Self-supervised Example Difficulty Balancing for Local Descriptor Learning (Supplementary Material)

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

In this supplementary material, in Section 1, we provide the proof of adaptive weight assignment of gradient mentioned in the paper. We present additional experiment results for the ablation studies in Section 2. Furthermore, we visualize the dynamic adjustment of example sampling during training in Section 3.

1. Proof of Adaptive Weight Assignment of Gradient in Balance Loss

To make $\mathcal{L}_{Balance}$ serve to minimize the positive distance and maximize the negative distance, we define different similarity measure functions $s(d)$ for positive and negative distances to determine the direction and magnitude of the gradient descent, which can achieve adaptive weight assignment of gradient. The following is the proof.

As mentioned in the paper, the similarity measure functions used in balance loss are:

$$\begin{aligned} s_i^{pos} &= \left(d_i^{pos} - P_{pos}\right)^{\alpha}, \\ s_i^{neg} &= \left(d_i^{neg} - P_{neg}\right)^{\alpha}. \end{aligned} \tag{1}$$

Additionally, the selection of triplet tuples and the computation of negative and positive distances can be described as follows:

$$\begin{aligned} d_i^{neg} = \min_{\forall j, j \neq i} \left(d\left(\boldsymbol{x}_i, \boldsymbol{x}_j\right), d\left(\boldsymbol{x}_i, \boldsymbol{x}_j^{+}\right), d\left(\boldsymbol{x}_i^{+}, \boldsymbol{x}_j\right), d\left(\boldsymbol{x}_i^{+}, \boldsymbol{x}_j^{+}\right)\right), \\ d_i^{pos} = d\left(\boldsymbol{x}_i, \boldsymbol{x}_i^{+}\right). \end{aligned} \tag{2}$$

Combined with Eq. (1) and Eq. (2), the gradient of our overall loss function can be represented as:

$$\begin{aligned}\frac{\partial \mathcal{L}_{Balance}}{\partial D} = & \frac{1}{N}\sum_{i=1}^{N}\frac{\partial \mathcal{L}}{\partial s_i^{pos}}\left(\frac{\partial s_i^{pos}}{\partial d_i^{pos}}\frac{\partial d_i^{pos}}{\partial \boldsymbol{p_i}}+\frac{\partial s_i^{pos}}{\partial d_i^{pos}}\frac{\partial d_i^{pos}}{\partial \boldsymbol{p_i}^{+}}\right) \\ & +\frac{1}{N}\sum_{i=1}^{N}\frac{\partial \mathcal{L}}{\partial s_i^{neg}}\left(\frac{\partial s_i^{neg}}{\partial d_i^{neg}}\frac{\partial d_i^{neg}}{\partial \boldsymbol{p_i}}+\frac{\partial s_i^{neg}}{\partial d_i^{neg}}\frac{\partial d_i^{neg}}{\partial \boldsymbol{p_i}^{-}}\right),\end{aligned} \tag{3}$$

where N denotes the batch size, D denotes $\left\{\boldsymbol{p_1}, \boldsymbol{p_2}, \ldots, \boldsymbol{p_N}; \boldsymbol{p_1}^{+}, \boldsymbol{p_2}^{+}, \ldots, \boldsymbol{p_N}^{+}\right\}$ representing all embeddings of the corresponding patches in a batch. $\boldsymbol{p_i}^{-}$ and $\boldsymbol{p_i}^{+}$ represent the selected patch embeddings that have the maximum or minimum distance from the anchors. In our settings, $\frac{\partial \mathcal{L}}{\partial s_i^{pos}}$ is equal to 1 for positive and negative similarity measure functions, so combined with Eq. (1) the above equation can be reduced to

$$\begin{aligned}\frac{\partial \mathcal{L}_{Balance}}{\partial D} = & \frac{1}{N}\sum_{i=1}^{N}\left(\alpha\left(d_i^{pos}\right)^{\alpha-1}\frac{\partial d_i^{pos}}{\partial \boldsymbol{p_i}}+\alpha\left(d_i^{pos}\right)^{\alpha-1}\frac{\partial d_i^{pos}}{\partial \boldsymbol{p_i}^{+}}\right) \\ & +\frac{1}{N}\sum_{i=1}^{N}\left(\left(\alpha\left(d_i^{neg}-P_{neg}\right)^{\alpha-1}\right)\frac{\partial d_i^{neg}}{\partial \boldsymbol{p_i}}\right. \\ & \left.+\left(\alpha\left(d_i^{neg}-P_{neg}\right)^{\alpha-1}\right)\frac{\partial d_i^{neg}}{\partial \boldsymbol{p_i}^{-}}\right).\end{aligned} \tag{4}$$

Note that in Eq. (4), α is selected to range from even numbers greater than or equal to 2 and $\alpha\left(d_i^{neg}-P_{neg}\right)^{\alpha-1}$ is a negative number when it comes to actual training, so the direction of gradient decline is consistent with the triplet loss. The original triplet loss and its derivative can be represented as:

$$\mathcal{L}_{Triplet} = \frac{1}{N}\sum_{i=1}^{N}\max\left(0, t+d_i^{pos}-d_i^{neg}\right), \tag{5}$$

$$\frac{\partial \mathcal{L}}{\partial D} = \frac{1}{N}\left(\sum_{i=1}^{N}\left(\frac{\partial d_i^{pos}}{\partial \boldsymbol{p_i}}+\frac{\partial d_i^{pos}}{\partial \boldsymbol{p_i}^{+}}\right)+\sum_{i=1}^{N}\left(-\frac{\partial d_i^{neg}}{\partial \boldsymbol{p_i}}-\frac{\partial d_i^{neg}}{\partial \boldsymbol{p_i}^{-}}\right)\right), \tag{6}$$

when compared Eq. (3) with Eq. (6), $\frac{\partial s_i^{pos}}{\partial d_i^{pos}}$ and $\frac{\partial s_i^{neg}}{\partial d_i^{neg}}$ are the terms that differentiate the derivative of balance loss and the derivative of triplet loss. In $s(d)$, the introduction of the exponential term allows the updated weight of its gradient to automatically fit the size of the distances of positive or negative examples. The behaviors of triplet loss and balance loss and their derivatives are shown and compared in Figure 1.

In the backpropagation of balance loss, due to the similarity measure functions $s(d)$, the derivation of the network will generate additional functions with respect to the value of distances. Usually, we can set them as a primary function so that their gradient values vary linearly with the increase or decrease of distances, as shown in Figure 1 (c, d), so that

the network can pay more attention and produce larger weight for patch embeddings with smaller d_i^{neg} and larger d_i^{pos} in the backpropagation process of each batch.

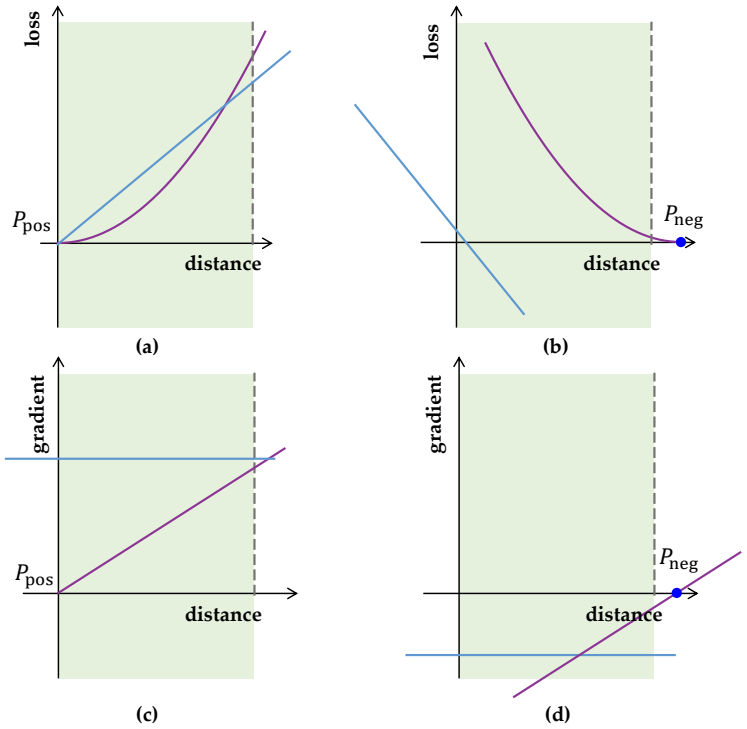


Figure 1: Comparison between balance loss and triplet loss, and a schematic of their respective gradients. (a), (c) represent the positive part loss and its gradient, while (b), (d) represent the negative part loss and its gradient. **Purple** indicates the proposed balance loss, **blue** indicates triplet loss. The green region is the actual distribution range of L_2 distances of patches after normalization, which indicates the action range of loss functions.

2. Ablation Studies

In this section, we present detailed analyses of the contribution made by each component of Self-TNet towards the overall performance, as well as the factors that influence its efficacy. Specifically, we trained different models on Liberty dataset and tested them on the Hpatches (Balntas et al. (2017)) matching task (data split 'full') to obtain the influence of our two modifications - loss function and annealing training.

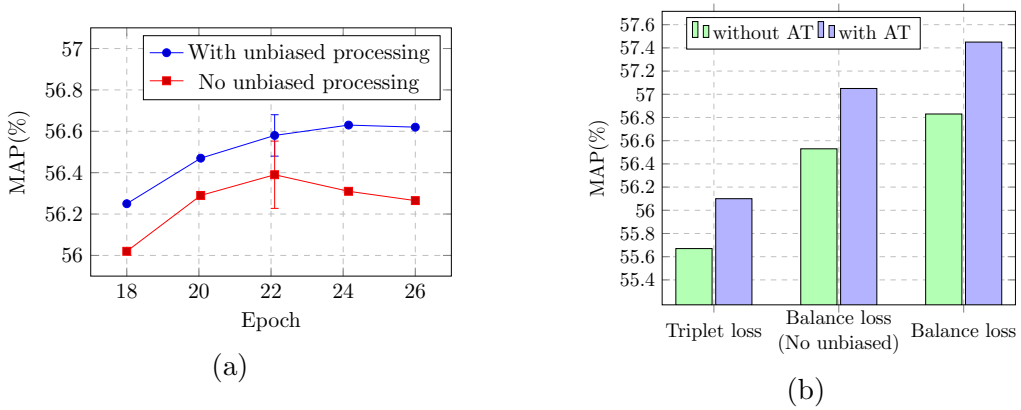
2.1. Balance Loss

Adaptive weight assignment of gradient. In Table 1, compared to triplet loss, our balance loss yields a performance improvement of approximately 0.7 mAP. Additionally, we have implemented a version of balance loss without unbiased processing through a self-supervised network, which achieves an mAP of 56.53. In contrast, the network that uses

Modification	Choice	Has unbiased processing	MAP(%)
Loss Function	Triplet loss	✗	55.67
	Triplet loss	✓	56.10
	Balance loss	✗	56.53
	Balance loss	✓	56.83
Annealing Training	AT+Triplet loss	✗	56.25
	AT+Triplet loss	✓	56.59
	AT+Balance loss	✗	57.05
	AT+Balance loss	✓	57.45

Table 1: Ablation experiments to explore the performance improvement of each modification. The result is tested on data split 'full' from the Hpatches benchmark. mAP score for subtask image matching is reported.

the complete balance loss results in an mAP of 56.83. Overall, further performance gains of approximately 0.3 to 0.4 are achievable with our complete balance loss as compared to its unbiased processing-free version. These results provide evidence of the effectiveness of our improvement strategy and the compatibility between different modifications.



(a) (b)

Figure 2: Experiment results. (a): Performance comparison of the models with and without a supervising network. (b): Performance comparison of the models with triplet loss and balance loss before and after annealing training.

Unbiased processing with a supervising network. In Table 1, we observe that unbiased processing typically operates effectively in conjunction with other components, and can result in a moderate performance gain when integrated into models. Furthermore, we have implemented unbiased processing to improve the performance of the original triplet loss. This adaptation yields an mAP increase from 55.67 to 56.10. Figure 2(a) shows that as the number of training epochs increases, the mAP performance of the model with balance loss but without unbiased processing tends to first increase and then decrease. Compared with the unbiased processing-free version, the introduction of the supervising network alleviates the problem of the model overly focusing on difficult examples in the

later stages of training, and provides a more relaxed selection interval for the optimal model, without worrying about the timing of early stopping. At the same time, Figure 2(a) shows the mean and variance of the performance using the balance loss (no unbiased processing) with 4 different random seeds trained after 22 epochs under the same settings, which has a larger variance compared with the balance loss. Thus, the supervising network can also help the supervised network to improve performance stability.

2.2. Annealing Training

The main hyperparameters in AT are the step sizes of *bs* and *thr*. In this paper, we use *bs* decreasing by 128 at a time and *thr* increasing by 0.05 at a time as a general adjustment strategy. In general, smaller step sizes mean smoother transitions between the two training phases and usually better performance, but they also increase the computational cost. Here we balance the effects of both to choose the hyperparameters.

AT has a similar effect on the performance improvements shown in Figure 2(b), as long as the model has previously undergone the HNS process. In addition, we can see from Figure 2(b) that the model trained with balance loss can get more performance improvement in the annealing training. This can be attributed to the fact that the balance loss can handle the difficulty of the training examples by assigning more appropriate adaptive weights than the triplet loss, thus helping to learn more abstract rules from the examples. This proves that our new loss function and training strategy can collaborate to provide the best performance.

2.3. Influence of Hyperparameters

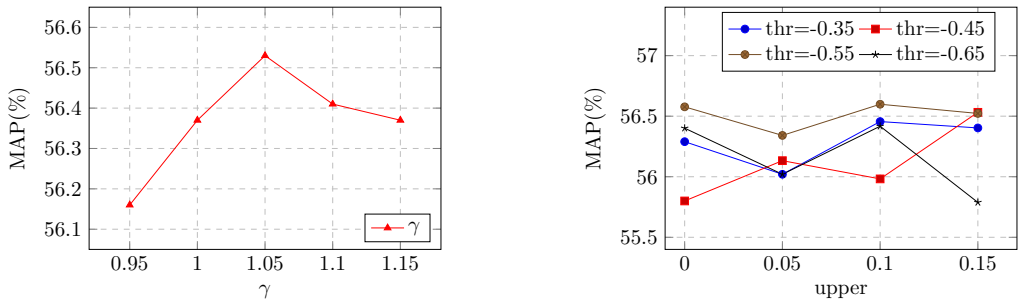


Figure 3: Experiment results. (a): The effect of hyperparameter γ on the model performance. (b): The effect of hyperparameters *threshold* and *upper* on the model performance.

Attention Coefficient γ when constructing balance loss. According to the previous discussions, the γ can be interpreted as a parameter here that balances the attention between positive and negative distances. Specifically, a value greater than 1 leads to greater attention for negative part, and a value less than 1 indicates greater attention for positive

part. The influence of the magnitude of hyperparameter γ on the final mAP performance of the model can be seen in the Figure 3(a). Through hyperparameter search, we found that the model achieved the highest mAP performance when A=1.05 for the current training settings.

Upper and threshold in sampling unbiased processing. The hyperparameter values of *upper* and *threshold* can be used to adjust the mitigation of weight for hard examples. Usually, the higher the threshold value means the higher the magnitude of alleviation, and the higher the upper value means the higher the number of samples involved in mitigation. The influence of *upper* and *threshold* on performance is shown in Figure 3(b). In addition, for models that can achieve similar performance in preliminary training, we found that selecting models that trained at a lower *threshold* and a lower *upper* conditions as possible could lead to larger performance improvement in subsequent annealing training.

3. Visualizations of the Training Process

The visualization of training sample distributions as training proceeds:

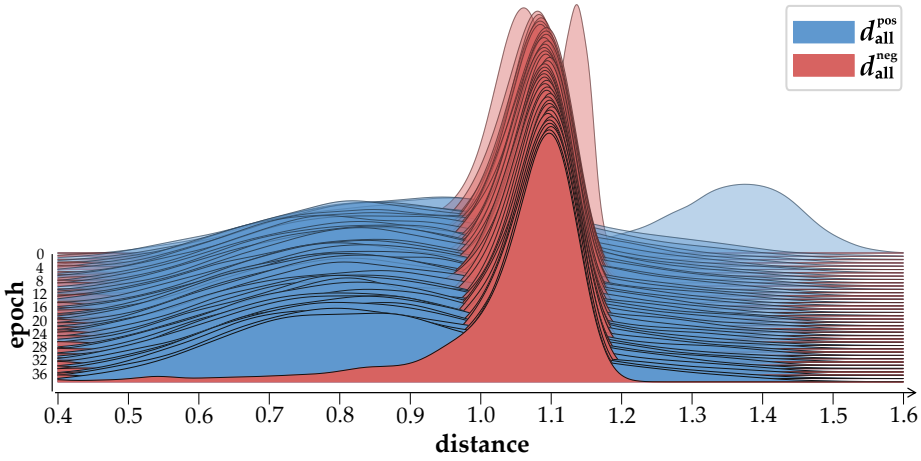


Figure 4: Positive and negative distance distributions with different epochs. As the training progresses the positive distance distribution tends to move in the positive direction of the x-axis, while the negative distance distribution tends to move in the opposite direction. The visualization is performed by taking the first batch in each iteration.

The adaptive variation of the zero position P_{neg} as training proceeds:

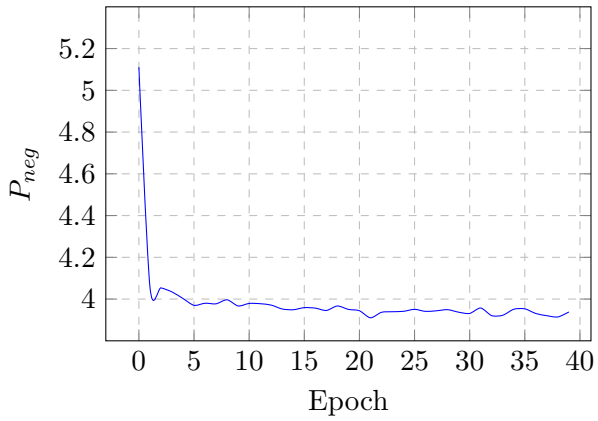


Figure 5: Adaptive changes of P_{neg} values during gradient modulation.

References

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.