the summary of the findings, which we discuss in more detail in the following paragraphs.

Figures 2 and 3 show that the perceptual loss (MS-SSIM loss) yields better results than non-perceptual loss (BCE) on all tasks. This can be explained by the fact that BCE is a pixel-wise loss function which implies (1) that it does not consider the relations between different pixels in the patch, leading to an encoding that is unable to capture spatial structures and (2) that it weighs all pixels equally, even though some groups of pixels may be more discriminative. MS-SSIM loss is a perception-based loss that alleviates these issues by considering the differences in structural information of an image patch (i.e. the inter-dependencies between pixels), as well as the perceptual aspects: luminance and contrast. It is important to note that the increased performance of descriptors trained using MS-SSIM loss comes at the cost of increased training time of an autoencoder. We accept this trade-off since the incurred cost only impacts the training time – the inference time (i.e. time it takes to calculate the descriptor) is not affected by the loss function.

We highlight this impact of using perceptual loss on descriptors' performance in Figures 4 and 5, where we show a comparison of perceptual loss (MS-SSIM) with non-perceptual loss (BCE), both when using a classical autoencoder
(Figure 4) and a variational autoencoder (Figure 5). In both cases, we use ReLU activation functions. The two loss functions compare similarly for the ELU activation function and for other values of β, both when looking at the overall HPatches performance and performance by tasks, as can be seen in Figures 2 and 3, respectively.

Further on, the MS-SSIM loss function together with the ReLU activation yields the best descriptors in terms of performance on all tasks and for both autoencoders and variational autoencoders, and, in case of VAEs, this holds for different values of β parameter. Interestingly, this performance is best with little to no data augmentation.

Comparison between VAEs and AEs leads to interesting insights. For all the three tasks as well as overall, the best performing descriptors are always trained using variational autoencoders. However, the descriptors trained with classical

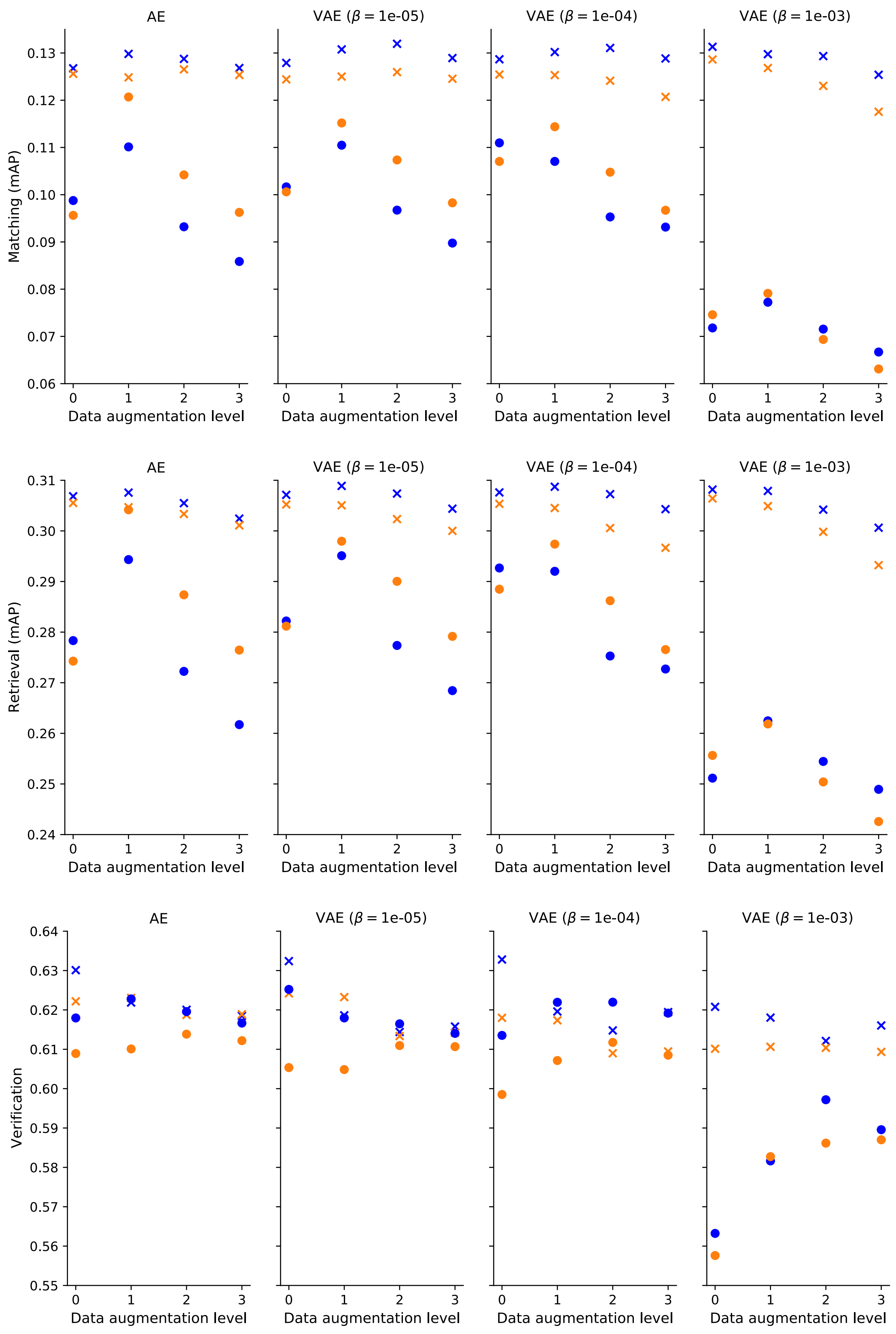

FIGURE 3. **Performance by HPatches tasks: matching (top), retrieval (middle) and verification (bottom), showing the effect of different hyperparameter choices. For each task, β value differs across the graphs, increasing from left to right, in the left-most graphs showing classical AEs (essentially a deterministic VAEs with $\beta = 0$), and the other three graphs VAEs with β values 1e-05, 1e-04, 1e-03, respectively. Data augmentation level is shown on x-axis of the graphs. The choice of activation function is indicated in different colours (ReLU, ELU) and the loss function with different markers (× MS-SSIM, ● BCE).**

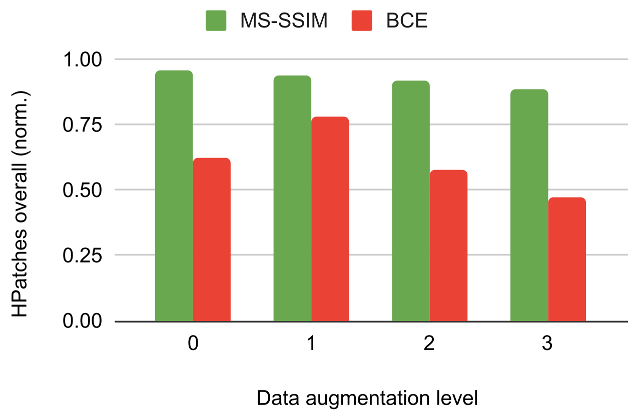

FIGURE 4. The effect of perceptual loss (MS-SSIM), versus BCE loss, on the performance of descriptors learned using an autoencoder and ReLU activation function.

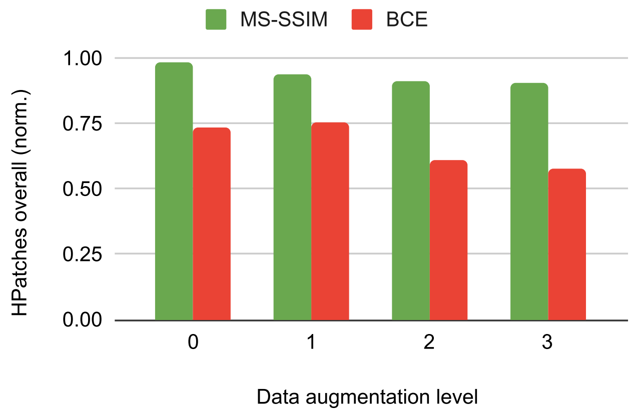

FIGURE 5. The effect of perceptual loss (MS-SSIM), versus BCE loss, on the performance of descriptors learned using a variational autoencoder (β = 1e-04) and ReLU activation function.

autoencoders are usually performing only slightly worse. Furthermore, we have also observed that the worst performing descriptors also come from variational autoencoders (when β value is too high). Therefore, rather modest improvements that VAEs offer over AEs in these tasks come at a price of additional training (tuning the β parameter), requiring additional computational resources.

Regarding the choice of the loss function, we observe that working with BCE favours small amount of data augmentation, while MS-SSIM sometimes works best with no data augmentation at all. We hypothesise that BCE needs more data augmentation in order to ''nudge'' the network into preserving similarities between patches – otherwise it may learn to reconstruct very well, but also be very sensitive to small perturbations of the patches, which is undesirable. Surprisingly, moderate to high levels of data augmentation are negatively impacting the performance, regardless of other parameters and for all the tasks.

Similarly, a too high β value (when using VAEs) is detrimental to the performance of the learned descriptors. High β values mean more weight on the KLD term of the loss function – which is essentially a form of regularisation. However, same as with the data augmentation regularisation, having some KLD regularisation is better than having none.

It is also interesting to see that when using MS-SSIM, ReLU activation function works better than ELU for all the tasks, but when using BCE, there is not a single best activation function – ReLU works better than ELU for verification task, but ELU outperforms ReLU for the matching and retrieval tasks.

Looking at different tasks (Figure 3), we found that hyperparameters influence each other in the same way for matching and retrieval tasks, i.e. descriptors' performance on them is almost perfectly correlated (0.97). Verification, too, is highly correlated with the other two tasks (with matching 0.91 and with retrieval 0.94), but the best performing hyperparameter combinations are not always the same between verification and either of the two tasks. Most notably, with matching and retrieval, MS-SSIM always outperforms BCE. However, with verification task, this is not always the case (nonetheless, the models that are performing the best on this task do use MS-SSIM, and the ones performing the worst use BCE).

We also notice that for MS-SSIM, ReLU performs consistently better than ELU, but for BCE it is not clear which activation function is better.

IV. EVALUATING LOCAL IMAGE DESCRIPTORS

In this section, we research metrics for evaluation of autoencoder-learned and variational-autoencoder–learned local image descriptors. We first give an overview of the literature on evaluating descriptors in Section IV-A, and then we present our study and results in Section IV-B.

A. HOW TO EVALUATE LEARNED DESCRIPTORS?

Evaluation of local image descriptors plays a crucial role in their design. The learning-based descriptors rely on the evaluation at different stages to make the decisions, both during the training of a neural network, and the optimisation of the hyperparameters. For a long time (up until 2017), the most widely-adopted benchmark for evaluating local image descriptors was the Oxford matching dataset [3] (from 2005), consisting of only 48 images. Other early datasets include DTU Robots dataset [47], Hanover dataset [48], Generated Matching dataset [12], WxBs dataset [49]. Most of them also contain a small amount of images, or do not support evaluation of different tasks that local image descriptors perform.

In 2017, Balntas *et al.* published now widely adopted, comprehensive HPatches dataset [11]. It enables evaluation of local image descriptors' performance on three different tasks (patch retrieval, image matching, and patch verification), each with varying difficulty levels ('easy', 'hard' and 'tough' – referring to the amount of geometric noise, as defined in [11]). The patch retrieval task tests how well a descriptor can match a query patch to a pool of patches extracted from many images, including many distractors. The image matching task tests to what extent a descriptor can

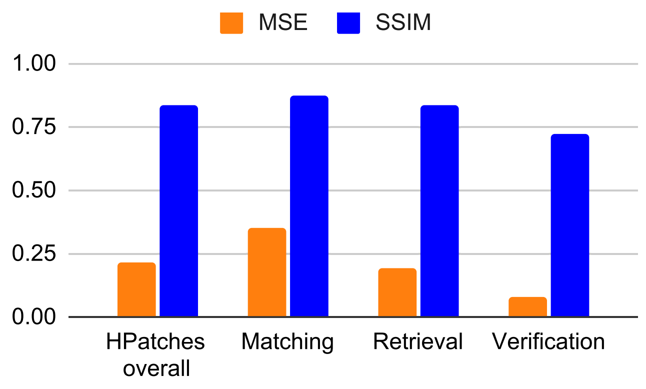

FIGURE 6. Correlation between performance of a descriptor learned using an autoencoder on HPatches benchmark, and the distance in terms of MSE and SSIM metric between the input and output patches of that autoencoder. The range of the correlation values is [−1, 1]. These results indicate that SSIM distance between input and output patches of an autoencoder is a good predictor of how well will the descriptor based on that autoencoder perform.

correctly identify correspondences in two images based on a pair of patches – one patch from each of the images. The patch verification task measures the ability of a descriptor to classify whether two patches match, i.e., whether they are extracted from the same measurement, as defined in the benchmark.

While this allows a comprehensive evaluation of a descriptor, the evaluation itself is very computationally expensive. It is, therefore, not viable to use this benchmark to quickly evaluate the learned descriptor after every training (or, better yet, every epoch in the training), in order to be able to get insights into which aspects of the network work best, and thus be able to make informed decisions about how to select hyperparameters, architecture of the neural network, etc. For supervised learning, such an evaluation can be performed using the labels provided in the dataset, but in unsupervised learning, we have no labelled data and thus we need some effective way to predict descriptor's performance. In the case of autoencoders, one can evaluate the similarity between the input and the output image patch (that is learned to be as close as possible to the input one). It is expected that evaluating this similarity by the mean squared error (MSE) cannot predict well the performance on standard patch retrieval tasks as MSE can be heavily influenced even by a small pixel shift. Thus using this metric during training is likely to result in very different encodings of structurally similar, but somewhat shifted or otherwise slightly deformed patches. We shall, therefore, explore incorporating a perceptual metric (SSIM) and analyse how it can predict the performance of local image descriptors on standard tasks.

B. EMPIRICAL RESULTS FOR EVALUATION METRICS

We now investigate and describe how to evaluate the performance of a local image descriptor learned with an autoencoder, based on how the autoencoder reconstructs patches. The goal is to find an evaluation metric that allows descriptor evaluation during the training, which is currently not possible with HPatches due to its high computational time and memory. We look at average distance using different metrics calculated between input and output patches of autoencoders, and its correlation with the HPatches performance of a descriptor learned using that autoencoder. The patches are taken from a test set (containing 12.5k patches), which the autoencoder has not seen during training.

Figure 6 shows these correlations for mean squared error (MSE) and Structural Similarity Index (SSIM) metrics. As expected, MSE does not correlate well with the overall performance on HPatches benchmark. The SSIM metric yields rather good correlation especially on matching (0.87) and retrieval tasks (0.83). We conclude that using SSIM as a quick method of evaluation will give meaningful insights to how well the descriptor will perform. Such evaluation is fast enough that it can be performed not only at the end of the training of the model, but even after every epoch of the training.

The high correlation between descriptor's performance and SSIM also explains why MS-SSIM loss function shows good results in learning patch descriptors.

V. REDUCING COMPUTATIONAL MEMORY USING INTERMEDIATE REPRESENTATION

After optimising the AE-based and VAE-based architectures for learning local image descriptors, here we introduce an architectural change to improve further their efficacy in image processing tasks. We introduce a structure in the encoder part of the autoencoder that we call *intermediate representation*. The key idea is to enable extracting an encoding of a single patch within the image with minimal computation, while having a representation that is not memory intensive.

A. INTERMEDIATE REPRESENTATION (IR)

Let $I := I^{(0,:)}$ be the input image. We define the intermediate representation $\mathcal{IR}(I)$ as:

$$\mathcal{IR}(I) = I^{(L,:)}, \tag{3}$$
$$I^{(L,c)} = \mathcal{A}(\mathcal{C}_{l_L}(\mathcal{A}\ldots(\mathcal{C}_{l_1}(I^{(0,:)})))). \tag{4}$$

where L is the number of convolutional layers in the encoder $\mathcal{E}$, $I^{(l_i,c)}$ is the c-th channel of the output of the l_i-th layer, $\mathcal{A}$ is some activation function, and $\mathcal{C}_{l_i}$ is the l_i-th convolutional layer.

From the intermediate representation of an image $\mathcal{IR}(I)$, we obtain the descriptor for a patch $I_{(i+u)(j+v)}$, with $u, v \in [0, p]$, where p is the patch size, as follows

$$\mathcal{E}(I_{(i+u)(j+v)}) = \mathcal{MP}(\mathcal{IR}(I)_{(i+u)(j+v)}). \tag{5}$$

We discard all the max-pooling layers in the encoder except for the last one. Using max-pooling is usually motivated by its advantages – adding non-linearity, playing the role of dimensionality reduction, and with that reducing the number of parameters to be trained and hence the training time. However, it has been shown that (max-)pooling is not necessary for a successful neural network, and other methods have been proposed to replace it [50].

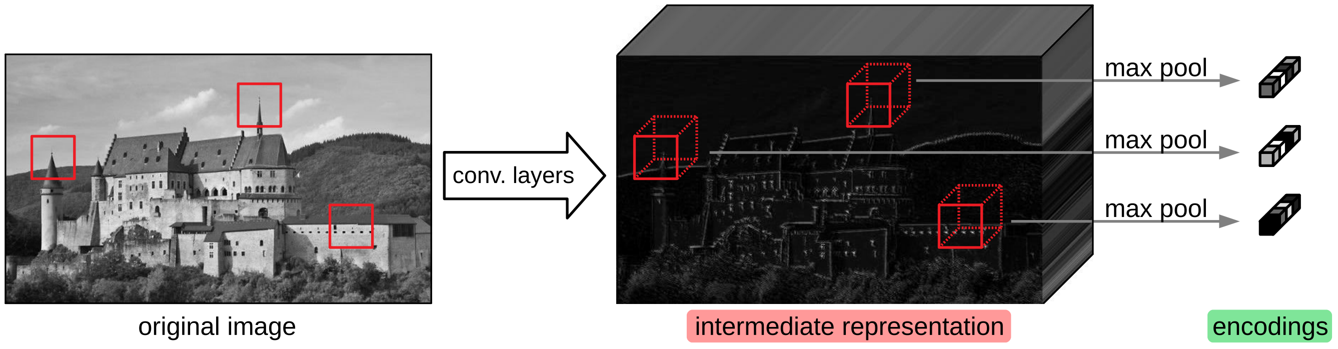

FIGURE 7. Exploiting the proposed intermediate representation (IR) of an image in algorithms that require many patch comparisons. The IR is calculated once from the original image through the convolutional layers of the encoder. In algorithms that need to compare patches (e.g. inpainting), the descriptors are extracted from the IR using the fast max-pooling operation, and then compared.

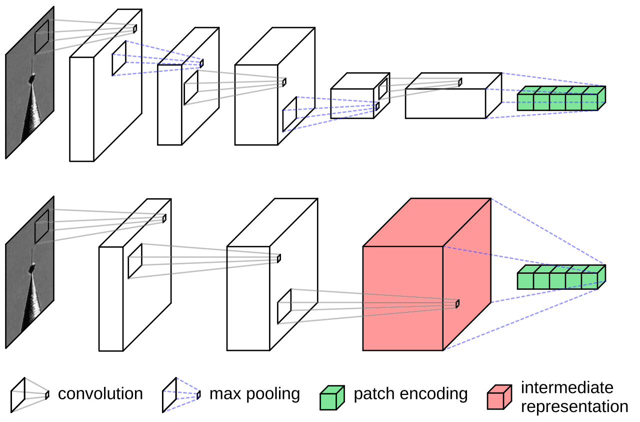

FIGURE 8. Top: traditional encoder architecture in autoencoders, with max-pooling layers after all convolutional layers. Bottom: proposed encoder architecture, which omits max-pooling layers after all the convolutional layers but the last one, in order to obtain an intermediate representation (IR) of image that preserves the spatial information in the height-width plane.

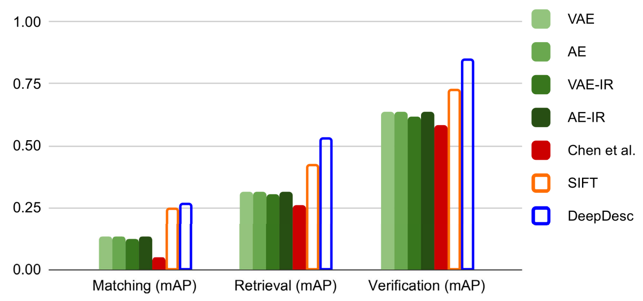

FIGURE 9. Comparison of performance on HPatches tasks (matching, retrieval and verification) between descriptors learned with VAE and AE (both non-IR and IR variants), a different autoencoder-based descriptor from Chen *et al.* [19], hand-crafted descriptor SIFT [1] and a supervised-learning–based descriptor DeepDesc [37] which achieves state-of-the-art performance [11].

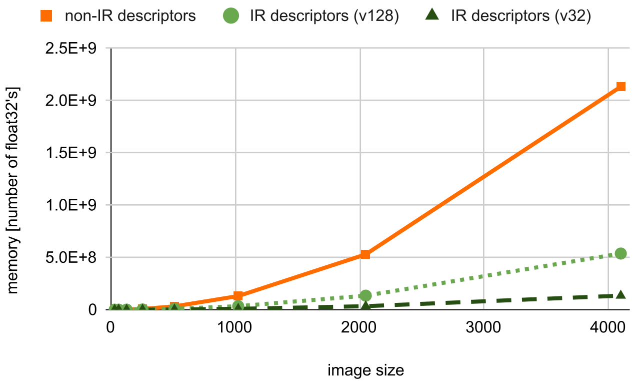

FIGURE 10. Memory needed for storing patch encodings with descriptors that do not incorporate IR architecture (Chen *et al.* descriptor, descriptors trained with regular (convolutional) AEs and VAEs) for all patches in an image compared to the memory for storing the intermediate representation of the image (showing versions where encodings are of length 32 and 128, respectively), from which a descriptor for a single patch can be obtained with minimal computation.

Indeed, non-linearity between layers is already achieved with non-linear activation functions. Dimensionality reduction is a crucial property of autoencoders and thus we do leave one max-pooling layer with large spatial extent at the end of the encoder to reduce the dimension of the code layer. The longer training time due to removing other max-pooling layers is a trade-off for decreasing the computational time and memory while using the descriptor.

The proposed approach is beneficial in image processing problems that require many patch comparisons within a single image. IR is obtained by propagating the complete image (containing patches of interest) through the convolutional layers in the encoder, but not the max-pooling. This is done only once and before the actual processing starts. During the particular image processing task, the descriptors are extracted from the stored IR using the fast max-pooling operation on the corresponding section of the IR. Figure 7 shows this process visually and Figure 8 shows the architecture of our network and the IR.

The memory reduction is achieved due to it being sufficient to store only the IR of the whole image and to get the descriptors on demand using the fast max-pooling in contrast to storing an encoding for a patch at each possible location in an image (Figure 7). It is not necessary to separately store IRs of all the patches because the IRs of overlapping patches are overlapping themselves. Conversely, this is generally not the case for the encodings of other descriptors – even for two patches shifted by one pixel in an image, their two encodings may be arbitrarily different and one would therefore need to store them both in memory. IR thus results in a tremendous

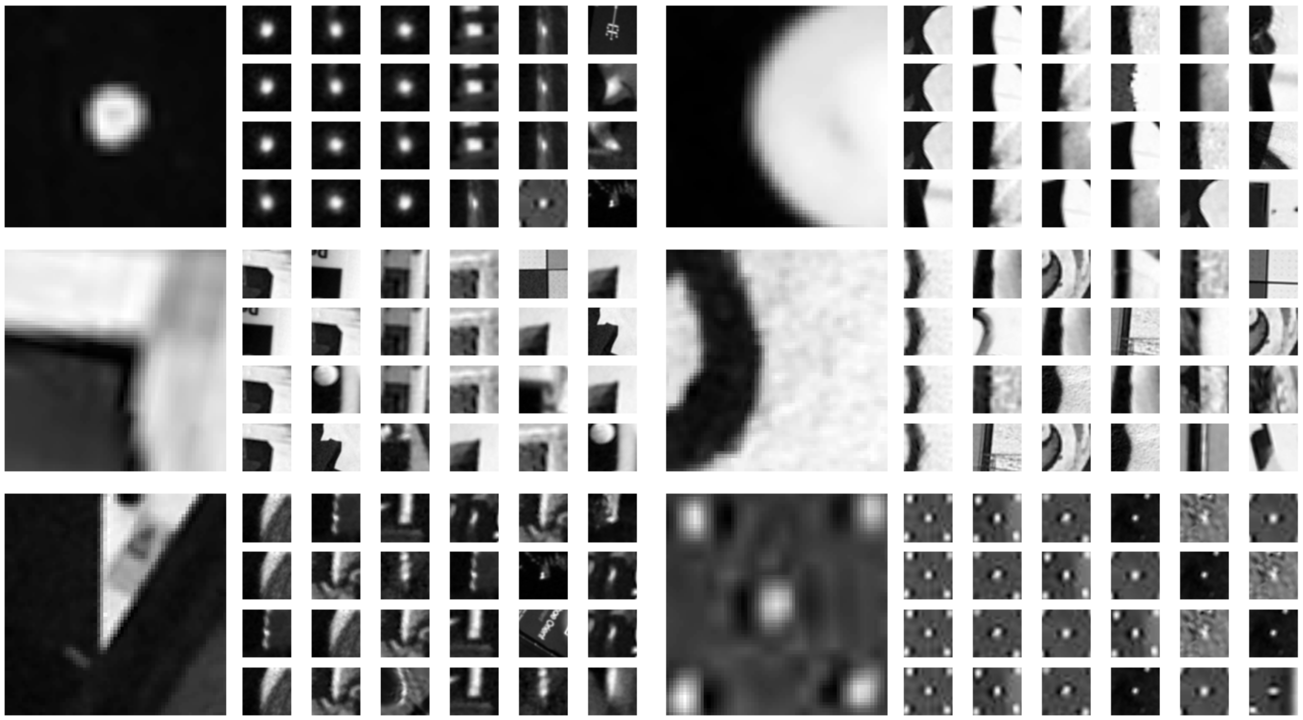
FIGURE 11. Patch retrieval examples. Large patch is the query patch. Rows (top to bottom): AE-based descriptor, VAE-based descriptor, AE-based descriptor with IR, VAE-based descriptor with IR.

decrease in memory usage for applications on single image, as shown in Figure 10. This decrease could make some algorithms that use many patch comparisons feasible for use on large images.

To summarise, the idea of intermediate representation is to allow (variational-)autoencoder-based descriptors to save memory (with respect to non-IR descriptors) while keeping their performance on the same level. The memory saving is possible due to the fact that IRs of overlapping patches are overlapping themselves, which is not the case with classical convolutional and variational autoencoders that do not implement IR architecture. In the next section, we show both the memory savings and the performance of IR descriptors that is comparable to that of non-IR descriptors.

B. EVALUATING THE PROPOSED IR ARCHITECTURE

In this section, we examine the performance of descriptors that incorporate our proposed IR method and compare their performance with their non-IR counterparts, as well as with the only other AE-based descriptor reported so far, from Chen *et al.* [19]. To put the results of these unsupervised methods into perspective, we also compare them with a state-of-the-art supervised descriptor DeepDesc [37] and with an established hand-crafted descriptor SIFT [1]. The goals of this evaluation are (1) to show comparison between AE-based and VAE-based descriptors, (2) to show comparison between IR-based and non-IR–based descriptors, (3) to show how our (V)AE-based descriptors compare to the only other AE-based descriptor (from Chen *et al.* [19]), and (4) to show how unsupervised-learning–based descriptors compare to the state-of-the-art supervised descriptors and hand-crafted descriptors. We report both performance evaluated on HPatches benchmark and performance in terms of memory requirements while using these descriptors as part of an image processing algorithm on a single image.

We have selected the best performing models based on the analysis of hyperparameters in Section III (for both models we choose ReLU activation function, no data augmentation, MS-SSIM loss function and for the VAE β value of 0.0001).

Figure 9 shows descriptors' performance evaluated on all three tasks of HPatches benchmark. We can conclude that incorporating IR architecture into a (V)AE-based descriptor does not significantly impact its performance – AE-IR version slightly outperforms AE while VAE slightly outperforms VAE-IR. All four models' performance is very similar, with VAE and AE-IR exhibiting the best performance. The similarities between these methods can be seen in Figure 11, which shows examples of the retrieved patches that are the most similar to a query patch for all our methods.

We also observe that all our four models outperform the other descriptor trained with autoencoders from [19]. However, it is clear from the Figure 9 that there is still a gap in performance between these unsupervised models on one hand and supervised and hand-crafted models on the other hand. This is to be expected given that the supervised models can leverage labels that show semantic similarities between patches without these patches being very similar in terms of their pixel values. These results also suggest that

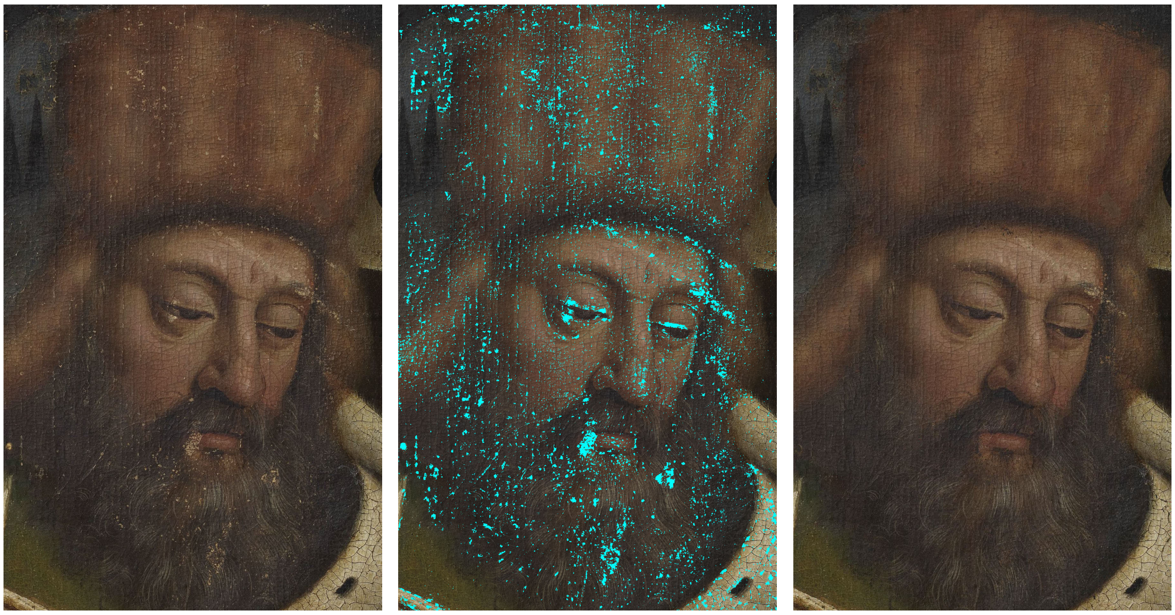

FIGURE 12. Image inpainting results. Left: original; middle: paint loss areas to be inpainted; right: inpainted with a patch-based method using the proposed local image descriptors. Image copyright: Ghent, Kathedrale Kerkfabriek, Lukasweb; photo courtesy of KIK-IRPA, Brussels.

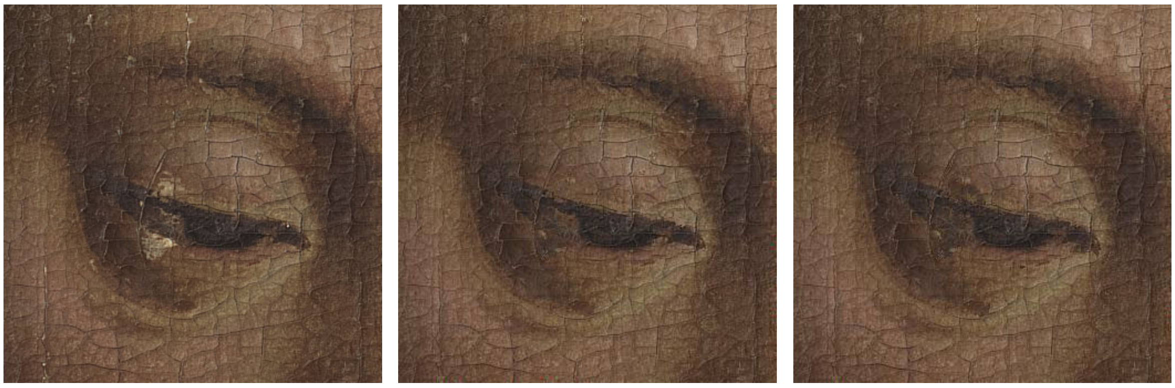

FIGURE 13. Image inpainting results. Left: original image detail containing paint-loss (showing as light brown); middle: inpainted with a patch-based method using the proposed local image descriptor; right: Inpainted without using the descriptors. Image copyright: Ghent, Kathedrale Kerkfabriek, Lukasweb; photo courtesy of KIK-IRPA, Brussels.

there is further room for improvement and innovation in the (V)AE-based descriptors in order to bridge the pefor-mance gap with the hand-crafted ones.

When it comes to the memory requirements for applications within a single image, however, descriptors that use IR (both AE-based and VAE-based) show a clear advantage over those that do not use IR, such as the descriptor from Chen *et al.* [19], SIFT [1], DeepDesc [37], and descriptors trained with regular AEs and VAEs (Figure 10). This is because our method takes advantage of the fact IRs of overlapping patches are overlapping themselves, so it is enough to simply store IR of the whole image and then extract from it descriptors using max pooling. Therefore, we can confidently say that IR architectural change is beneficial in image processing tasks on a single image because it shows a significant reduction in memory usage while keeping the descriptor's performance on the same level.

VI. INPAINTING–PROOF OF CONCEPT FOR THE PROPOSED ARCHITECTURE

We illustrate the use of the above described local image descriptors in image inpainting. The goal is to reconstruct the missing region of an image in a visually plausible way using the information from the surrounding regions of the image. Exemplar-based (sometimes called patch-based) inpainting methods fill in the missing region by sampling and copying the patches from the undamaged part of the image. These methods require many patch comparisons in order to find appropriate patches to be used to fill in the missing area. The patch size normally used is between 10 and 20 pixels depending on the image size (with [51] arguing larger patches tend to produce better results), and as the image dimensions are growing nowadays (with some of the bleeding-edge mobile phones having 100-megapixel cameras), these inpainting algorithms are becoming infeasible. This presents a window of opportunity for the local image descriptors to speed up these algorithms and make them usable in the high-definition context of the images that we have today.

Patch-based inpainting algorithms including [23], [52]–[56] search for well-matching candidate patches within some search window or within image segments with given textural and colour characteristics. We search instead well-matching patches using our learned patch descriptor and we incorporate this patch search into the inpainting algorithm from [23].

We test the resulting method in virtual restoration of master paintings. As a case study, we use images from the panels of the *Ghent Altarpiece* [24], [25]. The paint-loss areas to be inpainted are detected with the algorithm from [57].

Figure 12 shows the inpainting on a part of the panel *the Prophet Zachary*. On this particular panel, the paint-loss areas are showing as light brown. Figure 13 shows the zoomed detail. We have also used the inpainting algorithm without the descriptors, however, we were not able to obtain the inpainting results on the whole panel without using the descriptor due to the memory error on our computer.

The inpainting results are very promising and show that our descriptor was both able to improve visually the inpainted images as well as the computational aspect of the inpainting.

VII. CONCLUSION

In this paper, we performed a thorough comparison between autoencoders and variational autoencoders approach for learning local image descriptors. We show that VAE-based descriptors produce marginally better results than AE-based descriptors on HPatches benchmark, but due to other difficulties that come with VAEs (extra hyperparameters to tune being the most prominent one), classical autoencoders are most likely a better choice for learning a descriptor.

In addition, we carried out an analysis of the hyper-parameters' influence on the performance of the descriptor. We observed that autoencoders using MS-SSIM loss function, ReLU activation functions, and little to no data augmentation are producing the best descriptors in terms of performance on all HPatches tasks and for both classical and variational autoencoders.

We also investigated which metrics are the best for rapid evaluation of autoencoder-learned descriptors, and found SSIM distance between input and output patches of the autoencoder to show the highest correlation with the descriptors' performance on HPatches benchmark. We therefore propose using this metric for rapid approximate evaluation of local image descriptors learned with autoencoders.

Furthermore, we proposed an improvement to the encoder architecture which produces descriptors that perform equally as good but save memory in comparison to existing methods when used for patch search and matching within a single image. As a proof of concept, we have integrated this improved descriptor into an inpainting algorithm that resulted in visual improvements over the inpainted images and the ability to handle higher resolution images.

ACKNOWLEDGMENT

The authors would like to thank Izumi Ito and Laurens Meeus for the discussions and their valuable input in the first stages of this work.

REFERENCES

[1] D. G. Lowe, "Object recognition from local scale-invariant features," in *Proc. 7th IEEE Int. Conf. Comput. Vis.*, Sep. 1999, p. 1150.
[2] H. Bay, A. Ess, T. Tuytelaars, and L. Van Gool, "Speeded-up robust features (SURF)," *Comput. Vis. Image Understand.*, vol. 110, no. 3, pp. 346–359, Jun. 2008.
[3] K. Mikolajczyk and C. Schmid, "A performance evaluation of local descriptors," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 27, no. 10, pp. 1615–1630, Aug. 2005.
[4] Y. Ke and R. Sukthankar, "PCA-SIFT: A more distinctive representation for local image descriptors," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, vol. 2, Jun./Jul. 2004, pp. 506–513.
[5] 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.
[6] V. Balntas, E. Riba, D. Ponsa, and K. Mikolajczyk, "Learning local feature descriptors with triplets and shallow convolutional neural networks," in *Proc. Brit. Mach. Vis. Conf.*, 2016, vol. 1, no. 2, p. 3.
[7] K. He, Y. Lu, and S. Sclaroff, "Local descriptors optimized for average precision," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, Jun. 2018, pp. 596–605.
[8] Y. Ono, E. Trulls, P. Fua, and K. M. Yi, "LF-Net: Learning local features from images," in *Proc. Adv. Neural Inf. Process. Syst.*, 2018, pp. 6234–6244.
[9] Y. Tian, X. Yu, B. Fan, F. Wu, H. Heijnen, and V. Balntas, "SOSNet: Second order similarity regularization for local descriptor learning," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2019, pp. 11016–11025.
[10] Q. Wang, X. Zhou, B. Hariharan, and N. Snavely, "Learning feature descriptors using camera pose supervision," in *Proc. Eur. Conf. Comput. Vis. (ECCV)*. Cham, Switzerland: Springer, 2020, pp. 757–774.
[11] 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.
[12] P. Fischer, A. Dosovitskiy, and T. Brox, "Descriptor matching with convolutional neural networks: A comparison to SIFT," 2014, *arXiv:1405.5769*.
[13] 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. 1482–1491.
[14] G. E. Hinton and R. R. Salakhutdinov, "Reducing the dimensionality of data with neural networks," *Science*, vol. 313, no. 5786, pp. 504–507, 2006.
[15] D. P Kingma and M. Welling, "Auto-encoding variational Bayes," 2013, *arXiv:1312.6114*.
[16] A. Makhzani, J. Shlens, N. Jaitly, I. Goodfellow, and B. Frey, "Adversarial autoencoders," 2015, *arXiv:1511.05644*.
[17] I. Higgins, L. Matthey, A. Pal, C. Burgess, X. Glorot, M. Botvinick, S. Mohamed, and A. Lerchner, "β-VAE: Learning basic visual concepts with a constrained variational framework," in *Proc. 5th Int. Conf. Learn. Represent. (ICLR)*, Toulon, France, Apr. 2017, pp. 1–22.
[18] S. Pidhorskyi, D. A. Adjeroh, and G. Doretto, "Adversarial latent autoencoders," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2020, pp. 14104–14113.
[19] L. Chen, F. Rottensteiner, and C. Heipke, "Feature descriptor by convolution and pooling autoencoders," *Int. Arch. Photogramm., Remote Sens. Spatial Inf. Sci.*, vol. XL-3/W2, pp. 31–38, Mar. 2015.
[20] N. Žižakić, I. Ito, and A. Pižurica, "Learning local image descriptors with autoencoders," in *Proc. Int. Conf. Image Process. Commun.* Cham, Switzerland: Springer, 2019, pp. 214–221.
[21] N. Žižakić, I. Ito, L. Meeus, and A. Pižurica, "Autoencoder-learned local image descriptor for image inpainting," in *Proc. BNAIC/BENELEARN*, vol. 2491, 2019, pp. 1–4.
[22] N. Žižakić and A. Pižurica, "Invertible local image descriptors learned with variational autoencoders," in *Proc. IEICE Inf. Commun. Technol. Forum (ICTF)*, 2020, pp. 1–11.
[23] T. Ružić and A. Pižurica, "Context-aware patch-based image inpainting using Markov random field modeling," *IEEE Trans. Image Process.*, vol. 24, no. 1, pp. 444–456, Jan. 2015.

[24] A. Pižurica, L. Platiša, T. Ruzic, B. Cornelis, A. Dooms, M. Martens, H. Dubois, B. Devolder, M. De Mey, and I. Daubechies, "Digital image processing of the Ghent altarpiece: Supporting the painting's study and conservation treatment," *IEEE Signal Process. Mag.*, vol. 32, no. 4, pp. 112–122, Jul. 2015.
[25] R. Sizyakin, B. Cornelis, L. Meeus, H. Dubois, M. Martens, V. Voronin, and A. Pižurica, "Crack detection in paintings using convolutional neural networks," *IEEE Access*, vol. 8, pp. 74535–74552, 2020.
[26] N. Dalal and B. Triggs, "Histograms of oriented gradients for human detection," in *Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit.*, vol. 1, Jun. 2005, pp. 886–893.
[27] R. Arandjelović and A. Zisserman, "Three things everyone should know to improve object retrieval," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jun. 2012, pp. 2911–2918.
[28] E. Tola, V. Lepetit, and P. Fua, "DAISY: An efficient dense descriptor applied to wide-baseline stereo," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 32, no. 5, pp. 815–830, May 2009.
[29] M. Calonder, V. Lepetit, C. Strecha, and P. Fua, "BRIEF: Binary robust independent elementary features," in *Proc. Eur. Conf. Comput. Vis. (ECCV)*. Berlin, Germany: Springer, 2010, pp. 778–792.
[30] 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.
[31] 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.
[32] A. Alahi, R. Ortiz, and P. Vandergheynst, "FREAK: Fast retina keypoint," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jun. 2012, pp. 510–517.
[33] C. Strecha, A. M. Bronstein, M. M. Bronstein, and P. Fua, "LDAHash: Improved matching with smaller descriptors," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 34, no. 1, pp. 66–78, Jan. 2011.
[34] S. A. J. Winder and M. Brown, "Learning local image descriptors," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jun. 2007, pp. 1–8.
[35] H. Jin, Q. Liu, X. Tang, and H. Lu, "Learning local descriptors for face detection," in *Proc. IEEE Int. Conf. Multimedia Expo*, Jul. 2005, pp. 928–931.
[36] 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.
[37] 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.
[38] G. G. Pihlgren, F. Sandin, and M. Liwicki, "Improving image autoencoder embeddings with perceptual loss," in *Proc. Int. Joint Conf. Neural Netw. (IJCNN)*, Jul. 2020, pp. 1–7.
[39] G. Alain and Y. Bengio, "What regularized auto-encoders learn from the data-generating distribution," *J. Mach. Learn. Res.*, vol. 15, no. 1, pp. 3563–3593, 2014.
[40] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. Cambridge, MA, USA: MIT Press, 2016. [Online]. Available: http://www.deeplearningbook.org
[41] K. Janocha and W. M. Czarnecki, "On loss functions for deep neural networks in classification," 2017, *arXiv:1702.05659*.
[42] Z. Wang, E. P. Simoncelli, and A. C. Bovik, "Multiscale structural similarity for image quality assessment," in *Proc. 37th Asilomar Conf. Signals, Syst. Comput.*, vol. 2, Jul. 2003, pp. 1398–1402.
[43] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, "ImageNet: A large-scale hierarchical image database," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jun. 2009, pp. 248–255.
[44] V. Hosu, H. Lin, T. Sziranyi, and D. Saupe, "KonIQ-10k: An ecologically valid database for deep learning of blind image quality assessment," *IEEE Trans. Image Process.*, vol. 29, pp. 4041–4056, 2020.
[45] R. Krishna, Y. Zhu, O. Groth, J. Johnson, K. Hata, J. Kravitz, S. Chen, Y. Kalantidis, L. J. Li, D. A. Shamma, and M. S. Bernstein, "Visual genome: Connecting language and vision using crowdsourced dense image annotations," *Int. J. Comput. Vis.*, vol. 123, no. 1, pp. 32–73, 2017.
[46] E. Rosten and T. Drummond, "Machine learning for high-speed corner detection," in *Proc. Eur. Conf. Comput. Vis. (ECCV)*. Berlin, Germany: Springer, 2006, pp. 430–443.
[47] H. Aanæs, A. L. Dahl, and K. S. Pedersen, "Interesting interest points," *Int. J. Comput. Vis.*, vol. 97, no. 1, pp. 18–35, 2012.
[48] K. Cordes, B. Rosenhahn, and J. Ostermann, "High-resolution feature evaluation benchmark," in *Proc. Int. Conf. Comput. Anal. Images Patterns*. Berlin, Germany: Springer, 2013, pp. 327–334.
[49] D. Mishkin, J. Matas, M. Perdoch, and K. Lenc, "WxBS: Wide baseline stereo generalizations," 2015, *arXiv:1504.06603*.
[50] J. Tobias Springenberg, A. Dosovitskiy, T. Brox, and M. Riedmiller, "Striving for simplicity: The all convolutional net," 2014, *arXiv:1412.6806*.
[51] Y. Romano and M. Elad, "Con-patch: When a patch meets its context," *IEEE Trans. Image Process.*, vol. 25, no. 9, pp. 3967–3978, Sep. 2016.
[52] A. Criminisi, P. Pérez, and K. Toyama, "Region filling and object removal by exemplar-based image inpainting," *IEEE Trans. Image Process.*, vol. 13, no. 9, pp. 1200–1212, Sep. 2004.
[53] V. V. Voronin, V. I. Marchuk, N. V. Gapon, R. A. Sizyakin, A. I. Sherstobitov, and K. O. Egiazarian, "Exemplar-based inpainting using local binary patterns," in *Proc. 12th Image Process., Algorithms Syst.*, vol. 9019, 2014, Art. no. 901907.
[54] A. Newson, A. Almansa, Y. Gousseau, and P. Pérez, "Non-local patch-based image inpainting," *Image Process. Line*, vol. 7, pp. 373–385, Dec. 2017.
[55] M. Ghorai, S. Mandal, and B. Chanda, "Patch sparsity based image inpainting using local patch statistics and steering kernel descriptor," in *Proc. 23rd Int. Conf. Pattern Recognit. (ICPR)*, Dec. 2016, pp. 781–786.
[56] N. Zhang, H. Ji, L. Liu, and G. Wang, "Exemplar-based image inpainting using angle-aware patch matching," *EURASIP J. Image Video Process.*, vol. 2019, no. 1, p. 70, Dec. 2019.
[57] L. Meeus, S. Huang, B. Devolder, H. Dubois, M. Martens, and A. Pizurica, "Deep learning for paint loss detection with a multiscale, translation invariant network," in *Proc. 11th Int. Symp. Image Signal Process. Anal. (ISPA)*, Sep. 2019, p. 5.

NINA ŽIŽAKIĆ (Member, IEEE) received the bachelor's and master's degrees in computer science from the University of Novi Sad, Serbia, in 2014 and 2017, respectively. She is currently pursuing the Ph.D. degree in computer science and engineering with Ghent University, Belgium.

During her master's studies, she spent one year abroad as an Exchange Student at Paderborn University, Germany. She is currently a Researcher with the Group for Artificial Intelligence and Sparse Modeling (GAIM), Ghent University. Her research interests include deep learning for image processing, with applications in the biomedical domain and virtual art restoration.

ALEKSANDRA PIŽURICA (Senior Member, IEEE) received the Diploma degree in electrical engineering from the University of Novi Sad, Novi Sad, Serbia, in 1994, the M.Sc. degree in telecommunications from the University of Belgrade, Belgrade, Serbia, in 1997, and the Ph.D. degree in engineering from Ghent University, Ghent, Belgium, in 2002.

She is currently a Professor of statistical image modeling with Ghent University. Her research interests include signal and image processing and machine learning, including multiresolution statistical image models, Markov random field models, sparse coding, representation learning, and image and video reconstruction, restoration, and analysis.

Prof. Pižurica received the Scientific Prize "de Boelpaepe" for 2013–2014 awarded by the Royal Academy of Science, Letters and Fine Arts of Belgium for her contributions to statistical image modeling and applications to digital painting analysis. The work of her team has been awarded twice the Best Paper Awards of the IEEE Geoscience and Remote Sensing Society Data Fusion Contest, in 2013 and 2014. She has served as an Associate Editor for the IEEE TRANSACTIONS ON IMAGE PROCESSING, from 2012 to 2016, a Senior Area Editor for the IEEE TRANSACTIONS ON IMAGE PROCESSING, from 2016 to 2019, and an Associate Editor for the IEEE TRANSACTIONS ON CIRCUITS AND SYSTEMS FOR VIDEO TECHNOLOGY, from 2016 to 2020. She was a Lead Guest Editor of the *EURASIP Journal on Advances in Signal Processing* Special Issue "Advanced Statistical Tools for Enhanced Quality Digital Imaging with Realistic Capture Models," in 2013.