Self-supervised Example Difficulty Balancing for Local Descriptor Learning

Jiahan Zhang JIAHAN.ZHANG@SE19.QMUL.AC.UK
Queen Mary University of London, UK

Dayong Tian✉ DAYONG.TIAN@NWPU.EDU.CN
Tianyang Wu TIANYANGWU@MAIL.NWPU.EDU.CN
Yiqing Cao YIQINCAO@MAIL.NWPU.EDU.CN
Yaoqi Du 2287634944@QQ.COM
School of Electronics and Information, Northwestern Polytechnical University, China.

Yiwen Wei YWWEI@XIDIAN.EDU.CN
Xidian University, China

Editors: Berrin Yanıkoğlu and Wray Buntine

Abstract

In scenarios where there is an imbalance between positive and negative examples, hard example mining strategies have been shown to improve recognition performance by assisting models in distinguishing subtle differences between positive and negative examples. However, overly strict mining strategies may introduce false negative examples, while implementing the mining strategy can disrupt the difficulty distribution of examples in the real dataset and cause overfitting on difficult examples in the model. Therefore, in this paper, we explore how to balance the difficulty of mined examples in order to obtain and exploit high-quality negative examples, and try to solve the problem in terms of both loss function and training strategy. The proposed balance loss provides an effective discriminant for the quality of negative examples by incorporating a self-supervised approach into the loss function, employing dynamic gradient modulation to achieve finer adjustment for examples of different difficulties. The proposed annealing training strategy constrains the difficulty of negative examples drawn from mining and uses examples of decreasing difficulty to mitigate the overfitting issue of hard negative examples in training. Extensive experiments demonstrate that our new sparse descriptors outperform previously established state-of-the-art sparse descriptors.

Keywords: descriptor learning, hard negative sampling, unbiased negative sampling.

1. Introduction

In many computer vision tasks, such as structure from motion (SfM), simultaneous localization and mapping (SLAM), pose estimation (De Bem et al. (2018)), and 3D-reconstruction, extracting keypoints or local features from images to evaluate local correspondences is an important problem.

To get the correspondences, there are two mainstream approaches: classical two-stage pipeline, and end-to-end pipeline. The classical two-stage pipeline consists of two steps: keypoint detection and local descriptor generation. The keypoints detection can be done by Hessian-Hessian, Difference of Gaussians (DoG), and Harris-Laplace detectors to extract

keypoints. Local descriptors can be obtained by hand-crafted or learning-based methods. End-to-end approaches have emerged in recent years, they tried to integrate detector and descriptor as a single model (DeTone et al. (2018); Revaud et al. (2019); Tyszkiewicz et al. (2020)). They perform well on some benchmarks, but for reasons of computational efficiency and modular design in practice, the traditional pipeline is still competitive in the face of realistic matching scenarios (Jin et al. (2021)), (Ma et al. (2021)), with the existing rich, replaceable components as well as the allowance of incremental improvements in independent modules (Lee et al. (2022); Cavalli et al. (2020)).

Earlier descriptors for local features were usually hand-crafted. Recently, learning-based descriptors (Mishchuk et al. (2017); Tian et al. (2020)) have been proven to be more robust than hand-crafted descriptors. Local descriptors learned with deep neural networks have been shown to improve the performance of two-stage pipelines.

Overall, the inputs for generating the sparse local descriptors are image patches and the goal of sparse local descriptors learning is to sculpt a discriminative feature space in which descriptors with high matching similarity are projected to adjacent locations. In contrast, mismatched descriptors with low similarity are separated from each other. This allows us to predict whether a patch pair matches based on the distance between descriptors. Due to the limited perceptual field of image patches, the differentiation between non-matching and fixed patches can be very small. For this reason, loss functions with pair-based units are usually used to improve the recognition ability of nuances in descriptor networks, such as triplet loss (Kumar BG et al. (2016)). Pair-based loss functions are better suited to discriminating examples without clear distinction boundaries. In particular, HardNet (Mishchuk et al. (2017)) introduced an online Hard Negative Sampling (HNS) strategy to construct harder negative pairs based on the use of triplet loss, which allowed the model to learn more subtle differences between negative and positive examples.

In subsequent studies, a number of papers followed HardNet's hard negative mining strategy and modified the design of the loss function based on a simple intuition - harder examples should receive more attention from the network to improve the model's discriminative ability further (Zhang and Rusinkiewicz (2019); Tian et al. (2020)). However, it is worth noting that if the modulation strategy encourages too strict HNS strategy, the difference between the sampled positive and negative pairs may be too small, and networks that have been focusing on these extreme examples may overfit the difficult examples. This effect can be more pronounced when the model has a small amount of parameters, making it difficult for small models to learn the common paradigm used to identify examples in the dataset. At the same time, we need to consider how to effectively measure the quality of the sampled negative examples, strong negative cases beyond a certain threshold may pick up false negatives, and these false negatives can negatively affect the model (Chuang et al. (2020)). It also shows the poor results of hardest negative mining (Tian et al. (2017)) when using the hardest examples in the training dataset. In summary, developing a strategy to balance the difficulty of the extracted samples to provide the network with high-quality negative examples is a problem that warrants further investigation.

We proposed Self-TNet which tried to solve the problem from two aspects - loss function and training strategy. First, we try to introduce a dynamic-gradient-modulation strategy and use a self-supervised approach to design the loss function. This adaptive loss function combines the information generated during training, forcing the network to strike a balance

between focusing on hard examples and excluding potential low-quality hard examples. Then, we changed the sampling strategy in the training which divides the training process into two phases: preliminary training using the basic HNS, and annealing training (AT) afterward. We performed AT by a progressive sampling strategy and setting a threshold value to constrain the difficulty of the examples drawn from the HNS, so as to train the model with examples of progressively decreasing difficulty. In other words, the loss function aspect mainly provides fine-grained gradient modulation for the sampled data according to different learning stages, while the training strategy provides the network with data sources of different difficulty distributions.

Finally, we evaluated the effectiveness of our model. The superiority of the descriptors obtained by Self-TNet is confirmed on standard benchmarks including patch verification, matching, and retrieval tasks (Balntas et al. (2017)), and the performance of our model are evaluated on downstream tasks by evaluating the pose estimation in IMC2021 (Jin et al. (2021)). We also demonstrate the compatibility of the improved loss function with the training strategy through ablation experiments, which works best when the two are combined. Our contributions can be summarized as follows.

- We proposed a balance loss for the characteristics of the data distribution of local descriptor learning, which uses dynamic gradient modulation to achieve more refined hard negative mining.
- We proposed a self-supervised approach for sampling unbiased processing, which provides a valid discriminant for the quality of negative examples to alleviate the adverse effects of extreme values or outliers on the gradient modulation.
- We proposed a progressive sampling strategy based on difficulty to provide the network with data of different difficulty distributions. After using this annealing training strategy, the performance of the model in real scenarios can be improved.

2. Related Works

Sparse Local Descriptor Learning. Early works on local patch descriptors focused on hand-crafted descriptor extraction algorithms, such as SIFT (Lowe (2004)). With the advent of open patch datasets extracted on SIFT keypoints (i.e., Gaussian difference or DoG) (Brown et al. (2010)), data-driven descriptor-based learning methods showed significant superiority over earlier hand-crafted methods (Tian et al. (2017); Mishchuk et al. (2017); Zhang and Xu (2018)). TFeat (Balntas et al. (2016)) introduced triplet loss and used triplet margin loss to construct triplets. L2Net (Tian et al. (2017)) introduced a CNN network architecture that has been widely adopted by subsequent works and redesigned the loss function and corresponding normalization. HardNet (Mishchuk et al. (2017)) confirmed the importance of the mining strategy by using a simple but fruitful online HNS strategy to select the hardest examples from each batch to construct a triplet. HyNet (Tian et al. (2020)) used a hybrid similarity loss that balanced the gradients from negative and positive examples, and proposed a new network architecture suitable for large-batch training. In addition to training neural networks on pre-cropped patch datasets, there are also works that exploit other cues such as geometric context or image context to generate dense descriptors, such as ContextDesc (Luo et al. (2019)) and R2D2 (Revaud et al. (2019)).

Gradient Modulation. Gradient modulation strategies are often used in metric learning to design more reasonable loss functions. Usually, in order to follow HNS strategy, the gradient of the positive pair should be modulated with an increasing function, while the gradient of the negative pair requires a decreasing function. In recent years, Circle loss (Sun et al. (2020)) unifies triple loss and softmax loss from a new perspective and achieves this purpose by circle margin. For local descriptor learning, Keller *et al.* (Keller et al. (2018)) made each triplet axially symmetric and balance the gradients of positive and negative pairs according to the axis of symmetry. Exponential loss (Wang et al. (2019)) achieves the purpose of gradient modulation in exponential form so that pairs with greater relative distance receive greater attention during the update. Also, there are some related works focusing on modulation for triplet tuples according to margins. Starting from (Balntas et al. (2016)) introducing static hard margins for local descriptor learning, Zhang and Rusinkiewicz (Zhang and Rusinkiewicz (2019)) further added cumulative distribution functions (CDF) to formulate dynamic soft margins.

Negative Sampling Strategy. In general, when the number of positive examples and candidate negative examples is not in the same order of magnitude, better results can be achieved by using a certain strategy to sample the negative ones in training. In other research areas, except for HNS strategy (Canévet and Fleuret (2015)), it has been shown that showing semi-hard (Schroff et al. (2015)) or distance-weighted examples (Wu et al. (2017)) when training the model will help the improvement of performance. In the field of descriptor learning, Balntas *et al.* (Balntas et al. (2016)) used triple edge loss and triple distance loss for random sampling of triplet tuples. The batch-hard sampling strategy of HardNet (Mishchuk et al. (2017)) demonstrates superior performance compared to preceding negative sampling techniques and has been extensively adopted in subsequent research (Tian et al. (2019, 2020)).

3. Methodology

3.1. Method Framework

Our novelties can be shown from two aspects - loss function and training strategy. The calculation process of our new loss function is shown in Figure 1. For better representation, we refer to the network being trained as the *supervised network* and the network that provides information to guide the *supervised network* in training as the supervising network. The training patches are first input into the supervised network to generate the corresponding embeddings, then based on the L_2 distance matrix calculated from embeddings the triplet tuples of batch size can be selected. The supervising network then recalculates the positive and negative distances of these triplet tuples and generates *confidence levels* to guide the training of the supervised network. Finally, the *confidence weight* produced by the supervising network will be integrated into the *balance loss* of the supervised network to form weighted balance loss $\mathcal{L}_{WBlance}$, for parameters update.

For the training strategy, we divide it into two stages: *preliminary training* and then *annealing training*. The main difference between them is that *annealing training* uses training data of decreasing difficulty, which can further improve the performance and decrease the potential overfitting.

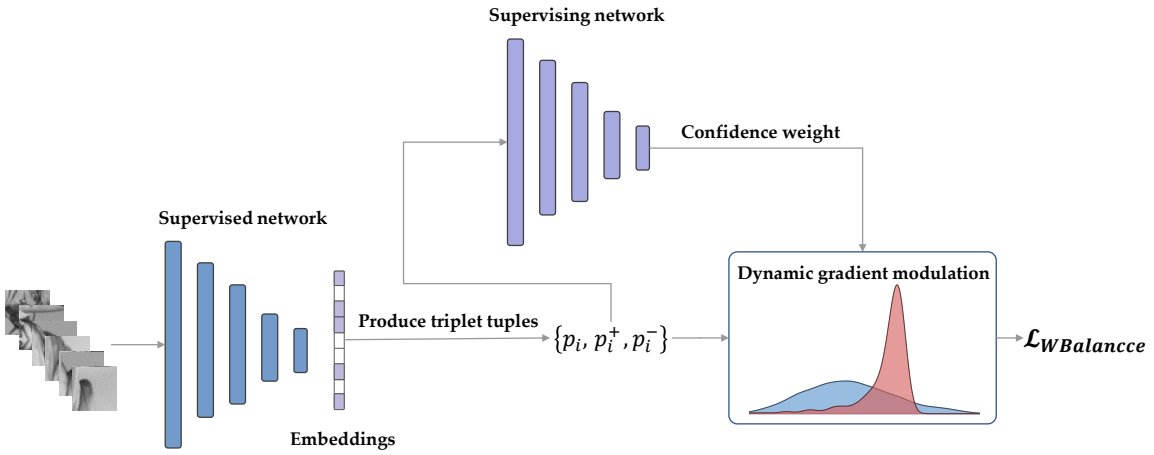

Figure 1: The calculation process of the proposed new loss function $\mathcal{L}_{WBalance}$. In our self-supervised strategy, the supervising network refers to the network being trained, which is located during the inference phase.

3.2. Loss Function

In order to better balance the difficulty of samples in real-time training of the model, our *balance loss* has integrated modifications in two main aspects - adaptive weight assignment of the gradient and unbiased processing of negative sampling through a self-supervised approach.

Compared with triplet loss, our new loss function removes the positive margin from the loss function, and the strategy of gradient modulation is changed to a strategy that is similar to constructing two potential wells for positive and negative distances. In the training process of the supervised network, it is like the process that the two potential wells of positive and negative distances are constantly and dynamically adjusted and finally reach relative equilibrium, as shown in Figure 2. Note that, in this paper we only use L_2 distance for quantifying the similarity between descriptors. The selection of triplet tuples, as well as the computation of negative and positive distances, can be described as:

$$\begin{gathered} d_i^{neg} = \min_{\forall j, j \neq i} \left(d\left(\boldsymbol{x}_i, \boldsymbol{x}_j\right), d\left(\boldsymbol{x}_i, \boldsymbol{x}_j^+\right), d\left(\boldsymbol{x}_i^+, \boldsymbol{x}_j\right), d\left(\boldsymbol{x}_i^+, \boldsymbol{x}_j^+\right), \right. \\ \left. d_i^{pos} = d\left(\boldsymbol{x}_i, \boldsymbol{x}_i^+\right), \right. \end{gathered} \tag{1}$$

where negative and positive distance denoted by d_i^{neg} and d_i^{pos} respectively and $\boldsymbol{x_i^+}$ is the corresponding patch for $\boldsymbol{x_i}$ with the same label in the dataset, where they are selected to compose a triplet tuple $\left\{\boldsymbol{p_i}, \boldsymbol{p_i^+}, \boldsymbol{p_i^-}\right\}$ based on the calculated distances.

3.2.1. Construct the Balance Loss

At this stage, our overall loss function can be expressed as:

$$\mathcal{L}_{Balance} = \frac{1}{N} \sum_{i=1}^{N} \left(s_i^{pos} + s_i^{neg}\right). \tag{2}$$

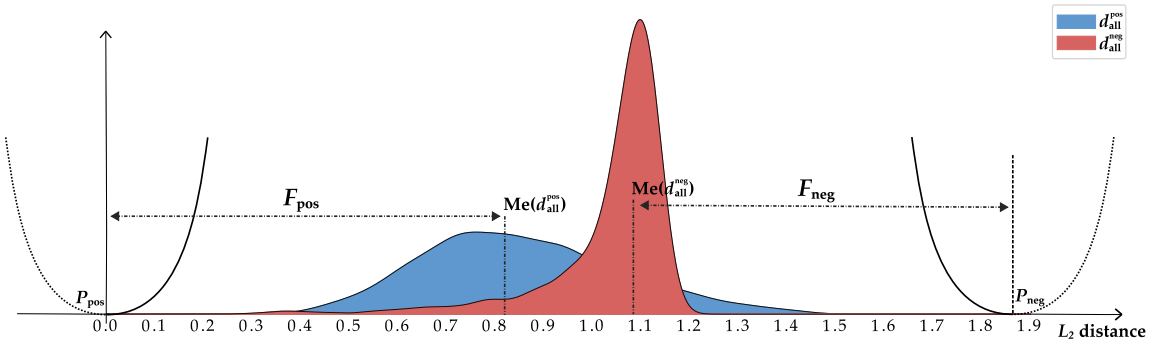

Figure 2: Schematic of the improved $\mathcal{L}_{Balance}$ and the superimposed distance distributions of d_{neg} and d_{pos}. The y-axis represents the value of the loss function or the distribution of d_{neg} and d_{pos} in a batch. The x-axis represents the L_2 distance between the examples in a pair. The two curves represent the similarity measure functions (s^{pos} and s^{neg}).

In Eq. (2), N is the the batch size and the similarity measure function $s(d)$ can be defined as two exponential functions:

$$\begin{aligned} s_i^{pos} &= \left(d_i^{pos} - P_{pos}\right)^{\alpha}, \\ s_i^{neg} &= \left(d_i^{neg} - P_{neg}\right)^{\alpha}, \end{aligned} \tag{3}$$

where α is a predefined constant and P_{neg} and P_{pos} are *zero positions* which are the keys to characterize the similarity measure functions. The zero positions correspond to the intersections of the x-axis with the zero gradients of s^{pos} and s^{neg}, as depicted in Figure 2. The use of exponential functions has the advantage of enabling adaptive weight assignment to gradients based on the calculated distances (i.e., harder examples should receive more attention during gradient updates). Detailed proof and comparison with the triplet loss can be found in the supplementary material. Considering that the selected samples of each batch vary across different training stages, zero positions are dynamically adjusted based on the sample distribution within a batch. Visualizations of the zero position adjustment and training sample distributions can also be found in the supplementary material.

Specifically, we define zero positions of the similarity measure functions as:

$$\begin{aligned} P_{pos} &= Me\left(d_{all}^{pos}\right) - F_{pos}, \\ P_{neg} &= Me\left(d_{all}^{neg}\right) + F_{neg}, \end{aligned} \tag{4}$$

where Me is a method to get the median in a batch of positive or negative distances, F_{pos} and F_{neg} are *focusing intensities* which are used to adjust how much attention is paid to the hard examples and can be represented as distance between P_{pos} and $Me\left(d_{all}^{neg}\right)$ shown in Figure 2. Based on the assumption that there is a strong correlation between positive and negative focusing intensities, we can link the two *focusing intensities* with:

$$\gamma F_{pos} = F_{neg}, \tag{5}$$

where γ is an *attention coefficient* to adjust the ratio between the two focusing intensities.

For positive distances, d_i^{pos} is a positive value representing the distance between the positive example ($\boldsymbol{p_i^+}$) and the anchor example ($\boldsymbol{p_i}$) in a triplet tuple. The model aims to minimize this distance as much as possible. Therefore, we set P_{pos} always situated at zero and only regulate P_{neg} for simplification. Substituting the expressions of F_{pos} and F_{neg} from Eq. (5) with the corresponding derived expressions from Eq. (4), we get the formula to calculate P_{neg}:

$$P_{neg} = \gamma Me\left(d_{all}^{pos}\right) + Me\left(d_{all}^{neg}\right), \tag{6}$$

where its value varies with the training process. This dynamic regulation reduces the trouble of over-parameterization and contributes to the stability of the final performance.

3.2.2. Negative Sampling Unbiased Processing through Self-supervising

The adaptive weighting in the previous subsection is designed to modulate the gradients based on information about the distribution of individual pairs with positive or negative distances, which makes the network more focused on the hard negative and positive examples. In this case, the subsequent negative sampling unbiased processing measures the relative distance between positive and negative distances in triplet pairs to mitigate the negative impact of potentially false negative examples or overly difficult negative examples. Specifically, we proposed to use a supervising network to help the model identify these outliers and reduce their weight.

We can use either the training model (i.e., the *supervising network* and the *supervised network* are identical) or a pre-trained model as the *supervising network*. The difference is that the output of this supervising model is not labels, but the confidence level of the original labels for some examples. First, we define I to represent the confidence level of the labels for positive and negative examples that are located within a dynamically selected triplet during the training process:

$$I_i = d_i^{pos} - d_i^{neg}. \tag{7}$$

Note that d_i^{pos} and d_i^{neg} are calculated by the *supervising network* for the triplet tuples which are selected by the *supervised network* through HNS. When d_i^{pos} - d_i^{neg}< $upper$, this means the anchor $\boldsymbol{p_i}$ is predicted to be more similar to the $\boldsymbol{p_i^-}$ than the $\boldsymbol{p_i^+}$ in a triplet tuple, where *upper* is a predefined hyperparameter. When the supervising network makes this judgment, we need to calculate the magnitude of the weight to be mitigated for that triplet tuple based on the value of I. Therefore, we define the *confident weight* for each triplet tuple in a batch can be calculated as:

$$W_i = \begin{cases} f(I_i), & I_i \in [upper, threshold], \\ 1, & I_i > upper, \\ 0, & I_i < threshold, \end{cases} \tag{8}$$

where W_i ranging form [0, 1], and $f(\cdot)$ is the monotonic function that maps I to the interval [0,1], which can usually be selected as an exponential function. *upper* and *threshold* are hyperparameters to determine the domain of the mapping function, as shown in Figure 3.

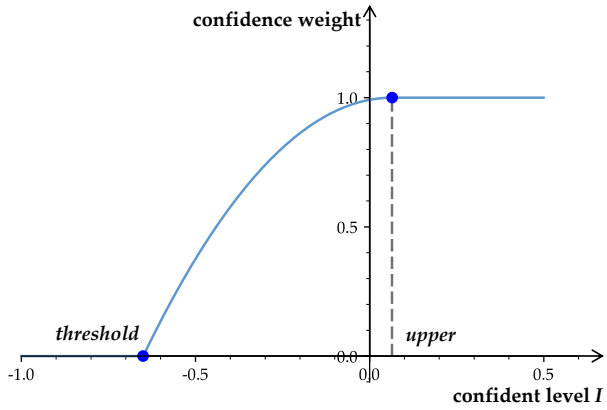

Figure 3: The blue curve represents the values of the *confident weight* (W_c) from Eq. (8) with different I values.

Finally, our weighted balance loss can be expressed as:

$$\mathcal{L}_{WBalance} = \frac{1}{N}\sum_{i=1}^{N} W_i \left(s_i^{pos} + s_i^{neg}\right). \tag{9}$$

And combine with Eq. (3) and Eq. (6) we can express it as:

$$\mathcal{L}_{WBalance} = \frac{1}{N}\sum_{i=1}^{N} W_i \left((d_i^{pos})^{\alpha} + \left(d_i^{neg} - \gamma Me\left(d_{all}^{pos}\right) - Me\left(d_{all}^{neg}\right)\right)^{\alpha}\right). \tag{10}$$

Because the supervising network only needs to process the patches selected by the supervised network and calculate the distance each time, this approach does not incur excessive computational overhead and only increases the training time by about 10% when using the training model as the supervising network.

3.3. Annealing Training

In the preliminary training, we used the strategy of Hard Negative Sampling to select triplet tuples for gradient updates. However, in the actual deployment and testing scenarios, the model needs to face a larger proportion of easy and medium difficult examples, so it is necessary to change the training strategy after the preliminary training is completed, i.e., to move from the strategy of selecting only the hardest examples in a batch to the strategy of selecting examples with decreasing difficulty to improve the data generalization ability of the model in the actual testing scenarios.

Specifically, in annealing training (AT), the weight of some triplets will be set to zero when $I_i < thr$ (I_i calculated from Eq. (7), and thr is a hyperparameter that is automatically updated during the AT process), which means that they will not participate in the parameter update of the model. This process is iterative as shown in Algorithm 1. It means bs and thr, represent the batch size and cut-off threshold, will be updated after each iteration

according to:

$$\begin{aligned} bs_t &= bs_{t-1} - stepsize_{bs}, \\ thr_t &= thr_{t-1} + stepsize_{thr}, \end{aligned} \tag{11}$$

until the set final value is reached. And the initial learning rate lr in one iteration is calculated by:

$$lr_t = lr_{t-1} * \epsilon^t, \tag{12}$$

where ϵ is the learning rate decay factor.

Algorithm 1 Annealing training algorithm

Initialization: current batch size bs, iteration started batch size bs_s, iteration ended batch size bs_e, current iteration threshold thr, step size for each threshold increase $stepsize_{thr}$, step size for each batch size decrease $stepsize_{bs}$, initial learning rate lr
$n = (bs_e - bs_s)/stepsize_{bs}$
$t = 0$
while $t < n$ **do**
  Update bs_t, thr_t based on Eq. (11)
  Update lr_t based on Eq. (12).
  Update the hyperparameters in annealing training of the current model as bs_t, lr_t, thr_t.
  Start training in this iteration:
  for a triplet i in a batch **do**
    if $I_i < thr_t$ for triplet i **then**
      The weight of triplet i will be set to zero, which means it will not participate in the process of model parameter update
    else
      Triplet i will participate normally in the process of model parameter update
    end if
  end for
  Construct overall loss by Eq. (10) in a batch
  Backpropagation and model update
  End training in this iteration
  $t = t + 1$
end while
Output: Well-trained model after annealing training

In AT, the instance of actual distribution of $W_c * d_{all}^{neg}$ is shown in Figure 4. As annealing training proceeds, the examples used for training will gradually have a larger value of confident level I, because the weight of triplets with low I is set to 0 in W_c. This is shown in Figure 4 - the distribution density of $W_c * d_{all}^{neg}$ located at zero of the x-axis becomes larger.

The key of AT is that the batch size gradually decreases after each iteration, while the cut-off threshold gradually increases. We define the value of I calculated from Eq. (7) for a triplet tuple as the basis for judging the difficulty of a sample. In this case, the increasing thr will force the samples with decreasing difficulty in each batch to be used for updating the parameters of the model after each iteration. Decreasing the batch size means that the probability of extracting a hard negative becomes smaller and the proportion of easy samples increases after each iteration.

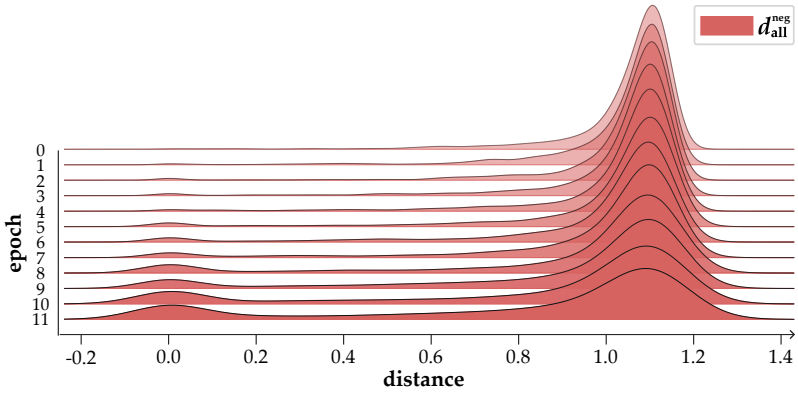

Figure 4: Density distribution of $W_c * d_{all}^{neg}$ with different epochs in AT. The visualization is performed by taking the first batch in each iteration.

4. Experiment and Results

4.1. Training Strategy

Preliminary training. We implemented our scheme with PyTorch by using the training model in the inference stage as the supervising network. This scheme was called Self-TNet. Our code can be found at https://github.com/zonszer/Self-TNet. As shown in Figure 5, for a fair comparison, the network architecture is the same as that proposed by HyNet (Tian et al. (2020)) and has the same amount of parameters as HardNet (Mishchuk et al. (2017)). In the specific implementation of balance loss, we set α=2, γ=1.05, and in the unbiased processing process, we set $upper$=0.10 and $threshold$=-0.55. The optimal model was trained with 25 epochs with 4050000 tuples per epoch and batch size equals 2560. Adam optimizer was used to update parameters with max lr=0.033, specifically using linear warm-up and cosine decay strategy for learning rate.

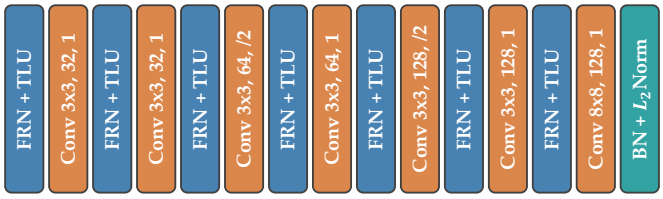

Figure 5: The network architecture is the same as that proposed by HyNet (Tian et al. (2020)) with a dropout rate of 0.3.

In Eq. (3), α is a scalar. Normally, α can be set equal to larger than 1 to represent an exponential function. We found a number larger than 2 does not necessarily lead to better results, so in the following experiments, we always set α equal to 2.

Annealing training. In AT, we tried to use triplet tuples of decreasing difficulty to update the network parameters at small learning rates. Specifically, the decrease in

difficulty was achieved by setting the batch size to decrease by $stepsize_{bs}$=128 per iteration and the threshold to increase by $stepsize_{thr}$=0.05. Other parameters included: bs_s=2944, bs_e=1024, initial thr=-0.15, number of batches per iteration=1400. We also set the initial lr=$1.5 * 10^{-6}$ with a decay factor ϵ=0.75 for each iteration.

4.2. Experimental Settings

We compared the performance of our solution with the state-of-the-art sparse descriptor networks on AMOS dataset (Pultar et al. (2019)) and three standard benchmarks: UBC (Brown et al. (2010)), HPatches (Balntas et al. (2017)) and IMC2021 (Jin et al. (2021)). Detailed ablation studies and additional analyses can be found in the supplementary material.

Train	ND	YOS	LIB	YOS	LIB	ND	**Mean**
Test	LIB		ND		YOS		
SIFT (Lowe (2004))	29.84		22.53		22.79		26.55
TFeat (Balntas et al. (2016))	7.39	10.13	3.06	3.80	8.06	7.24	6.64
L2Net (Tian et al. (2017))	2.36	4.70	0.72	1.29	2.57	1.17	2.23
HardNet (Mishchuk et al. (2017))	1.49	2.51	0.53	0.78	1.96	1.84	1.51
CDFDesc (Zhang and Rusinkiewicz (2019))	1.21	2.01	0.39	0.68	1.51	1.29	1.38
SOSNet (Tian et al. (2019))	1.08	2.12	0.34	0.67	1.03	0.95	1.03
HyNet (Tian et al. (2020))	**0.89**	1.37	**0.34**	0.61	0.88	0.96	0.84
Ours	**0.89**	**1.36**	0.37	**0.52**	**0.76**	**0.79**	**0.78**

Table 1: Patch verification performance on UBC Phototour with LIB for Liberty, YOS for Yosemite, and ND for NotreDame. Numbers shown are FPR@95 (%) with lower values being better. The most outstanding results are shown in **bold**.

UBC Benchmark evaluates on the Brown dataset (Brown et al. (2010)). It is a widely used patch dataset for evaluating the performance of local descriptors. The dataset consists of three subsets for three different scenarios: Liberty, Yosemite, and Notredame. Typically, deep descriptor networks are trained on one subset and tested on the other two subsets. According to the standard protocol (Brown et al. (2010)), the benchmark verifies that the network can correctly determine the 100K matching and non-matching pairs in the two test subsets. The protocol evaluates the false-positive rate at 95% recall (FPR@95), and we reported the experimental results including the average over all subsets in Table 1.

HPatches (Balntas et al. (2017)) is a more comprehensive benchmark that evaluates descriptors on three tasks: patch verification, image matching and patch retrieval. According to geometric distortion, subtasks are categorized into Easy, Hard and Tough. Furthermore, patch pairs from the same or different image sequences are separated into two test subsets for verification, denoted by Intra and Inter, respectively. And the matching task is designed to evaluate the viewpoint (VIEWP) and illumination (ILLUM) invariance of descriptors. According to the protocol, all models were trained on the Liberty dataset. As shown in Figure 6, our approach outperformed previous state-of-the-art methods across all three tasks. Notably, compared to HyNet which uses the same network architecture, our Self-TNet

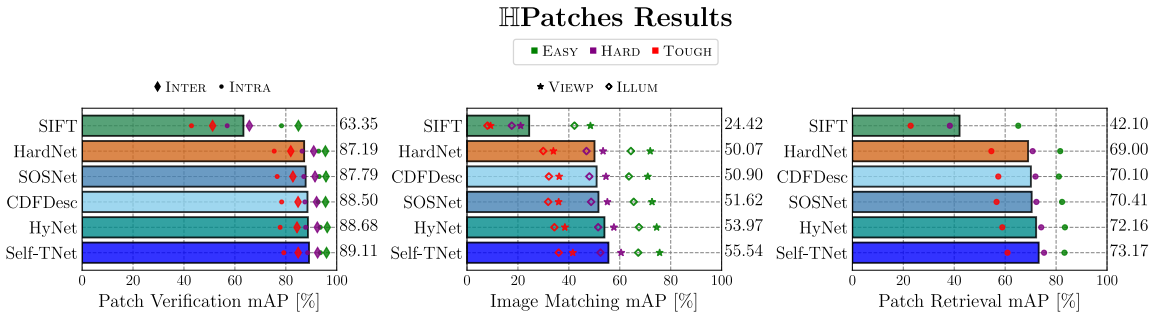

Figure 6: We tested the data split 'a' of the HPatches benchmark following the standard protocol. All networks were trained on the Liberty dataset from UBC PhotoTour. The bar chart shows the average scores of the networks for the three subtasks Patch Verification, Image Matching, and Patch Retrieval, evaluated as mean average precision (mAP).

scheme achieved a significant performance improvement of 1.57 on Image Matching task, demonstrating the effectiveness of our new loss function and training strategy.

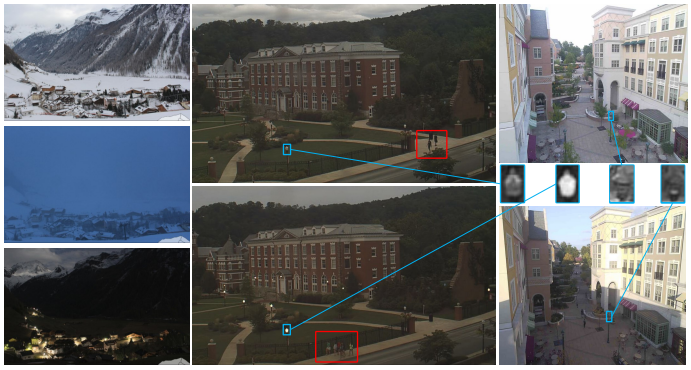

Figure 7: Challenges in AMOS dataset. The **red** boxes in the second column show examples where different images in the same sequence may contain some different dynamic contents, eg. pedestrians. The **blue** boxes in the second and third columns of images indicate similar content that may exist between different views.

AMOS Dataset (Pultar et al. (2019)) is composed of a set of images captured by outdoor cameras in the same location at different times, exhibiting strong variability in appearance and lighting conditions. The training set consists of 27 sequences, each containing 50 images captured at the same angle but in different weather and lighting conditions. However, training with this dataset presents some challenges. Due to variations in the shooting time, some images in a sequence may contain content that is not present in other images of the same scene (as shown in Figure 7). Such noisy data used for training may influence the learning of the descriptor network, potentially decreasing the performance. Furthermore, as

more scenes are added to the training set, patches from different views may contain patches similar to positive samples in the current view, which may, however, be treated as negative examples. Therefore, effectively learning from these multi-source training data with some noise poses a challenge to training descriptor networks on large-scale and expandable data.

Model Name	**Difficulty Division**			**Mean**
	Easy	Hard	Tough	
HardNet	72.46	56.57	37.89	55.64
CDFDesc	72.83	56.95	38.15	55.97
SOSNet	73.49	57.56	38.91	56.65
HyNet	74.47	58.61	40.14	57.74
Ours	**75.94**	**60.27**	**41.78**	**59.33**

Table 2: Performance (mAP) of image matching task on hpatches benchmark 'a' split, when all the models are trained on AMOS dataset.

Our Self-TNet scheme effectively alleviated these problems, because our strategy can effectively filter out these anomalous data during the training process. We retrained our descriptor network and existing state-of-the-art descriptor networks on the AMOS dataset and tested their mAP performance on the Hpatches benchmark. Table 2 reported the results. Our Self-TNet outperformed other descriptor networks observably, demonstrating its scalability on large-scale datasets and robustness to noisy training data.

#Keypoints	**2k**		**8k**		**Mean**
Method \Task	Stereo	Mutiview	Stereo	Mutiview	
DoG + HardNet	57.07	66.34	66.99	76.69	66.77
DoG + CDFDesc	57.16	66.30	67.03	76.72	66.80
DoG + SOSNet	57.03	66.76	66.78	76.96	66.88
DoG + HyNet	57.09	67.57	67.20	77.12	67.25
DoG + Ours	**58.27**	**68.39**	**67.98**	**78.10**	**68.19**

Table 3: Performance (mAA) on the validation set of IMC2021. Higher values of MAA indicate higher performance. All reported performance were matched using the ratio test (0.8 for multi-view task with 8K keypoints, 0.9 for all other tasks) with the fast library for approximate nearest neighbors (FLANN). Geometric verification was performed using the recommended settings with DEGENSAC (Chum et al. (2005)) on both stereo and multi-view tasks.

Image Matching Challenge (IMC) (Jin et al. (2021)) is a large-scale challenge dataset of wide baseline matching, which focuses on evaluating the performance of downstream tasks. In this experiment, We compared our method with the existing descriptor networks in a two-stage image matching pipeline. We utilized DoG (Lowe (2004)) as our detector and evaluated all the descriptor networks trained on the Liberty dataset using the validation sets of Phototourism. This benchmark takes the predicted matches as input and measures the 6-DoF pose estimation accuracy. The performance of the networks

was reported In Table 3, the mean Average Accuracy at a 10° angular error (mAA@10) is presented for both stereo and multi-view tasks. Our Self-TNet scheme achieved best performance in both tasks using the same two-stage image matching pipeline with 2k and 8k keypoints. Furthermore, we combined our method with the REKD detector (Lee et al. (2022)) and AdaLAM (Cavalli et al. (2020)) matcher to compare against several popular end-to-end image matching methods, as shown in Table 4. These results highlight the practical superiority and applicability of our approach in real-world image matching pipelines.

	#Keypoints	**2k**		**8k**		**Mean**
Pipeline	**Method \Task**	Stereo	Mutiview	Stereo	Mutiview	
End-to-end	SuperPoint (DeTone et al. (2018))	40.55	55.46	–	–	–
	R2D2 (MS) (Revaud et al. (2019))	46.67	62.47	59.86	70.42	59.85
	ASLFeat (MS) (Luo et al. (2020))	42.96	54.69	51.35	63.48	53.10
	DISK (Tyszkiewicz et al. (2020))	**64.53**	**74.04**	**70.37**	**81.72**	**72.67**
Two-stage	REKD (Lee et al. (2022)) + HyNet (Tian et al. (2020))	64.15	74.83	74.24	84.38	74.40
	REKD (Lee et al. (2022)) + Ours	**65.04**	**75.69**	**75.38**	**85.20**	**75.33**

Table 4: Performance comparison (mAA) of end-to-end image matching methods and two-stage pipelines on the validation set of IMC2021. We employed the optimal settings as described in the respective papers for each end-to-end method

5. Conclusion

In this paper, we investigated the acquisition and utilization of high-quality negative examples in the widely used hard negative sampling (HNS) strategy, with the goal of balancing the difficulty of the HNS extraction samples to improve the performance of the current model. The proposed balance loss integrates a self-supervised approach within a dynamic gradient modulation technique to achieve fine-grained gradient modulation for examples of varying difficulty. Additionally, the proposed annealing training provides data sources with different difficulty distributions for the loss function, which can reduce overfitting of hard examples and facilitate learning of the generalized rules. Our improved local descriptors demonstrate superiority on a variety of tasks and datasets.

6. Acknowledgments

We thank our colleagues for sharing their expert knowledge and open-source code. Our work was partly funded by the National Science Foundation of China under Grants 61806166 and 61801362, and the Natural Science Foundation of Shanxi Province under Grant 2020JQ-197, support for which we are deeply grateful. We also value the invaluable insights from our reviewers, whose guidance has significantly shaped our work.

References

Vassileios Balntas, Edgar Riba, Daniel Ponsa, and Krystian Mikolajczyk. Learning local feature descriptors with triplets and shallow convolutional neural networks. In *BMVC*, volume 1, page 3, 2016.

Vassileios Balntas, Karel Lenc, Andrea Vedaldi, and Krystian Mikolajczyk. Hpatches: A benchmark and evaluation of handcrafted and learned local descriptors. In *CVPR*, pages 5173–5182, 2017.

Matthew Brown, Gang Hua, and Simon Winder. Discriminative learning of local image descriptors. *T-PAMI*, 33(1):43–57, 2010.

Olivier Canévet and François Fleuret. Efficient sample mining for object detection. In *ACML*, pages 48–63. PMLR, 2015.

Luca Cavalli, Viktor Larsson, Martin Ralf Oswald, Torsten Sattler, and Marc Pollefeys. Handcrafted outlier detection revisited. In *ECCV*, pages 770–787. Springer, 2020.

Ching-Yao Chuang, Joshua Robinson, Yen-Chen Lin, Antonio Torralba, and Stefanie Jegelka. Debiased contrastive learning. *NeurIPS*, 33:8765–8775, 2020.

Ondrej Chum, Tomas Werner, and Jiri Matas. Two-view geometry estimation unaffected by a dominant plane. In *CVPR*, volume 1, pages 772–779. IEEE, 2005.

Rodrigo De Bem, Anurag Arnab, Stuart Golodetz, Michael Sapienza, and Philip Torr. Deep fully-connected part-based models for human pose estimation. In *ACML*, pages 327–342. PMLR, 2018.

Daniel DeTone, Tomasz Malisiewicz, and Andrew Rabinovich. Superpoint: Self-supervised interest point detection and description. In *CVPRW*, pages 224–236, 2018.

Yuhe Jin, Dmytro Mishkin, Anastasiia Mishchuk, Jiri Matas, Pascal Fua, Kwang Moo Yi, and Eduard Trulls. Image matching across wide baselines: From paper to practice. *IJCV*, 129(2):517–547, 2021.

Michel Keller, Zetao Chen, Fabiola Maffra, Patrik Schmuck, and Margarita Chli. Learning deep descriptors with scale-aware triplet networks. In *CVPR*, pages 2762–2770, 2018.

Vijay Kumar BG, Gustavo Carneiro, and Ian Reid. Learning local image descriptors with deep siamese and triplet convolutional networks by minimising global loss functions. In *CVPR*, pages 5385–5394, 2016.

Jongmin Lee, Byungjin Kim, and Minsu Cho. Self-supervised equivariant learning for oriented keypoint detection. In *CVPR*, pages 4847–4857, 2022.

David G Lowe. Distinctive image features from scale-invariant keypoints. *IJCV*, 60(2): 91–110, 2004.

Zixin Luo, Tianwei Shen, Lei Zhou, Jiahui Zhang, Yao Yao, Shiwei Li, Tian Fang, and Long Quan. Contextdesc: Local descriptor augmentation with cross-modality context. In *CVPR*, pages 2527–2536, 2019.

Zixin Luo, Lei Zhou, Xuyang Bai, Hongkai Chen, Jiahui Zhang, Yao Yao, Shiwei Li, Tian Fang, and Long Quan. Aslfeat: Learning local features of accurate shape and localization. In *CVPR*, June 2020.

Jiayi Ma, Xingyu Jiang, Aoxiang Fan, Junjun Jiang, and Junchi Yan. Image matching from handcrafted to deep features: A survey. *IJCV*, 129(1):23–79, 2021.

Anastasiia Mishchuk, Dmytro Mishkin, Filip Radenovic, and Jiri Matas. Working hard to know your neighbor's margins: Local descriptor learning loss. *NeurIPS*, 30, 2017.

Milan Pultar, Mishkin Dmytro, and Jiri Matas. Leveraging outdoor webcams for local descriptor learning. *Proceedings of CVWW*, feb 2019.

Jerome Revaud, Cesar De Souza, Martin Humenberger, and Philippe Weinzaepfel. R2d2: Reliable and repeatable detector and descriptor. *NeurIPS*, 32, 2019.

Florian Schroff, Dmitry Kalenichenko, and James Philbin. Facenet: A unified embedding for face recognition and clustering. In *CVPR*, pages 815–823, 2015.

Yifan Sun, Changmao Cheng, Yuhan Zhang, Chi Zhang, Liang Zheng, Zhongdao Wang, and Yichen Wei. Circle loss: A unified perspective of pair similarity optimization. In *CVPR*, pages 6398–6407, 2020.

Yurun Tian, Bin Fan, and Fuchao Wu. L2-net: Deep learning of discriminative patch descriptor in euclidean space. In *CVPR*, pages 661–669, 2017.

Yurun Tian, Xin Yu, Bin Fan, Fuchao Wu, Huub Heijnen, and Vassileios Balntas. Sosnet: Second order similarity regularization for local descriptor learning. In *CVPR*, pages 11016–11025, 2019.

Yurun Tian, Axel Barroso Laguna, Tony Ng, Vassileios Balntas, and Krystian Mikolajczyk. Hynet: Learning local descriptor with hybrid similarity measure and triplet loss. *NeurIPS*, 33:7401–7412, 2020.

Michał Tyszkiewicz, Pascal Fua, and Eduard Trulls. Disk: Learning local features with policy gradient. *NeurIPS*, 33:14254–14265, 2020.

Shuang Wang, Yanfeng Li, Xuefeng Liang, Dou Quan, Bowu Yang, Shaowei Wei, and Licheng Jiao. Better and faster: Exponential loss for image patch matching. In *ICCV*, pages 4812–4821, 2019.

Chao-Yuan Wu, R Manmatha, Alexander J Smola, and Philipp Krahenbuhl. Sampling matters in deep embedding learning. In *ICCV*, pages 2840–2848, 2017.

Guopeng Zhang and Jinhua Xu. Discriminative feature representation for person re-identification by batch-contrastive loss. In *ACML*, pages 208–219. PMLR, 2018.

Linguang Zhang and Szymon Rusinkiewicz. Learning local descriptors with a cdf-based dynamic soft margin. In *ICCV*, pages 2969–2978, 2019.