EE3-25 Deep Learning (2018-2019): Coursework

Martin Ferianc

mf2915@ic.ac.uk, CID:00984924

Abstract

This work investigates denoising and describing images in the N-HPatches dataset, a noisy adaptation of Homography patches (HPatches) dataset [1]. The baseline approach explores a shallow U-Net [2] in combination with L2-Net [3] to process a noisy image to a set of descriptors. The improved approach consists of a deeper DNCNN [4] with three denoising branches paired with a residual network to produce image descriptors. The best approach, having 128 filters and 3, 5, 9 nodes per branch together with a 20-layer deep residual network achieved approximately 5%, 4% and 1% mean improvement over the baseline in patch retrieval, image matching of learnt representations and patch verification tasks respectively.

1. Introduction

1.1. Problem

The task in this work is to perform image denoising and descriptor generation based on the noisy version of HPatches (Homography patches) dataset [1], named N-HPatches with the use of deep learning. The generalisation performance is being evaluated by using three separate metrics: patch verification, image matching and patch retrieval.

Patch verification measures the ability of a descriptor to classify whether two patches are extracted from the same measurement. Image matching tests to what extent a descriptor can correctly identify correspondences in two images. Patch retrieval tests how well a descriptor can match a query patch to a pool of patches extracted from many images, including many distractors [5]. All tasks are evaluated with mean average precision (mAP)[1].

The overall goal is to bring similar samples closer together in Euclidean space and otherwise for dissimilar samples, characterised for example by a triplet loss[1] [6].

1.2. Dataset

The N-HPatches dataset and its accompanying benchmark are primarily meant for evaluation of local descriptors in images. The input images themselves are 32×32 black and white images to which three levels of increasing noise values, together with affine transformations, were applied. The output dimensions for clean image remain the same and the descriptor is a 128 dimensional vector. The prior over the noise is unknown, but it is assumed to be additive. There are 31,179 and 19,050 images for denoising for training and validation respectively and 2,000 and 200 triplets divided across training and validation for the descriptors of the initially noisy images, using split *'a'* [2].

2. Baseline Approach

The baseline approach consists of two separate models, one for image denoising and the second for generation of the descriptors, which are connected in series.

The denoising network uses a bottom part of UNet, seen in Figure 12, which has been originally used for image segmentation [2], giving 59,777 trainable parameters. The descriptor network is following L2-Net [3], seen in Figure 13, with 1,336,032 trainable parameters, which was developed to extract local features from images, where the output descriptor can be matched in Euclidean space by L2 distance.

2.1. Training

Both networks are being trained by stochastic gradient descent (SGD). Additionally, the denoising network, employs Nesterov momentum. The denoising network employs mean absolute error loss, while the descriptor network is using triplet loss, where the weight parameters are shared across three identical networks. Both networks are trained only for a single epoch on all data. The learning rate of the denoising network was set as 0.00001 with momentum set to 0.9, while the descriptor's learning rate was 0.1.

SGD is known for fine generalisation properties, however it does not adapt its learning parameters after the weight update. This behaviour can cause a local convergence. A better option would be to use Adam optimiser [7], whose automatically updated hyperparameters have intuitive interpretation and typically require little tuning.

While, the triplet loss is reasonable, mean absolute error loss can be replaced by a mean squared loss, which due to non-linear weighting for errors enables faster convergence, while preserving the performance as proposed in [8]. Additionally, by using its non-linear property, it can be particularly effective for detecting high-frequency noise in the denoising network, which seems to be dominant as seen in Figure 3.

The improved approach should also consider including a stopping condition and an increased number of epochs to avoid underfitting or overfitting respectively, which was observed in the baseline by looking at Figures 8 and 9.

[1]For more details, please refer to the Appendix.

[2]The experiments were performed with respect to commit: `719576a29567a9a2b86658496eea7f196bc314f6`.

Although, the descriptor network uses regularisation by including batch normalisation (BN) [9] after each convolutional layer, the denoising network does not include any regularisation. The improved approach could address that by similarly using BN, to ensure balanced signal propagation and weight updates especially during training, after each convolution. Another means of regularisation can be achieved by using L2 loss for the individual weight parameters to avoid the weights adapting too much to outliers.

2.2. Evaluation

Figure 3 demonstrates on the left a sample noisy patch, in the middle is a denoised patch and on the right is the reference image. The Figure shows that a lot of the noise has been removed, however the proportions of different frequencies of noise being removed are inadequate and the cleaned image lacks edges, which suggests that the improved approach should aim to balance noise removal of different types and pay increased attention to high frequency noise.

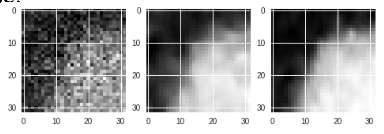

Figure 3: Denoising with the baseline method.

The exact results can be seen in Tables 1, 2 and 3. Overall, the baseline approach achieved mean scores of **0.42** mAP on the patch retrieval, **0.61** mAP on the patch verification and **0.107** mAP on the image matching. It is deemed that the denoising network in particular is not deep enough and hence it achieves a low mAP as seen in Table 3. However, it is difficult to draw conclusions as the complete architecture consists or multiple parts, which are dependant on each other.

To summarise, the main challenges are: *a)* to cope with different noise types in the noisy images, *b)* build a deeper network with regularisation and *c)* finding the best model which has to be jointly optimised.

3. Improved Approach

The improved approach still consists of two separate networks connected in series. The belief is that by separating the tasks the networks can be more specialised in the extraction and evaluation of current features which are learnt from the data, and thus achieve better performance.

3.1. Denoising Network

The denoising model is inspired by Zhang *et al.* [4] which is able to process additive Gaussian noise with unknown intensity by using residual learning. The general idea is that the original image $\boldsymbol{x}$ is observed as a degraded image $\boldsymbol{y}$ with noise $\boldsymbol{n}$ as $\boldsymbol{y} = \boldsymbol{x} + \boldsymbol{n}$. Their approach assumes no prior information about the noise, except being additive, which is similar to the assumption made in this work. It includes a single deep branch which contains a number of 3×3 convolutions interleaved with BNs and rectified linear units (ReLUs) [10]. Note, that all convolutions have the same capacity i.e. the same number of filters. The single branch attempts to learn the noise which is present in the data and subtract it in the last layer from the original noisy sample. Hence, it is assumed to be learning a single type of noise $\boldsymbol{n}$ which is then subtracted to arrive at the original clean image $\boldsymbol{x}$. The main benefit of this architecture is that the residual mapping is easier to optimise [11].

However, Zhang's approach, similarly to baseline, is limited to learning a particular noise $\boldsymbol{n}$, and not a combination of several external effects, which can have different intensity or carry hierarchical spatial information over the image. Hence, their approach lacks generalisation to the real world, where a number of different noise types can be present simultaneously. This work proposes an extension to the DNCNN by having several residual branches which learn different types of noise, characterised by different parameters with various levels of influence which are then gradually removed from the original.

The improved approach makes an assumption that the noise can be hierarchically subtracted and each noise has its own features and intensity $\boldsymbol{\theta_i}$ which can be learnt through a deep neural network. The formulation of the additive noise then changes into $\boldsymbol{y} = \boldsymbol{x} + \sum_{i=1}^{\infty} \boldsymbol{\theta_i n_i}$, where each noise $\boldsymbol{n_i}$, with weight $\boldsymbol{\theta_i}$, can be residually learnt and subtracted in order to gradually arrive at a clean image.

The different types of noise and their influence are modelled through different capacity of individual branches, kernel sizes, number of filters and a varying number of operations, which represent main hyper-parameters of the model's architecture, which target challenges *a)* and *b)*.

The proposed denoising network is based on the baseline by using convolutions to capture the spatial relationships of noise, however it does not include upsampling and downsampling. The belief is that information about the original, which is scarce, given the dimensions of the image is being lost, especially by using pooling. Thus, each branch is being constructed through a repeating combination of convolution, BN to account for regularisation and avoiding vanishing gradients during training. Similarly, ReLU is proposed as an activation function to speed up the training process and improve the denoising performance, by always subtracting the learnt noise $\boldsymbol{n_i}$ and avoid adding additional noise to the result.

After all branches subtract their learnt noise $\boldsymbol{n_i}$, the cleaned image is then forwarded to the descriptor network.

3.2. Descriptor Network

The improved descriptor network is similarly inspired by a residual architecture, which has been proven particularly worthy for end-to-end learning [11] and without the need for hand-crafted features and challenging optimisation.

In comparison to the baseline, it is unclear, in which part of the baseline descriptor network, features are being extracted and descriptors are being drawn. The improved ap-

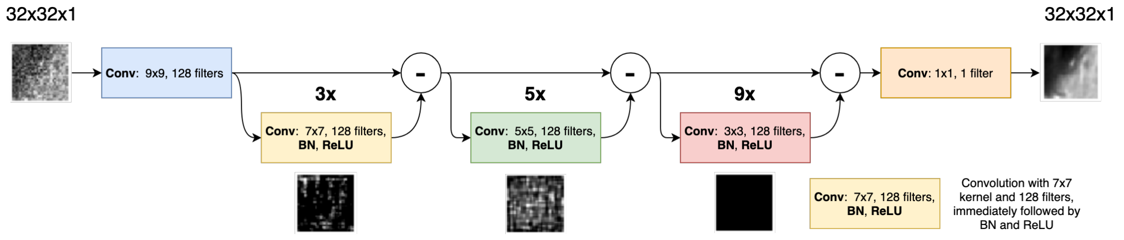
32x32x1
Conv: 9x9, 128 filters
3x
Conv: 7x7, 128 filters, BN, ReLU
5x
Conv: 5x5, 128 filters, BN, ReLU
9x
Conv: 3x3, 128 filters, BN, ReLU
Conv: 1x1, 1 filter
32x32x1
Conv: 7x7, 128 filters, BN, ReLU
Convolution with 7x7 kernel and 128 filters, immediately followed by BN and ReLU

Figure 1: Improved denoising architecture.

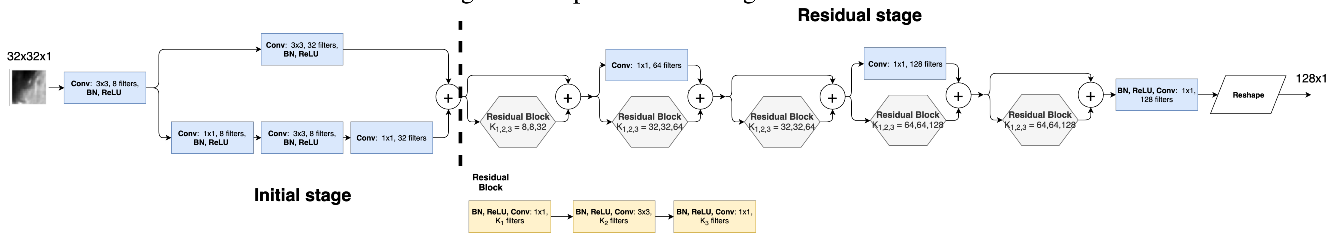
32x32x1
Conv: 3x3, 8 filters, BN, ReLU
Conv: 3x3, 32 filters, BN, ReLU
Conv: 1x1, 8 filters, BN, ReLU
Conv: 3x3, 8 filters, BN, ReLU
Conv: 1x1, 32 filters
Initial stage
Residual stage
Residual Block $K_{1,2,3}$ = 8,8,32
Conv: 1x1, 64 filters
Residual Block $K_{1,2,3}$ = 32,32,64
Residual Block $K_{1,2,3}$ = 32,32,64
Conv: 1x1, 128 filters
Residual Block $K_{1,2,3}$ = 64,64,128
Residual Block $K_{1,2,3}$ = 64,64,128
BN, ReLU, Conv: 1x1, 128 filters
Reshape
128x1
Residual Block
BN, ReLU, Conv: 1x1, K_1 filters
BN, ReLU, Conv: 3x3, K_2 filters
BN, ReLU, Conv: 1x1, K_3 filters

Figure 2: Improved descriptor architecture.

proach attempts to target this problem by first introducing a rich variety of features in the earlier stages of inference and later gradually draw descriptors from the detected features with the help of residual blocks which are similar in nature to the previously introduced denoising network.

The improved approach is based the second version of ResNet [12], which is a combination of the Inception [13] and the original ResNet [11], which has been used for generating descriptors [14]. The architecture can be separated into an *initial stage* and a later *residual stage*.

The *initial stage* is comprised of two branches of convolutions similarly structured as the denoising network comprising from convolution, BN and ReLU activation. Note, that the kernel size is smaller as in the denoising network mainly for compute reasons. This part of the network introduces keypoint locations as well as their attributes that can be later fed into the *residual stage*.

The *residual stage* comprises of residual blocks consisting of three blocks, each bearing BN, activation and then convolution with 1×1, 3×3 and 1×1 kernel sizes with an increasing number of filters in the last convolution. The *residual stage* attempts to locate a subset of the actual descriptors from the previous stage which is gradually added and reshaped to form the final result. Similarly to the denoising network, it leaves out subsampling to avoid the loss of information.

While, the *initial stage* usually remains fixed together with kernel sizes of convolutions, the depth of the network i.e. the number of residual blocks can be thought of as a hyperparameter that can be tuned for the best generalisation performance to combat challenge *b)*.

3.3. Training

The choosing of the hyperparameters for both networks had to be done jointly to combat challenge *c)*, as the networks are not independent with respect to the tasks that they were evaluated upon. The hyperparameters that were deemed to bear the most influence on the performance were the depth of the descriptor network, the number of filters and the number of nodes per the subtract block of the denoising network. Each of these parameters influences the capacities of the networks as well as the compute complexity during inference and training.

In particular, the denoising network was fixed to three separate branches due to resource limitations. Each separate branch bears a different characterising kernel size e.g.: the input 9×9 and then: 7×7, 5×5 or 3×3 respectively, which is deemed to be the main characteristic of that noise type n_i. The most influence is placed on the high-frequency noise, by gradually increasing the number of convolutions with lower dimensional kernels.

The parameters were arbitrarily selected ranging from depth, i.e. the number of residual blocks, of the descriptor network being 20, 29 or 38, the uniform number of filters 32, 64, 128 across the denoising network and the number of nodes per subtract block for the denoising network were 1,3,5 or 3,5,9 or 5,9,15, giving in total 27 different combinations. It was deemed that the more complicated networks, e.g.: the ones with the depth of 38 or the number of nodes 5,9,15 will tend to overfit while the simplest networks e.g. with 32 filters will underfit.

The experiments were performed on subsample of data: 2000 samples for training and 200 samples for validation per network respectively and the training endured only for 2 epochs. This disadvantages deeper models, which might not be able to improve fast enough in the initial stages of training. The training and evaluation time averaged approx. 2 hours per combination on Google Colaboratory's GPUs.

Both networks were distantly trained by using an Adam optimiser and an initial learning rate of 0.001 and other parameters were set as default. The outcomes can be seen in Figures 4, 5 and 6[3] for each of the evaluation tasks.

[3]Larger plots are made available in the Appendix.

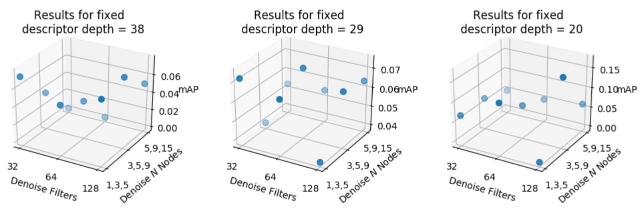

Figure 4: Patch verification results.

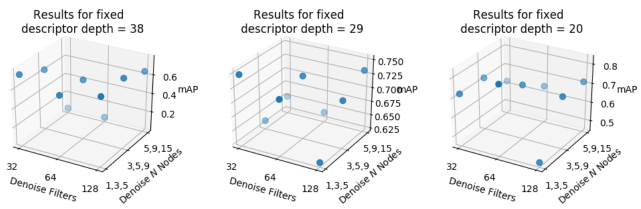

Figure 5: Image matching results.

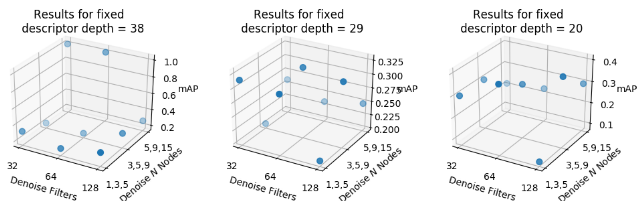

Figure 6: Patch retrieval results.

The hypothesis that more complex architectures are going to overfit, while the simple architectures are going to underfit was partially proven. While the performance of the denoising network was observed to be the best for the middle-sized network - 3,5,9 layers per subtract block and 128 filters, the performance of the descriptor steadily deteriorated while increasing depth from 20 to 38. In general, the middle sized network having 3,5,9 nodes per subtract blocks for the denoising model showed the best average performance across a varying number of filters. The overall best combination, which achieved the best performance across all tasks was with the depth of the descriptor network being 20 and the number of nodes and filters being 3,5,9 and 128 respectively. The final denoising network is depicted in Figure 1. It also demonstrates sample noise types learnt by the different branches. The Figure 2 depicts the final descriptor network. The total number of trainable parameters and the corresponding capacity of the networks were 5,805,057 and 1,193,840 for denoising and descriptor respectively.

The final architectures were similarly trained with the Adam optimiser with initial learning rate 0.001, and other parameters were set as default, which was gradually decreased by a factor of 10 approximately every 5 epochs for 20 epochs in total on all data. If the learning would hit a plateau, during consecutive mini-batches, which were the same as in the baseline, the learning rate was decelerated by a factor of 10. The convolutions' weights were initialised with He's initialisation [15] and regularised with L2 loss with 0.0001 penalty. Additionally, the convolutions for the denoising network included zero-padding to preserve the original height and width of the feature maps.

Additionally, to prevent overfitting, the network would stop training if validation loss stopped improving. The losses steadily decreased during the first epochs and smoothly stopped decreasing by approximately the 20th epoch. The plots can be seen in Figures 10 and 11 respectively. The achieved results for the three different tasks can be seen in Tables 1, 2 and 3. A sample denoised image can be seen in Figure 7 which shows on the left a noisy patch, in the middle is a denoised patch and on the right is the reference image.

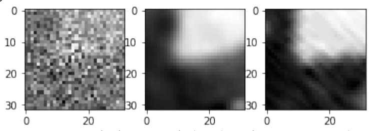

Figure 7: Denoising with the improved method.

In comparison to the baseline it appears to better hold the structural integrity of the original image, however it still remains blurry.

4. Evaluation

In comparison to the baseline approach the improved approach achieved approximately **5%**, **4%** and **1%** mean improvement on patch retrieval, image matching and patch verification tasks respectively. Overall, the improved approach achieved mean scores of **0.46** mAP on the patch retrieval, **0.65** mAP on the patch verification and **0.12** mAP on the image matching. The training time increased approximately 20×, while the number of parameters for the descriptor network decreased, the number of parameters for the denoising network increased approximately 100×.

Therefore, the improvement was achieved with a great computational cost, where the training was multiple times longer as in the baseline and the capacity of the network for denoising was increased almost 100×. This result implied that, the evaluation strategy did not identify the global minimum in terms of error because it was not trained for sufficient time as well as not enough data was supplied to draw conclusions. A better way how to pick the winning models would be by using a rigorous optimisation/evaluation strategy, such as Bayesian optimisation [16], for a larger number of epochs and all the available data, which could run the different options in parallel.

A drastic reduction in compute could be achieved by combining the denoising and descriptor generation into a single network. The combination would be trained sequentially by training and freezing responsible parts for denoising and descriptor generation and vice versa. In the future, another performance boost could be considered by pretraining the networks on a different dataset. Another experiments could consider balancing the different parts of the denoising network which could be more generalisable without any prior assumptions. Lastly, the training data could include data generated through augmentation of the original dataset to include more samples for training.

References

[1] V. Balntas, K. Lenc, A. Vedaldi, and K. Mikolajczyk, "HPatches: A benchmark and evaluation of handcrafted and learned local descriptors," in *CVPR*, 2017.

[2] O. Ronneberger, P.Fischer, and T. Brox, "U-Net: Convolutional networks for biomedical image segmentation," in *Medical Image Computing and Computer-Assisted Intervention (MICCAI)*, vol. 9351 of *LNCS*, pp. 234–241, Springer, 2015.

[3] Y. Tian, B. Fan, and F. Wu, "L2-Net: Deep learning of discriminative patch descriptor in euclidean space," in *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 6128–6136, July 2017.

[4] K. Zhang, W. Zuo, Y. Chen, D. Meng, and L. Zhang, "Beyond a gaussian denoiser: Residual learning of deep CNN for image denoising," *CoRR*, vol. abs/1608.03981, 2016.

[5] A. Barroso and A. Lopez, "Keras triplet descriptor," 2019.

[6] D. P. Vassileios Balntas, Edgar Riba and K. Mikolajczyk, "Learning local feature descriptors with triplets and shallow convolutional neural networks," in *Proceedings of the British Machine Vision Conference (BMVC)* (E. R. H. Richard C. Wilson and W. A. P. Smith, eds.), pp. 119.1–119.11, BMVA Press, September 2016.

[7] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *CoRR*, vol. abs/1412.6980, 2014.

[8] H. Zhao, O. Gallo, I. Frosio, and J. Kautz, "Loss functions for image restoration with neural networks," *IEEE Transactions on Computational Imaging*, vol. 3, pp. 47–57, March 2017.

[9] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," in *Proceedings of the 32Nd International Conference on International Conference on Machine Learning - Volume 37*, ICML'15, pp. 448–456, JMLR.org, 2015.

[10] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "ImageNet classification with deep convolutional neural networks," in *Advances in Neural Information Processing Systems 25*, pp. 1097–1105, Curran Associates, Inc., 2012.

[11] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," *CoRR*, vol. abs/1512.03385, 2015.

[12] C. Szegedy, S. Ioffe, and V. Vanhoucke, "Inception-v4, inception-resnet and the impact of residual connections on learning," *CoRR*, vol. abs/1602.07261, 2016.

[13] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. E. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich, "Going deeper with convolutions," *CoRR*, vol. abs/1409.4842, 2014.

[14] Y. Ono, E. Trulls, P. Fua, and K. M. Yi, "Lf-net: Learning local features from images," *CoRR*, vol. abs/1805.09662, 2018.

[15] K. He, X. Zhang, S. Ren, and J. Sun, "Delving deep into rectifiers: Surpassing human-level performance on imagenet classification," in *Proceedings of the 2015 IEEE International Conference on Computer Vision (ICCV)*, ICCV '15, (Washington, DC, USA), pp. 1026–1034, IEEE Computer Society, 2015.

[16] C. M. Bishop, *Pattern Recognition and Machine Learning (Information Science and Statistics)*. Berlin, Heidelberg: Springer-Verlag, 2006.

A. Appendix

A.1. Pledge

I, Martin Ferianc, pledge that this assignment is completely my own work, and that I did not take, borrow or steal work from any other person without referencing, and that I did not allow any other person to use, have, borrow or steal portions of my work. I understand that if I violate this honesty pledge, I am subject to disciplinary action pursuant to the appropriate sections of Imperial College London.

I have discussed my work with Alex Montgomerie-Corcoran and Miroslav Gasparek.

A.2. Code and Results

The instructions to run the code are in the `README.md` in the zip package. Note, that all the results were generated with reference to `MatchLab-Imperial/keras_triplet_descriptor` repository's commit history: `719576a29567a9a2b86658496eea7f196bc314f6` on GitHub.

A.3. Evaluation Metrics

An image L_k is a collection of N patches $L_k = (x_{ik}, i = 1, ..., N)$.

In patch verification, any two patches x_1, x_2 produce a confidence score $s_i \in R$ that the two patches correspond. The quality of the approach is measured as the average precision (AP) of the ranked patches, namely AP $(y_{\pi 1}, ..., y_{\pi N})$ where π is the permutation that sorts the scores in decreasing order.

In image matching, descriptors are used to match patches from a reference image to a target one. Consider a pair of images $D = (L_0, L_1)$, where L_0 is the reference and L_1 is the target. The pair D is used for evaluation given a reference patch $x_{i0} \in L_0$, which determines the index $\sigma_i \in (1, ..., N)$ of the best matching patch and computes AP $(y_{\pi 1}, ..., y_{\pi N})$, where π is the permutation that sorts the scores in decreasing order.

In patch retrieval, descriptors are used to find patch correspondences in a large collection of patches, a large portion of which are distractors extracted from confounder images. Consider a collection $P = (x_0, (x_i, y_i), i = 1, ..., N)$ consisting of a query patch x_0, extracted from a reference image L_0, and all patches from images $L_k, k = 1, ..., K$ in the same sequence (matching images), as well as many confounder images. The collection P is used to evaluate an algorithm that assigns to each patch x_i a confidence score $s_i \in R$ that the patch matches the query x_0 and then computes the AP $(y_{\pi 1}, ..., y_{\pi N})$, where π is the permutation that sorts the scores in decreasing order, is computed.

For complete description, please refer to the Section 5 in the original paper [1].

A.4. Triplet-loss

The triplet loss is defined as:

$$max(\|\boldsymbol{f^a} - \boldsymbol{f^p}\|_2^2 - \|\boldsymbol{f^a} - \boldsymbol{f^n}\|_2^2 + \alpha, 0)$$

The triplet loss takes an anchor image/patch denoted by $\boldsymbol{f^a}$ and compares it to a positive sample $\boldsymbol{f^p}$ and a negative sample $\boldsymbol{f^n}$. The dissimilarity between the anchor sample and positive sample must be low and the dissimilarity between the anchor sample and the negative sample must be high. The variable α defines how far away the dissimilarities should be in Euclidean space. For this work, α was set to 1.

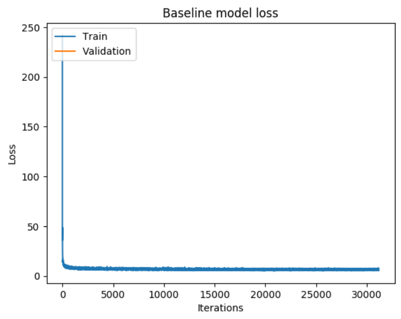

Figure 8: Loss for the baseline denoising network.

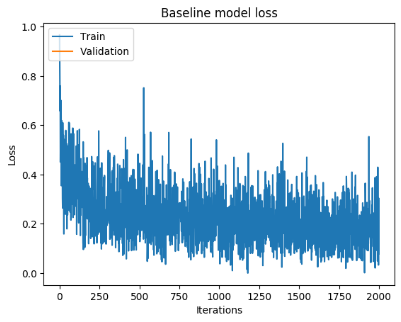

Figure 9: Loss for the baseline descriptor network.

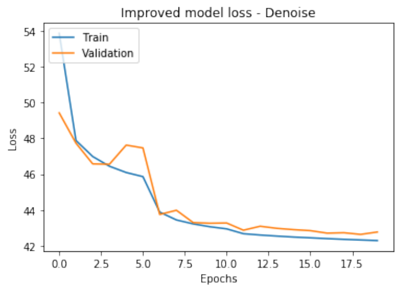

Figure 10: Loss for the improved denoising network.

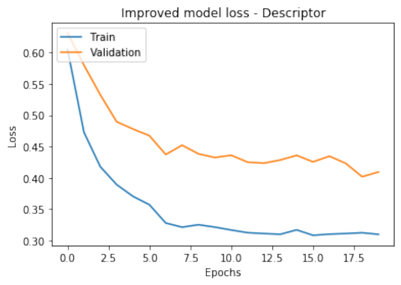

Figure 11: Loss for the improved descriptor network.

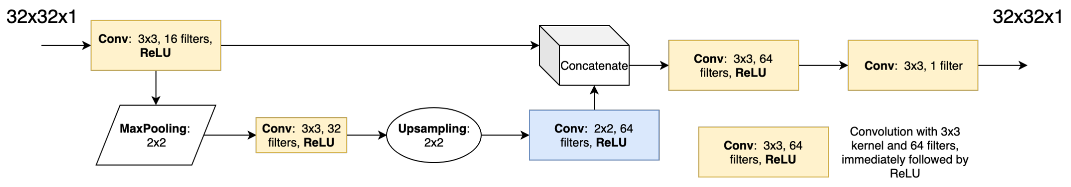
32x32x1
Conv: 3x3, 16 filters, ReLU
MaxPooling: 2x2
Conv: 3x3, 32 filters, ReLU
Upsampling: 2x2
Conv: 2x2, 64 filters, ReLU
Concatenate
Conv: 3x3, 64 filters, ReLU
Conv: 3x3, 1 filter
32x32x1
Conv: 3x3, 64 filters, ReLU
Convolution with 3x3 kernel and 64 filters, immediately followed by ReLU

Figure 12: Baseline denoising architecture.

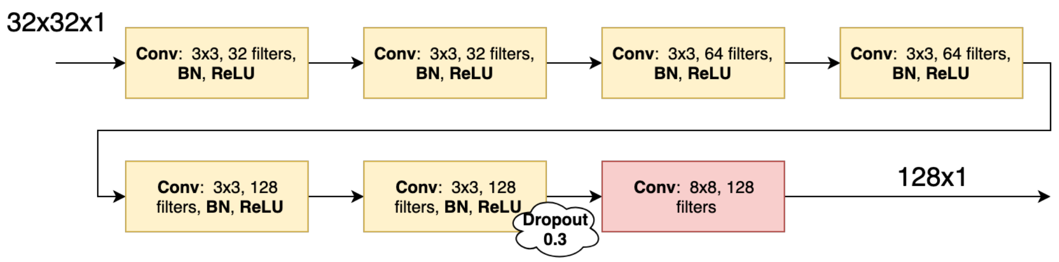
32x32x1
Conv: 3x3, 32 filters, BN, ReLU
Conv: 3x3, 32 filters, BN, ReLU
Conv: 3x3, 64 filters, BN, ReLU
Conv: 3x3, 64 filters, BN, ReLU
Conv: 3x3, 128 filters, BN, ReLU
Conv: 3x3, 128 filters, BN, ReLU
Dropout 0.3
Conv: 8x8, 128 filters
128x1

Figure 13: Baseline descriptor architecture.

Baseline Approach	**Balanced Variant [AUC]**		**Imbalanced Variant [mAP]**		**Improved Approach**	**Balanced Variant [AUC]**		**Imbalanced Variant [mAP]**	
Noise	**Inter**	**Intra**	**Inter**	**Intra**	**Noise**	**Inter**	**Intra**	**Inter**	**Intra**
Easy	0.929	0.897	0.828	0.742	**Easy**	0.944	0.914	0.852	0.770
Hard	0.910	0.870	0.761	0.646	**Hard**	0.926	0.889	0.795	0.683
Tough	0.880	0.834	0.678	0.551	**Tough**	0.898	0.853	0.713	0.584

Table 1: Patch verification results. *Balanced* or *Imbalanced* refers to whether the number of positive or negative samples is balanced or not respectively. *Inter* refers to a set of negative pairs which are sampled from images within the same sequence and *Intra* refers to samples from different sequences. *Easy*, *High* and *Tough* represent the level of distortion and noise applied.

Baseline Approach								**Improved Approach**							
	100	**500**	**1000**	**5000**	**10000**	**15000**	**20000**		**100**	**500**	**1000**	**5000**	**10000**	**15000**	**20000**
Easy [mAP]	0.745	0.609	0.552	0.434	0.386	0.361	0.345	**Easy [mAP]**	0.773	0.635	0.579	0.460	0.413	0.389	0.373
Hard [mAP]	0.675	0.488	0.412	0.264	0.215	0.190	0.175	**Hard [mAP]**	0.715	0.535	0.462	0.311	0.258	0.231	0.216
Tough [mAP]	0.574	0.355	0.278	0.148	0.111	0.094	0.084	**Tough [mAP]**	0.627	0.414	0.336	0.195	0.152	0.132	0.120
Mean [mAP]	0.665	0.484	0.414	0.282	0.237	0.215	0.201	**Mean [mAP]**	0.705	0.528	0.459	0.322	0.275	0.250	0.236

Table 2: Patch retrieval results. *Easy*, *High* and *Tough* represent the level of distortion and noise applied. The numbers in the horizontal row represent the size of the image-patch pool for retrieval.

Baseline Approach				**Improved Approach**			
Easy [mAP]	**Hard [mAP]**	**Tough [mAP]**	**Mean [mAP]**	**Easy [mAP]**	**Hard [mAP]**	**Tough [mAP]**	**Mean [mAP]**
0.213	0.079	0.028	0.107	0.227	0.098	0.041	0.122

Table 3: Image matching results. *Easy*, *High* and *Tough* represent the level of distortion and noise applied.

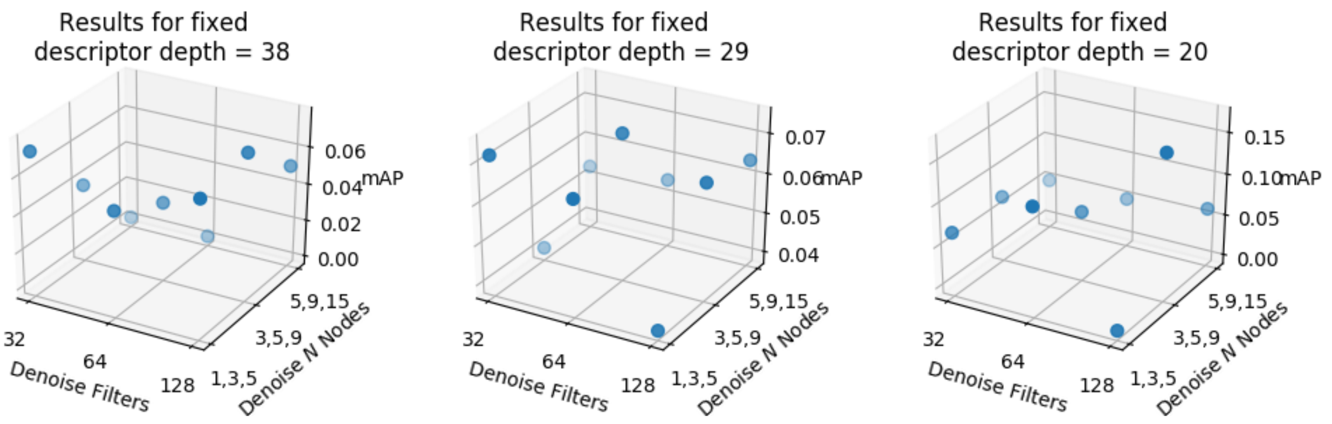

Figure 14: Patch verification results.

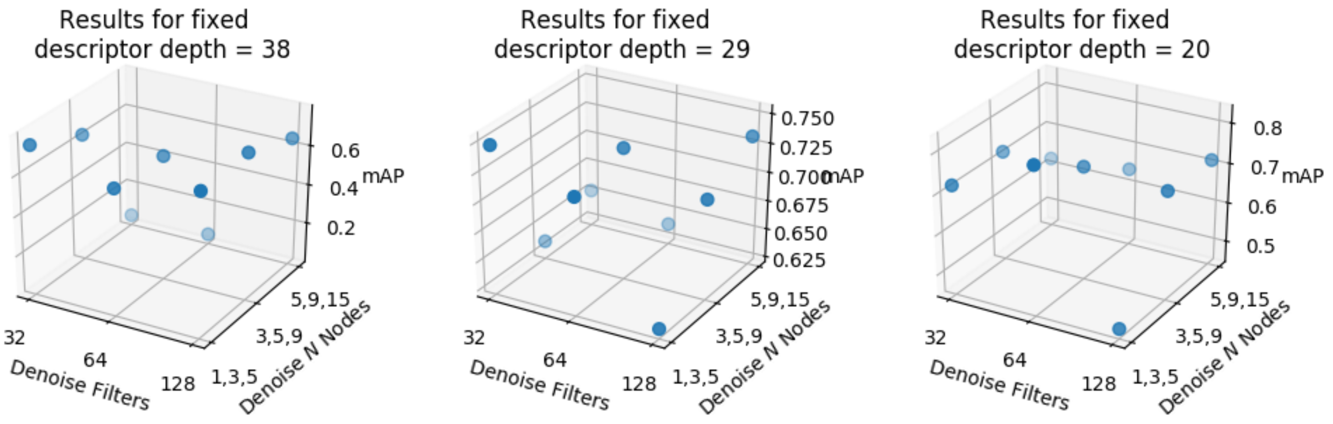

Figure 15: Image matching results.

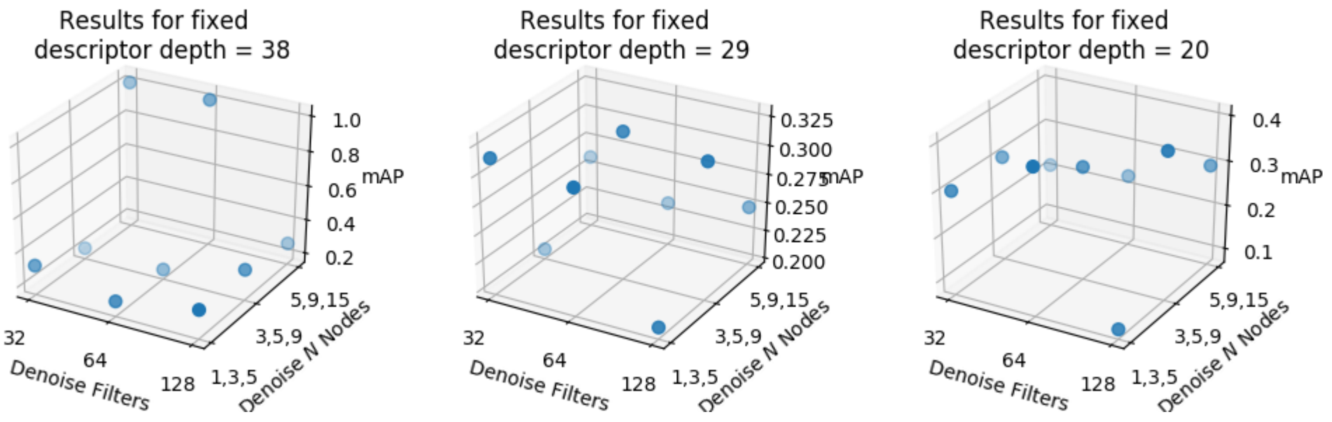

Figure 16: Patch retrieval results.