Learning Deep Descriptors with Scale-Aware Triplet Networks

Michel Keller Zetao Chen Fabiola Maffra Patrik Schmuck Margarita Chli
Vision for Robotics Lab, ETH Zurich, Switzerland
michel.keller@alumni.ethz.ch, {chenze, pschmuck, chlim}@ethz.ch, fmaffra@mavt.ethz.ch

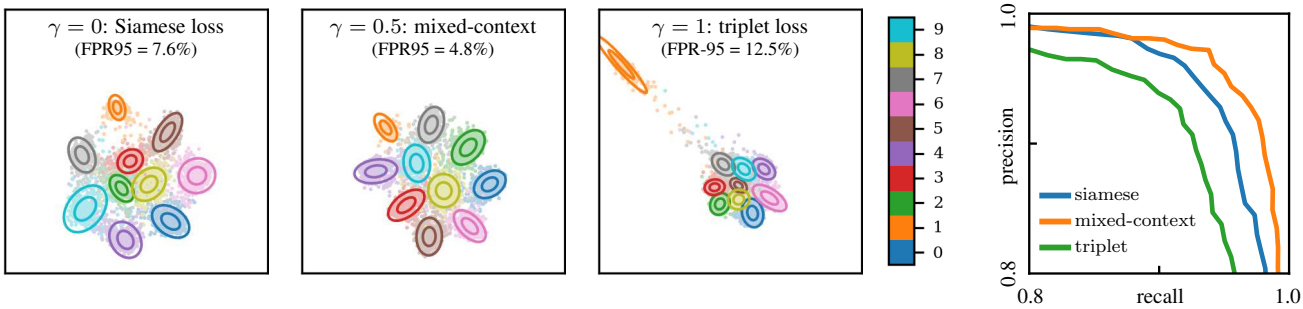

Figure 1: Two-dimensional descriptors obtained with our *mixed-context loss approach* (center) in comparison with a Siamese (left) and a triplet loss (right) evaluated on the MNIST test set. A normal distribution has been fit to each cluster and its confidence ellipses plotted. The triplet network shows clear signs of the localized-context problem (Section 3.2), resulting in inconsistently scaled descriptors. While the Siamese loss does not show this effect, it does not properly take advantage of context and therefore still learns rather poor descriptors. Our mixed-context loss (Section 4.1) in contrast shows neither problem, yielding consistently scaled descriptors and the lowest false positive rate at 95% recall (FPR95) and best PR curves.

Abstract

Research on learning suitable feature descriptors for Computer Vision has recently shifted to deep learning where the biggest challenge lies with the formulation of appropriate loss functions, especially since the descriptors to be learned are not known at training time. While approaches such as Siamese and triplet losses have been applied with success, it is still not well understood what makes a good loss function. In this spirit, this work demonstrates that many commonly used losses suffer from a range of problems. Based on this analysis, we introduce mixed-context losses and scale-aware sampling, two methods that when combined enable networks to learn consistently scaled descriptors for the first time.

1. Introduction

Local feature descriptors are a key in a plethora of tasks in Computer Vision and in fact, the quality of a feature descriptor, which is typically measured in terms of the distinctiveness and the descriptability of the image region it summarizes, is of vital importance to the performance of the task at hand. It is, therefore, not surprising that considerable effort has been undertaken to improve on hand-crafted descriptors such as SIFT [17] and SURF [5] in order to make them more robust to common image changes, such as viewpoint and illumination variations. In the last ten years, features obtained with machine learning techniques have become increasingly popular, a development that was partly driven by the introduction of high-quality labelled datasets [7, 27, 3]. While in earlier approaches [1, 7, 13, 20], learning was limited to selecting the hyper-parameters of otherwise hand-crafted descriptors, research focus has gradually shifted towards models that have more learnable degrees of freedom (e.g. [25, 26]). Driven by the recent advances in deep learning, attention has now almost completely turned to learning descriptors with neural networks [30, 4, 2, 23, 18] an approach that is entirely data-driven.

The main challenge encountered in this context is that the descriptors to be learned are not known at training time, making the formulation of loss functions difficult. This is most commonly addressed by first propagating multiple patches through a network and then comparing the obtained descriptors relative to one another. Generally, the goal is to encourage similar patches to have descriptors that are closer to each other in the descriptors' space than dissimilar ones. While this leads to losses that are simple and concise, it

leaves a lot of freedom to the networks in the way that the descriptors are organized. In this paper, we show that additional constraints are necessary if we want to guarantee that descriptors with consistent scaling are learned. Furthermore, we illustrate that it is important to align the scale of the loss and the descriptor. Our main contributions are: (1) new mixed-context losses that learn fast and yield consistently scaled descriptors, (2) a framework to scale losses which we combine with a novel way to visualize losses, and (3) a scale-aware sampling strategy that reduces the impact of badly scaled losses.

2. Related work

Descriptors such as [7], DAISY [24], and FREAK [1] are examples of early learned descriptors. These were still largely hand-crafted and machine learning was only used to find an optimal set of parameters. Gradually, research shifted towards descriptors that were explicitly designed to be used with machine learning, for example D-BRIEF [26] or Bin-Boost [25] and recently, neural networks were introduced, which can learn almost any descriptors. A major difficulty here is that the descriptors are not known a priori which renders formulating loss functions difficult.

To avoid this problem, CNNH [29] and FastHash [14] deploy a two-step procedure, where suitable descriptors (or parts of them) are inferred first and these descriptors then learned by a regressor network. An alternative approach that avoids explicitly working with descriptors at training time is taken by Deep Hashing (DH) [16] and, more recently, by DeepBit [15]. The idea here is to take advantage of the observation that the hidden layers of neural networks tend to be good feature extractors, even when not explicitly trained for this purpose.

Recent work, however, has primarily focused on learning descriptors directly and in a supervised manner, where most approaches build in one way or another on an idea first introduced by Bromley *et al*. [6] with their *Siamese network*. In a Siamese network, two patches $\boldsymbol{x}_r$ and $\boldsymbol{x}_l$ are propagated through two copies of a network, yielding floating point descriptors $\boldsymbol{y}_r$ and $\boldsymbol{y}_l$. The goal is to obtain similar descriptors for similar input patches and dissimilar descriptors otherwise, where descriptor similarity is measured in terms of the Euclidean distance $d = ||\boldsymbol{y}_r - \boldsymbol{y}_l||_2$. Formally, a Siamese loss has the following structure:

$$\mathcal{L}_{\text{Siam}}(d) = \begin{cases} \mathcal{L}_p(d) & \text{if patches similar} \\ \mathcal{L}_n(d) & \text{if patches not similar} \end{cases} \quad (1)$$

Note that this is a general formula. For example, the semi-truncated Siamese loss in [22] can be modelled with $\mathcal{L}_p(d) = d$ and $\mathcal{L}_n(d) = [\alpha - d]_+$. Several approaches that train neural networks for descriptor learning [8, 30, 22, 19] use such a loss. While Siamese losses perform well in general, they have been reported to learn relatively slowly, which can be partially attributed to a lack of contextual information [9]. *Triplet losses* build on this insight by juxtaposing a patch (called the *anchor* $\boldsymbol{x}_a$) with both a *positive example* $\boldsymbol{x}_p$ (similar to the anchor) and a *negative example* $\boldsymbol{x}_n$ (not similar to the anchor). After propagation through the network, one obtains descriptors $\boldsymbol{y}_a$, $\boldsymbol{y}_p$, and $\boldsymbol{y}_n$, respectively, from which a *positive distance* $d_p = ||\boldsymbol{y}_a - \boldsymbol{y}_p||_2$ and a *negative distance* $d_n = ||\boldsymbol{y}_a - \boldsymbol{y}_n||_2$ are computed. A triplet loss encourages d_n to become larger than d_p. Several researchers have formulated this objective in terms of the softmax function $\text{smax}(d_p, d_n) := \exp(d_p)/(\exp(d_p) + \exp(d_n))$, for example in the *log loss* and the *sum-of-squared-errors (SSE) loss*:

$$\mathcal{L}_{\text{log}} = -\log \text{smax}(-d_p, -d_n), \quad (2)$$

$$\mathcal{L}_{\text{sse}} = \text{smax}(d_p, d_n)^2. \quad (3)$$

Both of those losses were empirically evaluated by Hoffer and Ailon [9], where they observed that $\mathcal{L}_{\text{sse}}$ performs better in their specific scenario. Consecutively, $\mathcal{L}_{\text{sse}}$ has been used in PN-Net [2] and T-Feat [4] which set new standards on the UBC dataset. An alternative approach to formulating a triplet loss that has been taken is to introduce a margin α and encourage $d_p + \alpha < d_n$, giving the *(squared) subtractive hinge loss* [4, 21, 11]:

$$\mathcal{L}_{\text{sub}} = [d_p - d_n + \alpha]_+, \quad (4)$$

$$\mathcal{L}_{\text{sub2}} = [d_p^2 - d_n^2 + \alpha]_+, \quad (5)$$

where we define $[x]_+ := \max\{0, x\}$. A loss with completely different properties is the *division loss* [28, 10] that expresses the relationship between d_p and d_n as a quotient rather than a difference:

$$\mathcal{L}_{\text{div}} = \left[1 - \frac{d_n}{d_p + \epsilon}\right]_+ = \frac{1}{d_p + \epsilon}[d_p - d_n + \epsilon]_+, \quad (6)$$

where ϵ is set to a small positive value to prevent divisions by zero.

3. The importance of scale in triplet losses

In the following, we introduce the concept of performance functions with the help of which we then show that triplet losses suffer from the *localized-context problem* that sometimes leads to a distorted descriptor space. Furthermore, we illustrate that triplet losses can act highly differently depending on the scale of the descriptor space.

3.1. Performance functions

In order to facilitate our treatment of the losses, we introduce the notion of a *performance function* that assesses how well the descriptors of a triplet perform. In short, the

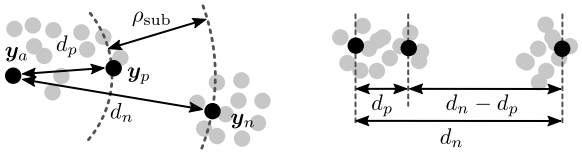

Figure 2: Illustration of $\rho_{\text{sub}} = d_n - d_p$.

smaller d_p and the larger d_n, the better the performance. Hence, we define a performance to be a function $\rho(d_p, d_n) : \mathbb{R}^2 \longrightarrow \mathbb{R}$ that satisfies $\rho(d_p + \epsilon, d_n) < \rho(d_p, d_n)$ and $\rho(d_p, d_n + \epsilon) > \rho(d_p, d_n)$ for all $\epsilon > 0$. All of the five losses that we introduced above can be rewritten in terms of one of the following three performance functions:

$$\rho_{\text{sub}}(d_p, d_n) := d_n - d_p, \tag{7}$$

$$\rho_{\text{sub2}}(d_p, d_n) := d_n^2 - d_p^2, \tag{8}$$

$$\rho_{\text{div}}(d_p, d_n) := (d_n)/(d_p + \epsilon). \tag{9}$$

In particular:

$$\mathcal{L}_{\text{log}}(\rho_{\text{sub}}) = -\log \text{smax}(\rho_{\text{sub}}, 0), \tag{10}$$

$$\mathcal{L}_{\text{sse}}(\rho_{\text{sub}}) = \text{smax}(-\rho_{\text{sub}}, 0)^2, \tag{11}$$

$$\mathcal{L}_{\text{sub}}(\rho_{\text{sub}}) = [\alpha - \rho_{\text{sub}}]_+, \tag{12}$$

$$\mathcal{L}_{\text{sub2}}(\rho_{\text{sub2}}) = [\alpha - \rho_{\text{sub2}}]_+, \tag{13}$$

$$\mathcal{L}_{\text{div}}(\rho_{\text{div}}) = [1 - \rho_{\text{div}}]_+. \tag{14}$$

Please see the additional material for a derivation of these alternative formulations. The choice of the performance function has a fundamental impact on the descriptors that are learned because they determine what we consider a well-trained model. The three performances we introduced above do this quite differently. For instance, we have $\rho_{\text{sub}}(1, 2) = \rho_{\text{sub}}(0.9, 1.9)$, but $\rho_{\text{sub2}}(1, 2) > \rho_{\text{sub2}}(0.9, 1.9)$ and $\rho_{\text{div}}(1, 2) < \rho_{\text{div}}(0.9, 1.9)$. In order to get an intuitive understanding of how a specific performance function affects the descriptors, we can picture d_p and d_n in descriptor space which is illustrated in 2D in Figure 2. In the case of feature descriptors that we are considering here, our goal can be described as clustering the descriptors of patches that stem from the same classes. Hence, the performance function affects the trade-off between getting intra-cluster distances small and inter-cluster distances large. From this point of view, the choice of the performance function is of minor importance as long as the clustering works well.

However, our goal is not primarily to cluster the training data in descriptor space, but rather to learn networks that generalize well to new examples. This implies that the *scale* of the descriptor space should be consistent in some sense with the performance function. On the right-hand side in Figure 2 we illustrate that $\rho_{\text{sub}} = d_p - d_n$ and the Euclidean distance in descriptor space conform in a more natural way

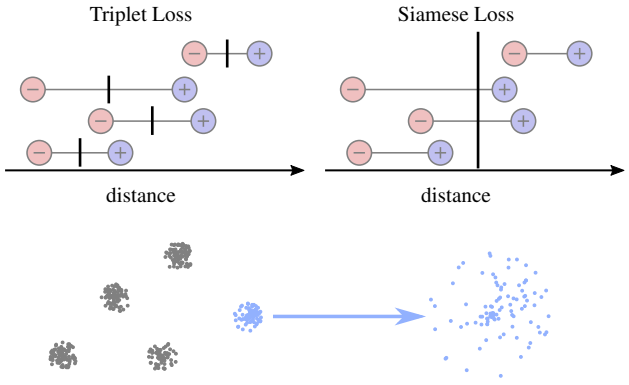

Figure 3: At the top, the conceptual threshold is shown in the case of both a triplet (left) and a Siamese (right) loss. At the bottom, the localized-context problem is illustrated in a two-dimensional descriptor space. The blue cluster is expanded (increasing d_p) and pushed away from the other clusters (increasing d_n). Therefore, the performance $\rho_{\text{sub}} = d_n - d_p$ increases slightly even though the scaling of the descriptors has deteriorated.

than ρ_{div} and ρ_{sub2}. In our experiments, we indeed observed that ρ_{sub} performs the best, although it is not clear whether this is really due to the choice of performance function or due to other design choices such as the actual formulation of the loss. Still, based on this intuition we will focus on losses in ρ_{sub} in the remainder of this paper. We will also see several times that thanks to this consistency between descriptor space and performance function, losses in ρ_{sub} are significantly easier to understand and work with.

3.2. The localized-context problem

Triplet losses are generally reported to outperform Siamese losses, sometimes by a large margin. Hoffer and Ailon [9] report that Siamese losses failed completely in their experiments while triplet losses performed just well. They suspect that this is due to the lack of context when provided only pairs of patches. To illustrate this, suppose we are given two photos of distinct people, one smiling, the other with a sad expression and we are told that the two images are not similar. The question that arises here is what aspect makes them dissimilar. Is it because the images show two different people or the different emotions? This question can simply not be answered without additional information. Were we additionally told that the image with a smiling person is similar to an image depicting a third person who is laughing, it would be clear that we are interested in clustering faces by the emotion they express. Hence, triplets add context and remove ambiguity.

It is interesting to observe how the triplets are handled by the losses. Principally, a triplet loss could treat the

positive and the negative distances as if they were from two separate and unrelated pairs in which case the loss could be formulated as $\mathcal{L}(d_p, d_n) = \mathcal{L}_{\text{Siam}}(d_p \,|\, \text{similar}) + \mathcal{L}_{\text{Siam}}(d_n \,|\, \text{not similar})$. This is a pure Siamese loss with no information exchange between the positive and the negative pairs. However, none of the five losses we have discussed above can actually be written in this form, simply because those losses do not treat positive and negative pairs in isolation. Instead, they are formulated in terms of how the positive and negative distances relate to one another, as expressed by the performance function. Siamese and triplet losses are therefore more different than it might seem at first glance.

To make this a little more intuitive, we can conceptually express the performance of a pair with respect to a distance threshold. A positive pair performs well if its distance is far smaller than that threshold. Similarly, a negative pair performs well if its distance is far greater. For a Siamese loss, this threshold would be the same for all pairs since the loss is formulated on pairs in isolation. Differently, a triplet loss is formulated based on how d_p an d_n relate to one another, and consequently the threshold is different for every single triplet. This is illustrated at the top in Figure 3.

We believe that it is this dynamic behavior that causes triplet losses to learn faster than Siamese losses as this guarantees that both the positive and the negative distances are acted on with equal strength. However, it is also clear that there is nothing that encourages triplet losses to agree on a global threshold that separates *all* positive from negative distances, not only those inside a triplet. For instance, we have $\rho_{\text{sub}}(d_p, d_n) = \rho_{\text{sub}}(d_p + \Delta, d_n + \Delta)$. This means that increasing positive and the negative distances by the same amount does not change the loss, but it has a significant impact on the scale of the clusters as illustrated in Figure 3 on the blue cluster. Note that this behavior applies to all performance functions: increasing d_p can be compensated by increasing d_n (and vice versa), potentially leading to an inconsistent scaling of the clusters that is not mirrored by the loss.

The reason for the problem we just described is a little ironic: on the one hand, triplet losses generally perform well which we attribute to a better usage of the context contained in the data *inside* of the triplet. At the same time, triplet losses neglect to exchange contextual information about their scale *amongst* each other. For this reason, we refer to this behavior as the *localized-context problem*. In Section 4.1, we address this problem by introducing *mixed-context losses* that combine the desired properties of both Siamese and triplet losses.

3.3. Scale of the losses

Most works in the literature normalize their descriptors to unit length, leading to a maximum ℓ_2-distance between two descriptors of $d_{\text{max}} = 2$. Besides, simply using a sigmoid or hyperbolic tangent activation at the output layer is quite common too, leading to much larger distances between descriptors (for example $d_{\text{max}} = 32$ with the tanh-descriptors of size 256 in PN-Net [2]). If we use the same loss for different d_{max}, we can expect very different behavior during training. If our loss is $\mathcal{L}_{\text{sub}}$ with $\alpha = 5$, the triplet condition $d_p + \alpha < d_n$ is never going to be satisfied in the case of $d_{\text{max}} = 2$, simply because the descriptors cannot be five units of distance apart. Conversely, $\alpha = 5$ seems reasonable when $d_{\text{max}} = 32$.

Another aspect that the scale affects in non-scale-invariant losses is how strongly it acts. If we increase the size of the descriptors, the derivatives of the loss might change too. For losses in ρ_{sub}, we can show using the chain rule that multiplying d_p and d_n by a factor δ also scales the derivatives by the same factor. We can compensate this by dividing the loss by δ. We want to investigate this a little further and therefore introduce generalized scale versions of $\mathcal{L}_{\text{log}}$ and $\mathcal{L}_{\text{sse}}$ (marked by a tilde):

$$\tilde{\mathcal{L}}_{\text{log}}(\rho_{\text{sub}}; \alpha, \delta) := 1/\delta \mathcal{L}_{\text{log}}(\delta(\rho_{\text{sub}} - \alpha)), \tag{15}$$

$$\tilde{\mathcal{L}}_{\text{sse}}(\rho_{\text{sub}}; \alpha, \delta) := 1/\delta \mathcal{L}_{\text{sse}}(\delta(\rho_{\text{sub}} - \alpha)). \tag{16}$$

These losses are parametrized by the scale δ (which we call the *scale correction parameter*) and, inspired by $\mathcal{L}_{\text{sub}}$, we additionally introduce a margin parameter α that manually decreases the performance and thus encourages a larger margin between d_p and d_n. It is now interesting to consider the corner cases of $\delta \to 0$ and $\delta \to \infty$. For the log loss (consult the additional material for derivations):

$$\lim_{\delta \to 0} \tilde{\mathcal{L}}_{\text{log}}(\rho_{\text{sub}}; \alpha, \delta) \propto -0.5\rho_{\text{sub}}, \tag{17}$$

$$\lim_{\delta \to \infty} \tilde{\mathcal{L}}_{\text{log}}(\rho_{\text{sub}}; \alpha, \delta) = \mathcal{L}_{\text{sub}}(\rho_{\text{sub}}; \alpha). \tag{18}$$

Hence, by changing the scale, we transitioned to completely different losses: either a linear loss that acts with equal strength on all triplets or the subtractive hinge loss that has a hard border between active and non-active regions. The generalized scale losses $\tilde{\mathcal{L}}_{\text{log}}$ and $\tilde{\mathcal{L}}_{\text{sse}}$ are the losses we will primarily use in the remainder of this text as they are formulated in terms of ρ_{sub}. We empirically show in Section 5.2 that varying δ has a significant impact on the performance of the learned descriptors.

4. Methodology

We discussed above that there are several issues with triplet losses. In this section, we introduce three tools to resolve these problems: *mixed-context losses*, an approach to choosing the right scale of a loss based on a novel visualization technique, and *scale-aware sampling*. Further, we detail our network architecture and training procedure.

4.1. Mixed-context losses

Ensuring that the learned descriptors are consistently scaled is of paramount importance for most applications that rely on descriptors. In image stitching for example, it should not happen that the matching patches are separated by a larger descriptor distance than non-matching patches. We've seen that due to the localized-context problem, triplet losses do not in general achieve this goal. Conversely, Siamese losses do not suffer from the localized-context problem, but still perform worse than triplet losses in the typical case. We attribute this to them no properly taking advantage of the context contained in triplets. In the following, we propose mixed-context losses that bring the advantages of both together: the fast learning of triplet losses and the consistent scale of Siamese losses.

As a starting point, recall the thresholds that we introduced conceptually above (compare the top images in Figure 3) and which we want to formalize now. As already mentioned, we exclusively focus on losses in the performance function $\rho_{\text{sub}} = d_n - d_p$. Notice that such losses are *antisymmetric* by which we mean that the loss acts equally strongly on d_p and d_n and pushes them in opposite directions:

$$\frac{\partial \mathcal{L}(\rho_{\text{sub}}(d_p, d_n))}{\partial d_p} = -\frac{\partial \mathcal{L}(\rho_{\text{sub}}(d_p, d_n))}{\partial d_n}. \tag{19}$$

This can be shown easily using the chain rule (see the additional material). Based on this property, it seems reasonable to place the threshold θ_{loc} halfway between d_p and d_n: $\theta_{\text{loc}} := (d_p + d_n)/2$. Note that we have the relationships $d_p = 2\theta_{\text{loc}} - d_n$ and $d_n = 2\theta_{\text{loc}} - d_p$. Consequently, we can rewrite the triplet loss as $\mathcal{L}(d_p, d_n) = \mathcal{L}(\rho_{\text{sub}}(d_p, 2\theta_{\text{loc}} - d_p)) = \mathcal{L}(\rho_{\text{sub}}(2\theta_{\text{loc}} - d_n, d_n))$. If we now define

$$\mathcal{L}_{\text{Siam}}(d \,|\, m; \theta) = \begin{cases} \frac{1}{2}\mathcal{L}(\rho_{\text{sub}}(d, 2\theta - d)) & \text{if } m = 1 \\ \frac{1}{2}\mathcal{L}(\rho_{\text{sub}}(2\theta - d, d)) & \text{if } m = 0 \end{cases} \tag{20}$$

(where m indicates whether the pair is similar) we can rewrite the triplet loss in a Siamese format that is parametrized by θ_{loc}:

$$\mathcal{L}(d_p, d_n) = \mathcal{L}_{\text{Siam}}(d_p \,|\, 1; \theta_{\text{loc}}) + \mathcal{L}_{\text{Siam}}(d_n \,|\, 0; \theta_{\text{loc}}). \tag{21}$$

This is still the original triplet loss, but written in terms of a threshold-like parameter θ_{loc}. We mentioned before that it is not possible to rewrite a pure triplet loss in a Siamese format because the d_p and d_n are not treated independently by the performance function, but relative to one another. In the form we've just proposed, we made this context explicit in the form of θ_{loc}. It is different for every triplet. For those reasons, we refer to θ_{loc} as the *triplet-local context*.

We can now immediately convert our triplet loss into a pure Siamese loss by replacing θ_{loc} by a constant *global context* θ_{glo} in all triplets. But it is more interesting to mix the global and the local context by choosing

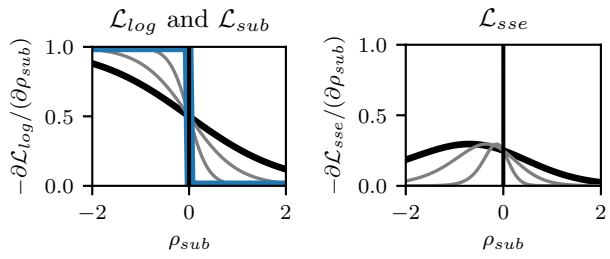

Figure 4: Activity plots of the triplet losses in ρ_{sub} discussed in the text. The log and SSE losses are plotted for $\delta = 1$ (bold), 2, and 5 and α fixed to zero. The log loss is additionally plotted for $\delta \longrightarrow \infty$ in blue which is equivalent to $\mathcal{L}_{\text{sub}}$. Generally speaking, the larger δ, the more the loss focuses training on triplets that perform badly.

$$\theta_{\text{mix}} := \gamma\theta_{\text{loc}} + (1 - \gamma)\theta_{\text{glo}}, \tag{22}$$

where $\gamma \in [0, 1]$. For $\gamma = 1$, we obtain the original triplet loss and for $\gamma = 0$, we get a pure Siamese loss. For any value of γ in between 0 and 1, we get a *mixed-context loss* that draws from the advantages of both Siamese and triplet losses. In our experiments, we will simply set $\gamma = 1/2$.

A question that remains is how we should choose the global context θ_{glo}. One can choose it manually (which we did for most of our experiments), but this is sometimes difficult to do appropriately. Instead, we can set it to a reasonable starting value and then treat it simply as a trainable parameter. We found that this often performs quite well, but it can cause the network to diverge in some cases. A more stable and equally successful approach is to estimate θ_{glo} from data: From time to time during training, we evaluate the network on a set of validation pairs and then choose the threshold that minimizes the classification error. The value of θ_{glo} is then updated in a moving average fashion.

4.2. Choosing the scale of the loss

In order to get a better intuition of the losses and their scale, we propose a novel approach to visualizing them, where we take advantage of the fact that triplet losses are formulated in terms of a performance function. We will plot the derivatives of the losses rather than their actual values as this tells us how strongly a loss acts. In short, we propose to plot the performance ρ against $-\partial\mathcal{L}/(\partial\rho)$ (also see the additional material). Larger values of $-\partial\mathcal{L}/(\partial\rho)$ indicate a stronger corrective activity of the loss, and we therefore refer to these plots as *activity plots*. We give the activity plots for $\mathcal{L}_{\text{log}}$ (which we've seen becomes $\mathcal{L}_{\text{sub}}$ for $\delta \longrightarrow \infty$ in Equation (18)) and $\mathcal{L}_{\text{sse}}$ in Figure 4.

We've already observed in Section 4.1 that losses in ρ_{sub} are antisymmetric. Thanks to this property, we know that both positive and negative distances are acted on equally strongly which makes it relatively easy to picture to oneself how such a loss acts in descriptor space. Differently, losses in ρ_{sub2} and ρ_{div} are not antisymmetric and it is therefore not possible in a straightforward way to apply the knowledge obtained with the visualization to the descriptor space. This again highlights that the performance function ρ_{sub} interacts more naturally with the Euclidean descriptor space that we use than other performance functions.

4.3. Scale-aware sampling

Taking a closer look at the activity plots in Figure 4, one can see that the losses act only weakly on triplets with a very high performance. This is an important property of triplet losses: they focus on the hard cases, allowing the network to converge towards a stable solution. Still, one cannot completely ignore triplets with high performance as the network would eventually stop learning halfway through the training process otherwise. It is evident from the activity plots that our scale correction parameter δ can thus be understood as a measure of how much more weight we give to hard cases than to easier triplets. Experiments we carried out show very clearly that networks are highly sensitive to the value δ (Section 5.2). Unfortunately, choosing an appropriate value for δ is rather difficult in general. For this reason, we train our networks with a special type of hard mining which we call *scale-aware sampling*. It automatically puts most weight on the hardest cases (in some sense, it automatically chooses the scale), and combines this with a relatively small value of δ. A general problem with hard mining that we have to be aware of is its sensitivity to outliers. We take two precautions: firstly, we mine only for hard negatives and secondly, we only mine inside of the current training batch. Given that there are hundreds of thousands of classes in the UBC dataset [7], having a false negative in a batch of typical size (e.g. 128) is vanishingly small. In an experiment where we randomly assigned wrong labels to five per cent of all patches in the UBC dataset, we observed practically no performance deterioration, showing that scale-aware sampling is reasonably invariant to outliers.

In detail, scale-aware sampling uses batches of N *positive pairs* with distances $\{d_p^i\}_{i\in 1,\dots N}$ which are all chosen from different classes. Hence, one can form negative pairs by cross-pairing descriptors, yielding a set of $2N-2$ negative distances $\{d_n^{i,j}\}_{j\in 1,\dots 2N-2}$ for each positive distance d_p^i. The overall loss of a batch is then computed as:

$$\mathcal{L}(\{d_p^i\}_i, \{d_n^{i,j}\}_{i,j}) = \sum_{i=1}^{N} \mathcal{L}_{\text{triplet}}\left(d_p^i, \min_k\{d_n^{i,k}\}\right). \quad (23)$$

At the beginning of each epoch, we generate a new training set by randomly selecting one pair from each class and then use each pair exactly once in that epoch. This ensures that no two pairs in a batch are from the same class.

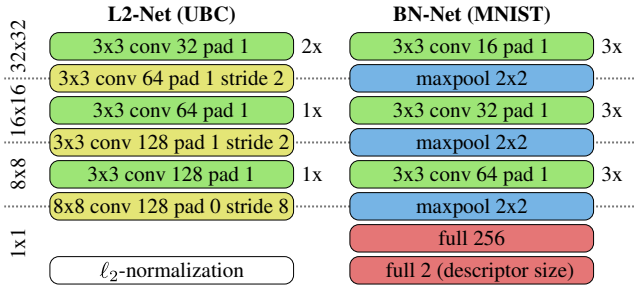

Figure 5: Networks used. The original L2-Net [23] is used for all experiments involving the UBC dataset [7], while BN-Net for the experiment on the MNIST dataset. Batch Normalization (BN) is applied after each conv or fully connected layer where the scale and offset are fixed at 1 and 0, respectively. A ReLu nonlinearity is applied after each BN-layer except the last. L2-Net produces normalized descriptors of length 128 while BN-Net produces two-dimensional descriptors that are scaled to a normal distribution by batch normalization.

4.4. Network and training procedure

For the final results in Section 5, we train Tian *et al.*'s L2-Net [23] that is detailed in Figure 5. The network is trained with standard steepest gradient descent with a momentum term of 0.9 in batches of size 128. The learning rate starts at 0.1 and is multiplied by 0.9 after every of fifty epochs. As a loss, we use the mixed-context loss of the generalized loss $\tilde{\mathcal{L}}_{\log}$ as described in Section 3.3. That is, our loss is (we choose $\alpha = 0$):

$$\theta_{\text{mix}} = \gamma(d_p + d_n)/2 + (1-\gamma)\theta_{\text{glo}}, \quad (24)$$

$$\begin{aligned}\mathcal{L}(d_p, d_n) = &-1/(2\delta)\log\operatorname{smax}(2\delta(\theta_{\text{mix}} - d_p), 0)\\ &-1/(2\delta)\log\operatorname{smax}(2\delta(d_n - \theta_{\text{mix}}), 0),\end{aligned} \quad (25)$$

where we choose $\theta_{\text{glo}} := 1.15$, a value that works well in practice. We use scale-aware sampling and set δ to 5 based on the activity plots in Figure 4, backed up by the empirical results given in Section 5.2. We choose $\gamma = 0.5$, i.e., we give equal weight to the global and the triplet-local context. We normalize the images with the average mean and standard deviation (per image, not per pixel).

5. Experiments

In the following, we first report results on two sets of experiments. The former empirically shows that we can indeed observe the limited context problem, while the latter investigates the impact of the scale correction parameter δ and scale-aware sampling. In the second part of this section, we report the performance results of our training procedure against other state-of-the-art algorithms on the UBC

benchmark [7] and the recently introduced HPatches benchmark [3].

5.1. Visualization of the localized-context problem

In our first experiment, we investigate whether the localized-context problem can indeed be observed with real data and whether mixing the context improves the learned descriptors. To this end, we train a network on the MNIST [12] dataset of handwritten digits. Since it is a particularly easy dataset and has only ten classes, it is possible to learn descriptors of dimensionality only two. The distribution of the descriptors can then be illustrated on paper without further dimensionality reduction. As a network, we use our own BN-Net detailed in Figure 5. Its descriptors are scaled to a normal distribution by a batch normalization layer at the output (hence the name). No other normalization or nonlinearity is applied at the output layer (for comparison, L2-Net projects all descriptors on a hypersphere), allowing the network to use the full 2D descriptor space. The 28×28 input patches are white-padded to fit the input size of the network (32×32). The network is trained with steepest gradient descent for 1,000 iterations with batches of 128 randomly sampled triplets. The learning rate is exponentially decreased from 0.1 to 0.001. As a loss function, we used $\mathcal{L}_{\text{sub}}$ with $\alpha = 0.1$ where we train a network with a pure Siamese loss ($\gamma = 0$), a second network with a mixed-context loss ($\gamma = 0.5$) and a third network with a triplet loss ($\gamma = 1$) where γ is kept fixed during training in all three cases. As a numerical error measure, we use the false positive rate at 95% recall (FPR95), a measure commonly used in the literature. The lower the FPR95, the better the performance it represents.

Results are given in Figure 1. In this particular case, the Siamese loss actually outperforms the triplet loss because the cluster representing the digit 1 is particularly badly scaled. This is in line with what we explained in Section 3.2: large positive distances are compensated by pushing the corresponding cluster away from the other clusters, thus increasing the negative distance. Overall, the descriptors are highly inconsistently spaced. The loss that performs the best by far is the newly proposed mixed-context loss, backing that it can profit from the advantages of both Siamese and triplet losses.

5.2. Scale of loss and scale-aware sampling

In our second set of experiments, we investigate the impact of the scale correction parameter δ, introduced in Section 3.3. Furthermore, we compare networks trained with random sampling and networks trained with our scale-aware sampling scheme introduced in Section 4.3. To this end, we train a large number of L2-Nets (Figure 5) on the UBC dataset [7] where we vary δ and the sampling strategy on the `liberty` scene and test on `notredame`. As

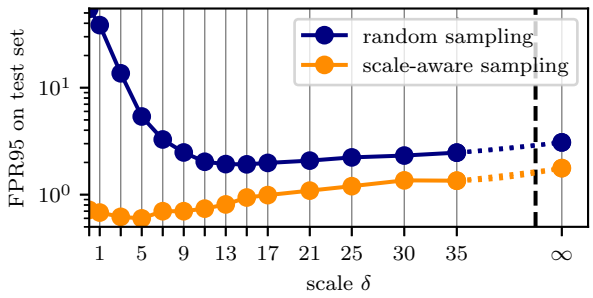

Figure 6: Comparison of random sampling and scale-aware sampling (training on `liberty`, testing on `notredame`) It can be clearly seen that scale-aware sampling performs better in all cases and is much less sensitive to a bad choice of the scale correction parameter δ.

a measure of performance we report the false positive rate at 95% recall (FPR95). The training procedure detailed in Section 4.4 is used, where we fix $\gamma = 1.0$, i.e., we use a simple triplet loss.

The corresponding results are given in Figure 6. As expected, the value of δ has a significant impact on the performance of the descriptors, which is particularly clear when using random sampling (the values plotted in blue). Conversely, the descriptors with our scale-aware sampling significantly outperform random sampling, effectively halving the FPR95 values. Apart from that, we also observe the other effect that we predicted: scale-aware sampling is less prone to bad choices of δ. If in doubt, we can simply choose a relatively small δ as the sampling strategy automatically gives more weight to hard triplets.

5.3. UBC benchmark

The first benchmark, on which we report results is the UBC benchmark [7]. According to the standard procedure described in [7], a separate network is trained on each of the three scenes `liberty`, `notredame`, and `yosemite`, and then the false positive rate at 95% recall (FPR95) is reported on the other datasets. Our results can be found in Table 1. To illustrate the impact of the measures we propose, we first train a network with a regular log-loss $\mathcal{L}_{\text{log}}$ and random sampling which gives quite poor results as the loss has a completely wrong scale for a descriptor with $d_{\text{max}} = 2$. We then train a second network with the generalized log-loss $\tilde{\mathcal{L}}_{\text{log}}$ where we set the scale-correction parameter δ to 12.5 based on the plot in Figure 6, which gives significantly better results, confirming that the scale of the loss and the descriptor should be in accordance. A comparison to T-Feat-margin* [4] is interesting here as the authors have reported some of the best results with pure triplet losses. As can be seen from the FPR95 values, our approach achieves similarly good results.

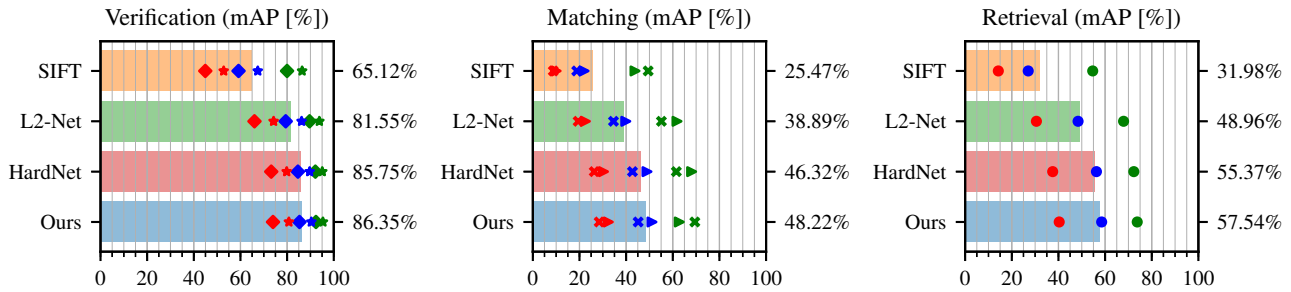

Figure 7: HPatches benchmark. The marker color indicates the level of geometrical noise: easy (green), hard (blue) and tough (red). ♦ (SAMESEQ) and ∗ (DIFFSEQ) indicate the source of negative examples in verification. ▶ (VIEWPT) and × (ILLUM) indicate the used sequence for matching. Our net here uses scale-aware sampling with $\delta = 5$ and a mixed-context loss with $\gamma = 0.5$. All descriptors (except for SIFT) were trained on the `liberty` scene from the UBC dataset.

Next, we investigate whether scale-aware sampling can further improve the performance of the descriptors. Here, we choose $\delta = 5$ rather than 12.5, again based on Figure 6 and our text above where we discussed that scale-aware sampling should use smaller values of δ than random sampling. We can see that this brings a significant boost to descriptor quality, cutting the FPR95 rates by more than half. A final little performance boost is achieved by switching to the mixed-context loss of $\tilde{\mathcal{L}}_{\log}$ with $\gamma = 0.5$. Note that the Siamese loss, whose results are also given, performs significantly worse than both mixed-context and triplet loss. In summary, the results of the UBC benchmark demonstrate that all three proposals we make in this text help improve the quality of the descriptors. In particular, we are able to report the best results on this dataset to date. As a standard measurement reported in other papers, in the appendix we also report the full PR curves and their AP on our mixed-context loss by following the procedures described in [22].

5.4. HPatches benchmark

In Figure 7, we report results on the recently proposed HPatches benchmark [3]. Here, we use the final network trained on the `liberty` dataset (with scale-aware sampling, $\gamma = 0.5$ and $\delta = 5$). This makes it comparable to both L2-Net and HardNet as these networks were trained on the same dataset. The performance measure here is the mean average prediction (mAP) where larger values are better. It can be seen that our network achieves state-of-the art performance. Note that we evaluated the networks ourselves, which guarantees that the results are comparable and explains the small difference to the results reported in [18].

6. Conclusion

Driven by the recent boom of deep learning techniques for feature descriptors in Computer Vision tasks, this paper discussed commonly used triplet losses and our analysis highlighted the importance of several aspects that should be considered when formulating a loss. Based on these insights, we proposed three tools that help in the design of a loss – namely, (1) mixed contexts, (2) a scale correction parameter δ along with a new approach to visualization that helps to choose this parameter, and (3) scale-aware sampling. Empirical evaluation revealed that all of these help in improving triplet losses, while when brought together in a unified framework, we showed that they can achieve comparable or even better state-of-the art results on the UBC and HPatches benchmarks. Overall, based on the thorough analysis presented here, the theoretical insights highlight the importance of scale in particular, and open up new research directions into how to incorporate this meaningfully for effective feature learning.

training set	NOT	YOS	LIB	YOS	LIB	NOT
test set	LIB		NOT		YOS	
SIFT [17]	29.84		22.53		27.29	
T-Feat [4]	7.22	9.53	3.12	3.83	7.82	7.08
L2-Net [23]	3.64	5.29	1.15	1.62	4.43	3.30
HardNet [18]	3.06	4.27	0.96	1.40	3.04	2.53
Random sampling, no scale correction ($\delta = 1$)						
Triplet ($\gamma = 1$)	45.0	43.2	37.2	37.1	66.0	62.6
Random sampling, with scale correction factor $\delta = 12.5$						
Triplet ($\gamma = 1$)	4.82	9.72	1.90	3.55	10.5	6.87
Scale-aware sampling, with scale correction factor $\delta = 5$						
Siamese ($\gamma = 0$)	2.61	4.70	1.36	1.62	3.89	2.91
Triplet ($\gamma = 1$)	**1.61**	3.04	**0.64**	**1.02**	3.15	2.21
Mixed ($\gamma = \frac{1}{2}$)	1.79	**2.96**	0.68	**1.02**	**2.51**	**1.64**

Table 1: UBC benchmark. The reported values are the false positive rates at 95% recall (in per cent) on the experiments discussed in the text. LIB: `liberty`, NOT: `notredame`, YOS: `yosemite`

References

[1] A. Alahi, R. Ortiz, and P. Vandergheynst. FREAK: Fast Retina Keypoint. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2012. 1, 2

[2] V. Balntas, E. Johns, L. Tang, and K. Mikolajczyk. PN-Net: Conjoined Triple Deep Network for Learning Local Image Descriptors. *arXiv preprint arXiv:1601.05030*, 2016. 1, 2, 4

[3] V. Balntas, K. Lenc, A. Vedaldi, and K. Mikolajczyk. Hpatches: A benchmark and evaluation of handcrafted and learned local descriptors. *arXiv preprint arXiv:1704.05939*, 2017. 1, 7, 8

[4] V. Balntas, E. Riba, D. Ponsa, and K. Mikolajczyk. Learning local feature descriptors with triplets and shallow convolutional neural networks. *Proceedings of the British Machine Vision Conference (BMVC)*, 2016. 1, 2, 7, 8

[5] H. Bay, A. Ess, T. Tuytelaars, and L. Van Gool. Speeded-Up Robust Features (SURF). *Computer Vision and Image Understanding (CVIU)*, 110(3):346–359, 2008. 1

[6] J. Bromley, I. G., Y. LeCun, E. Säckinger, and R. Shah. Signature Verification using a "Siamese" Time Delay Neural Network. In *Advances in Neural Information Processing Systems*, pages 737–744. 1994. 2

[7] M. A. Brown, G. Hua, and S. A. J. Winder. Discriminative Learning of Local Image Descriptors. *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, 33(1):43–57, 2011. 1, 2, 6, 7, 12

[8] X. Han, T. Leung, Y. Jia, R. Sukthankar, and A. C. Berg. MatchNet: Unifying Feature and Metric Learning for Patch-Based Matching. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015. 2

[9] E. Hoffer and N. Ailon. Deep metric learning using Triplet network. *International Workshop on Similarity-Based Pattern Recognition*, 2015. 2, 3, 10

[10] B. G. Kumar, G. Carneiro, and I. D. Reid. Learning Local Image Descriptors with Deep Siamese and Triplet Convolutional Networks by Minimising Global Loss Functions. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016. 2

[11] H. Lai, Y. Pan, Y. Liu, and S. Yan. Simultaneous Feature Learning and Hash Coding with Deep Neural Networks. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015. 2

[12] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998. 7

[13] S. Leutenegger, M. Chli, and R. Y. Siegwart. BRISK: Binary Robust Invariant Scalable Keypoints. In *Proceedings of the International Conference on Computer Vision (ICCV)*, 2011. 1

[14] G. Lin, C. Shen, and A. van den Hengel. Supervised hashing using graph cuts and boosted decision trees. *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, 2015. 2

[15] K. Lin, J. Lu, C. Chen, and J. Zhou. Learning Compact Binary Descriptors With Unsupervised Deep Neural Networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016. 2

[16] V. E. Liong, J. Lu, G. Wang, P. Moulin, and J. Zhou. Deep hashing for compact binary codes learning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015. 2

[17] D. G. Lowe. Distinctive Image Features from Scale-Invariant Keypoints. *International Journal of Computer Vision (IJCV)*, 60(2):91–110, 2004. 1, 8

[18] A. Mishchuk, D. Mishkin, F. Radenovic, and J. Matas. Working hard to know your neighbor's margins: Local descriptor learning loss. *arXiv preprint arXiv:1705.10872*, 2017. 1, 8

[19] M. Norouzi and D. J. Fleet. Minimal Loss Hashing for Compact Binary Codes. In *International Conference on Machine Learning (ICML)*, 2011. 2

[20] E. Rublee, V. Rabaud, K. Konolige, and G. Bradski. ORB: An Efficient Alternative to SIFT or SURF. In *Proceedings of the International Conference on Computer Vision (ICCV)*, 2011. 1

[21] F. Schroff, D. Kalenichenko, and J. Philbin. FaceNet: A unified embedding for face recognition and clustering. In *CVPR*, 2015. 2

[22] E. Simo-Serra, E. Trulls, L. Ferraz, I. Kokkinos, P. Fua, and F. Moreno-Noguer. Discriminative Learning of Deep Convolutional Feature Point Descriptors. *Proceedings of the International Conference on Computer Vision (ICCV)*, 2015. 2, 8, 12

[23] Y. Tian, B. Fan, and F. Wu. L2-Net: Deep Learning of Discriminative Patch Descriptor in Euclidean Space. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017. 1, 6, 8, 12

[24] E. Tola, V. Lepetit, and P. Fua. DAISY: An Efficient Dense Descriptor Applied to Wide-Baseline Stereo. *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, 32(5):815–830, 2010. 2

[25] T. Trzcinski, M. Christoudias, and V. Lepetit. Learning Image Descriptors with Boosting. *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, 37(3):597–610, 2015. 1, 2

[26] T. Trzcinski and V. Lepetit. Efficient Discriminative Projections for Compact Binary Descriptors. In *Proceedings of the European Conference on Computer Vision (ECCV)*, 2012. 1, 2

[27] S. A. J. Winder and M. Brown. Learning Local Image Descriptors. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2007. 1

[28] P. Wohlhart and V. Lepetit. Learning Descriptors for Object Recognition and 3D Pose Estimation. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015. 2

[29] R. Xia, Y. Pan, H. Lai, C. Liu, and S. Yan. Supervised Hashing for Image Retrieval via Image Representation Learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, 2014. 2

[30] S. Zagoruyko and N. Komodakis. Learning to Compare Image Patches via Convolutional Neural Networks. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015. 1, 2