Deep Learning Report

Ricardo Vera - CID: 01061547
Electrical and Electronic Engineering Department
Imperial College London

Abstract

This report explores the image representation problem by means of descriptor vectors. A noisy version of the HPatches dataset [1] is used and evaluated using verification, matching and retrieval, according to the HPatches benchmark. A baseline approach is presented composed of a shallow UNet[8] and a L2-Net[10] descriptor network. This report then explores the various modifications to the baseline and evaluates their performance. The final proposed method involving a deeper denoiser and the modification of the loss function using softmax activation and harder triplet exposure, improves the baseline average mAP of 0.417 *to an improved average mAP of* 0.558.

1. Problem Formulation

The aim of the project is successful image representation. This involves developing a descriptor vector that best represents the N-HPatches dataset [5], using matching, verification and retrieval tasks as evaluation measures, using the HPatches benchmark [1]. Hence, the aim is to develop a network that is able to provide a satisfactory representation of the patches via a descriptor vector, with its input being an image (32×32) and output a descriptor vector (128×1).

The N-HPatches dataset is a noisy version of the HPatches [1] dataset with varying magnitudes of noise. The dataset comprises a set of a set of patch sequences varying in difficulty according to photometric or geometric variations, representing local sections of the original images. The additional noise added with respect to HPatches, adds an additional difficulty for the descriptor model, likely needing to include a denoising step. The patch sequences representing the images are split as $76 : 40$ for training and validation, this is done in order for validation to be a good test set representation and hence be appropriate for evaluation.

Image matching assesses the descriptor's ability to identify correspondences between two images. Retrieval tests how well a descriptor can match a query patch to a pool of patches extracted from many images. Patch verification measures the ability of a descriptor to classify whether two patches are extracted from the same measurement. These are further developed in the Appendix. The performance of the model is evaluated as the average Mean Average Precision (mAP), Equation 1, of the 3 tasks.

$$mAP = \frac{1}{|classes|} \sum_{c \in classes} \frac{\#TP(c)}{\#TP(c) + \#FP(c)} \quad (1)$$

$$TP = TruePositive \qquad FP = FalsePositive$$

Successful image representation should be able to extract the key features that describe the image, similarly to how the image itself does visually. The problem thus has a certain clustering consideration, where similarity can be measured as the distance between descriptor vectors. The objective can thus be posed as the minimisation of the triplet loss (Equation 2), which encourages bringing similar patches together and dissimilar ones apart by considering anchor (reference) patches to which compare. In Equation 2, $d_{\mathbf{A}}(\mathbf{x_i}, \mathbf{x_j})$ represents the euclidean distance of similar images and $d_{\mathbf{A}}(\mathbf{x_i}, \mathbf{x_k})$ the squared distance of dissimilar ones, whilst the subscript $+$ indicates the ReLu nature of the loss.

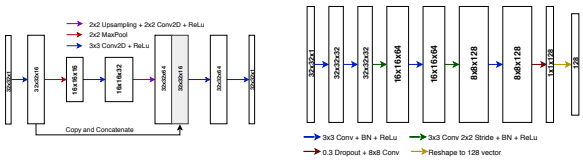
$$\text{Loss} = \sum_{\substack{i,j,k \\ y_i = y_j \neq y_k}} \Big[\alpha + d_{\mathbf{A}}(\mathbf{x_i}, \mathbf{x_j}) - d_{\mathbf{A}}(\mathbf{x_i}, \mathbf{x_k})\Big]_+ \quad (2)$$

Successful descriptor vector development will achieve clustering, thus leading to optimum results in all 3 evaluation tasks by tackling the common factor, vector distance.

2. Baseline Approach

The baseline model is composed of a two-stepped approach: using a shallower version of the UNet [8] for denoising, followed by a L2-Net [10] for descriptor vector generation.

The U-Net is a convolutional network developed for biomedical image segmentation with form of the letter U, following an autoenconder architecture. The denoiser implemented is composed of an encoder, composed of a convolutional and max-pooling layer; a bottleneck, composed of a convolutional layer; and a decoder, composed of 3 convolutional layers, with the first layer being preceded by an upsampling layer, and followed by a merging layer, merging the output of the encoder convolutional layer. Figure 1a.



(a) Denoise Net structure

(b) Descriptor Net structure

Figure 1: Denoise & Descriptor Nets (Larger in Appendix)

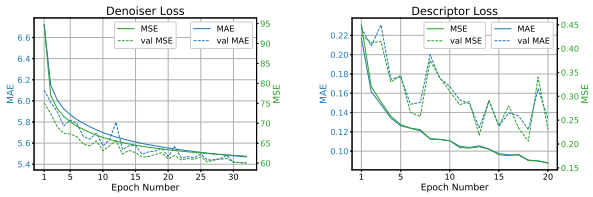
The implemented descriptor network replicates the L2-Net, Figure 1b, with a set of 7 convolutional layers with batch normalisation and ReLU activations, using strides of two alternatively for downsampling and a final stride of 8 to achieve the 128-dimensional vector after flattening. The triplet loss considers images in triplets (anchor, positive, negative) with $\alpha = 1$, minimising Equation 2 to develop descriptor vectors outlining the similarities and differences among images.

Performance

The denoiser network was trained for 32 epochs using *SGD* as an optimiser with learning rate $\mu = 0.00001$ and 0.9

nesterov momentum. The descriptor network on the other hand was trained for 20 epochs using SGD optimiser and $\mu = 0.1$. Both networks were compiled using MAE as distance loss, which deals better with outliers than MSE, hence achieving higher model stability, and weights initialised using the He normal initialiser.

The decreasing trend in denoiser validation loss, Figure 2a, suggests further training would result in better performance, showing how training and validation loss are very similar which suggests the network may be underfitting. A possible explanation for this is that not all relevant features are being extracted, thus losing information, whilst the single skip connection limits the amount of hard to recover information being passed, thus limiting the network's ability to generate the images back.



(a) Denoiser loss vs epochs (b) Descriptor loss vs epochs

Figure 2

From Figure 2b, it can be seen how even though training loss does decrease uniformly, validation loss plateaus early and remains level. This, combined with the large variance of the validation loss itself suggests an overfitting of the training data due to the strength of the descriptor network.

Overall the baseline approach achieved 0.76, 0.17 and 0.47 in verification, matching and retrieval mAPs respectively to give a final performance of 0.471.

Descriptor on noisy patches

By evaluating the performance of the descriptor network on the noisy patches we can assess the performance of the denoising network. As mentioned the descriptor network is a state-of-the-art model and so is very robust leading to overfitting in the baseline approach. By evaluating the descriptor network on the noisy patches directly the network would need to learn the denoising step itself and so the expected performance would decrease.

Upon training the descriptor network achieved very similar results as the baseline approach, Table 7 (Appendix), whilst the training curve showed how the validation loss didn't oscillate as much, Figure 9 (Appendix). This then reinforces our previous observations, suggesting both that the denoiser network is too shallow and that the descriptor network is considerably robust.

3. Proposed Improved Approach

As mentioned above the baseline approach suffers from certain issues such as overfitting or information loss in the denoising stage. To tackle these a set of methods are suggested for the improved approach. All proposals are compared against the baseline method for the same number of epochs (32 denoiser, 20 descriptor network), to directly compare if there is any improvement.

The first consideration to make is the choice of **optimiser**. With models exposed to settling in a local minima the choice of optimiser affects both the performance and convergence speed of the network. The *Adam* optimiser presents a state-of-the-art optimiser which adapts the learning rate over time by combining the benefits of AdaGrad and RMSProp optimisers. This benefits have led *Adam* to become a very popular optimiser, due to its ability to achieve better results, as show in the original paper [4]. Under the same training setup as the baseline, *Adam* did not improve significantly performance, Table 7 (Appendix), the increased stability in validation loss however, Figure 10 (Appendix), supports the choice of *Adam* over *SGD*. Additionally MAE was again used to provide a fair comparison with the baseline, and specifically chosen due to its robustness to outliers.

The effect of **alpha** and **dropout** on the descriptor network were explored, however were found to be optimally set as per the baseline, $\alpha = 1$ and no dropout. These analyses are described in the appendix.

Denoiser

The first major change proposed to the baseline approach is the modification of the denoiser network. As mentioned the shallow denoiser network leads to loss of information in the denoising process. To tackle this, the network can be extended to a higher degree of the original implementation Figure 7 (Appendix) into a much deeper network which can properly denoise the patches.

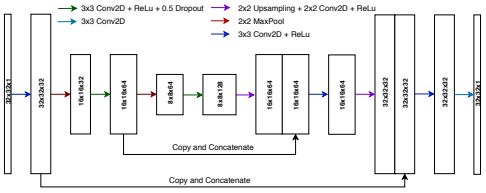


Figure 3: UNet implemented architecture

The first implemented denoiser network, Figure 3, is composed of a set of convolutional, max-pooling and ReLU activation layers, with dropout layers in alternating decoding part and upsampling in the encoding part. The architecture is a shallower version of the original UNet implementation [8], adapted for input dimension difference. The 2 skip connections additionally prevent information to be lost through the denoising network.

The UNet implementation can be however modified to achieve a more stable network with easier flow of information. Based on the structure of GANs, more specifically the DCGAN [7], the maxpooling layers are substituted by convolution layers with 2×2 strides and the upsampling layers are substituted by transposed convolutions of stride 2×2. The use of strided convolutional layers over maxpooling allows for the spatial consideration needed for downsampling being learnt. Additionally, the use of transposed convolutions over upsampling layers allows for a better upsampling

procedure avoiding sparse gradients and aiding in training stability. As such Figure 3 is modified into Figure 11 (Appendix) to avoid the loss of information.

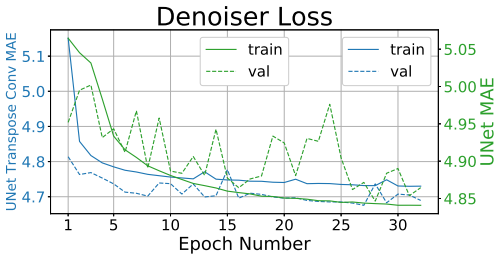


Figure 4: Loss for both UNet implementations

Figure 4 shows the training and validation losses for both implementations under same baseline training conditions, except using *Adam* optimiser. The validation loss specially shows a very clear improvement using the transpose convolution implementation in terms of loss and stability. Both networks outperform considerably the baseline denoiser, however the first UNet implementation does suffer from training instability. The deeper network allows for better validation performance, however the presence of transpose convolution and strided convolutions allows information to be passed on better than using maxpooling and upsampling. It is clear how the second implementation stabilises quickly and gives an optimum validation loss for ~ 27 epochs.

Descriptor

As mentioned, to provide a fair comparison with the baseline, training conditions were kept equal, and the baseline denoiser was used for evaluation.

The effect of the **batch size** can affect greatly the descriptor network performance as the amount of triplets the network sees per batch will affect how good the updates are. Large batch size is known to converge to sharp minimisers characterised by large eigenvalues, which tend to generalise less well [3]. However, too small of batch size can affect the exposure of the network towards hard triplets, specially affecting the ability to learn hard negatives and positive differences [6].

Batchsize	Verification	Matching	Retrieval	Average
50	0.76	0.17	0.47	0.47
100	0.79	0.19	0.49	0.49
200	0.78	0.19	0.48	0.48
500	0.79	0.18	0.46	0.47
1000	0.77	0.15	0.43	0.45

Table 1: Performance for varying batchsize

Under the same baseline training conditions, Figure 13b (Appendix) shows how it is hard to decide upon batchsize due to the large oscillations through training. It can be seen though, how batchsizes of $B = 100, 200$ give best results if selected at the appropriate epoch, with the baseline batchsize $B = 50$ giving better results for lower epoch number. This was backed up by performance Table 1. These results highlight the trade-off mentioned between generalisation and triplet exposure with the lower baseline $B = 50$ struggling at larger epoch numbers.

The choice of **activation function** in the triplet loss determines the loss propagation throughout the network, thus non-linear functions can provide more complex ways of optimising the loss. The baseline descriptor network has a triplet loss defined by the ReLu activation function (ie.$\max(0, x)$), however different activations can be applied to improve the network's performance. The different activations explored were ReLU, ELU and softplus, Figure 14a (Appendix). ELU is an extension of ReLu with $e^x + 1$ for negative values, reducing the loss for triplets achieving the margin. Softplus however offers a smoother behaviour, $y = \log(1 + e^x)$, penalising even the triplets that slightly achieve the margin.

Activation	Verification	Matching	Retrieval	Average
ReLU	0.76	0.17	0.47	0.47
ELU	0.78	0.17	0.47	0.48
Softplus	0.82	0.22	0.52	0.52

Table 2: Performance for different activations

The activation functions change the loss function, requiring comparison to rely on task performance. From Table 2 it can be seen how *softplus* outperforms significantly the other activations. By providing a smoother loss around zero, loss can propagate through the network even when the margin is fulfilled, leading to an improved learning by the network.

Based on [11] an **in-triplet hard negative mining** is possible by swapping the anchor and positive of the triplet. The procedure calculates the distance between positive and negative and compares that to the anchor-negative distance, if the new distance is larger the anchor and positive swap roles thus achieving a harder negative due to the larger distance with the negative sample, Figure 14b (Appendix). This allows for the network to learn on harder triplets which has been shown to improve performance [2] by forcing the network to separate the examples breaking the margin α.

Activation	Verification	Matching	Retrieval	Average
ReLU	0.83	0.24	0.55	0.54
ELU	0.81	0.21	0.52	0.51
Softplus	0.84	0.26	0.55	0.55

Table 3: Performance for different activations

As can be seen from Table 3 hard negative mining provides a significant improvement in performance with *softplus* achieving 0.548 average mAP, although ReLU gave the largest improvement from 0.471 to 0.540. The ability to learn from hard triplets makes the network more selective, extracting only the key features to differentiate among similar patches, thus boosting its performance.

An additional **regularisation** parameter is explored to consider the geometric alignment of the triplet. By considering the actual spatial relation of the triplet, in the form of its **colinearity**, a regularisation factor θ, can be used to account for the angle between the distance vectors. A larger angle means the triplet is less relevant, whilst a smaller one makes

the triplet highly relevant, Figure 6 (Appendix). As such the Equation 2 is modified to Equation 3. The rational behind the regularisation is further developed in the Appendix.

$$\text{Loss} = \sum_{\substack{i,j,k \\ y_i=y_j\neq y_k}} \Big[\alpha + d_{\mathbf{A}}(\mathbf{x_i}, \mathbf{x_j}) - d_{\mathbf{A}}(\mathbf{x_i}, \mathbf{x_k}) + \beta\theta\Big]_+ \quad (3)$$

β	Verification	Matching	Retrieval	Average
0.01	0.83	0.24	0.55	0.54
0.05	0.82	0.22	0.53	0.52
0.1	0.84	0.25	0.56	0.55
0.5	0.82	0.23	0.53	0.53
1	0.83	0.23	0.53	0.53
2	0.0.83	0.24	0.54	0.53

Table 4: Performance for varying β

Under the same conditions as for hard negative mining, Table 4 shows the performance of varying the hyperparameter β. The optimum β was found to be 0.1, improving the above hard negative mining procedure. By considering the angle of the triplet, the network is able to weigh the losses accordingly and hence learn the more relevant examples.

A common issue with descriptor networks is its variance against data transformations. If the network only sees the images/patches in a certain environment it will be fooled by simple data transformations. In order to develop robust descriptor vectors the network should be able to represent patches independent of transformations such as rotation or translation. In order to do this **data augmentation** can be explored, performing transformations on the training data provides a means to *acquire* new data with which train the network, thus making it more robust.

By performing random horizontal/vertical flips and random rotations in range $0, 90°$, the method however didn't yield any improvement in performance. Figure 16 (Appendix) shows how validation loss oscillates considerably and the performance was actually lower with average mAP of 0.397, Table 7 (Appendix).

VGG

From the observed descriptor network results on noisy data, an end-to-end approach is possible by modifying a pretrained network for the task in hand. The VGG network [9] is an architecture presented by Simonyan and Zisserman, composed of 3×3 convolutional and 2×2 maxpooling filters widely used for image classification. By loading the pretrained network and using only certain layers, additional layers can be added to adapt to the image representation task, Figure 17a (Appendix).

Under the same training conditions as the baseline, the VGG network slightly improved performance of the descriptor network on the noisy patches but failed to improve the denoiser-descriptor basic approach, achieving an average mAP of 0.466. This suggests the denoiser-descriptor model is a more appropriate choice for the given task. Figure 17b shows how the validation is more stable but remains at higher values than the denoiser-descriptor network.

Final Improved Approach

The final proposed approach is a merge of the observed best techniques. Both the denoiser and descriptor networks are trained with the *Adam* optimiser and the shallow baseline denoiser is substituted by the deeper UNet with strided and transposed convolutions.

The descriptor structure remains untouched, the L2Net is a robust state-of-the-art architecture that manages to extract the key features for image representation. Triplet loss margin, α, is set equal to 1 providing a way to separate classes whilst aiming at hard triplets, and batchsize is set to 100 to provide a way of learning from larger hard triplet exposure whilst maintaining generalisation.

The triplet loss is further modified to boost the network's performance form higher exposure to harder triplets. Activation function is changed from *ReLU* to *softplus*, providing a smoother function which allows easier gradient flow along the network, and the in-triplet hard negative mining procedure is performed to increase hard triplet exposure. Finally, the mentioned angle regularisation parameter is applied to account for colinearity considerations, with $\beta = 0.1$ for best performance.

Validation loss of the model oscillates considerably, however, picking the best validation loss result leads to the best results. As mentioned the UNet with transpose convolution minimises validation loss, Figure 12b (Appendix) far beyond baseline results, Figure 2a, achieving a minimum validation loss of 4.68 compared to the baseline's 5.41. This aids the descriptor loss which along with the mentioned improvements manages to stabilise validation loss to a certain extent, Figure 18, and improve overall performance.

Difficulty	Verification	Matching	Retrieval	Average
Easy	0.91	0.43	0.69	0.67
Medium	0.86	0.24	0.57	0.56
Hard	0.78	0.11	0.43	0.44
Average (3 d.p.)	0.853	0.257	0.563	0.558

Table 5: Performance for easy, medium and hard patches of final approach

Table 5 shows the full breakdown of the final approach performance. This method achieves a considerable increase from the baseline approach after training the denoiser and descriptor network for 32 and 20 epochs respectively. Picking the models from best validation losses during training (denoiser:27, descriptor net:16 epochs), shows the improvement of the proposed method with respect to the baseline's average mAP 0.471, achieving a final average mAP of 0.558.

4. Conclusion

This report has aimed at presenting the image representation task and the various ways to tackle it. The baseline method is presented as a shallow denoiser and a descriptor network achieving average mAP 0.471 of the 3 tasks. Various methods are explored to improve the baseline, making the denosier deeper or adjusting the triplet loss to account for harder triplets, from which the proposed improved solution is achieved with an average mAP of 0.558 in the 3 tasks.

References

[1] V. Balntas, K. Lenc, A. Vedaldi, and K. Mikolajczyk. Hpatches: A benchmark and evaluation of handcrafted and learned local descriptors. pages 3852–3861, 07 2017.

[2] A. Hermans, L. Beyer, and B. Leibe. In defense of the triplet loss for person re-identification, 2017.

[3] N. S. Keskar, D. Mudigere, J. Nocedal, M. Smelyanskiy, and P. T. P. Tang. On large-batch training for deep learning: Generalization gap and sharp minima. *CoRR*, abs/1609.04836, 2016.

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

[5] K. Mikolajczyk, S. Iodice, A. Lopez Rodriguez, and A. Barroso Laguna. keras triplet descriptor. `https://github.com/MatchLab-Imperial/keras_triplet_descriptor`, 2019.

[6] A. Mishchuk, D. Mishkin, F. Radenovic, and J. Matas. Working hard to know your neighbor's margins: Local descriptor learning loss. In *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, 4-9 December 2017, Long Beach, CA, USA*, pages 4829–4840, 2017.

[7] A. Radford, L. Metz, and S. Chintala. Unsupervised representation learning with deep convolutional generative adversarial networks. *CoRR*, abs/1511.06434, 2015.

[8] O. Ronneberger, P. Fischer, and T. Brox. U-net: Convolutional networks for biomedical image segmentation, 2015.

[9] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. *CoRR*, abs/1409.1556, 2014.

[10] Y. Tian, B. Fan, and F. Wu. L2-net: Deep learning of discriminative patch descriptor in euclidean space, 2017.

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

Appendix

Evaluation tasks

Extracted from the HPatches paper [1].

Patch Verification evaluates if two patches are in correspondence or not. Given a list $\mathcal{P} = ((\mathrm{x_i}, \mathrm{x_i'}, y_i), i = 1, ...N)$ of positive and negative patches, where $\mathrm{x_i}$ are 32×32 patches and y_i is the label where $y_i = \{-1, 0, 1\}$ indicating negative, *to ignore* and negative match. The dataset is then used to evaluate a matching procedure $\mathcal{A}$ to produce a confidence interval on the correspondence of two given patches. The quality of the approach is measured by the average precision of the ranked patches.

Image Matching consists of descriptors being used to match patches from a reference image to a target one. Representing an image L_k as a collection of patches $L_k = (x_{ik}, i = 1, ...N)$. If we consider two images, reference L_0 and target L_1, after matching, x_{i0} is in correspondence with x_{i1}. The pair L_0, L_1 is then used to evaluate an approach $\mathcal{A}$, where given a reference batch x_{i0} the method evaluates the target index $x_{i\sigma}$. The labels are then assigned $y_i = \pm 1$ according to the success/failure of the method $\mathcal{A}$. The approach is then again evaluated using the average precision measure.

Patch Retrieval involves descriptors being used to find patch correspondences in a large collection of patches, a large portion of which are distractors, extracted from confounder images. Given a portion $\mathcal{P} = (\mathrm{x_0}, (\mathrm{x_i}, y_i), i = 1, ...N)$ with a query patch x_0 from the reference image L_0 and all patches from images $L_k, k = 1, ..K$ as well as confounder images. Positive labels, $+1$, are given to corresponding patches to the query patch, resulting in a maximum of K corresponding patches (K images). However, retrieved patches that do not correspond to the query patch but at least belong to a matching image are ignored($y_i = 0$). The approach $\mathcal{A}$ is then evaluated and average precision is again used for quantitative evaluation.

Triplet margin α

One of the parameters explored to improve the descriptor network is the margin α in the triplet loss. The value of alpha will determine the loss that is fed into the network during backpropagation and so will directly affect its learning. As such a low alpha will focus on the worst case triplets where the positive is further away from the anchor than the negative, but will fail to separate by classes overall. A large α will ideally separate classes by the margin, however due to large loss, the model is likely to improve on the easy (already separated) triplets by pushing negatives further and bringing positives closer, without improving the actual hard triplets.

α	Verification	Matching	Retrieval	Average
0	0.54	0.07	0.31	0.31
0.5	0.76	0.13	0.41	0.43
1	0.76	0.17	0.47	0.47
2	0.79	0.17	0.45	0.47
3	0.78	0.16	0.45	0.46

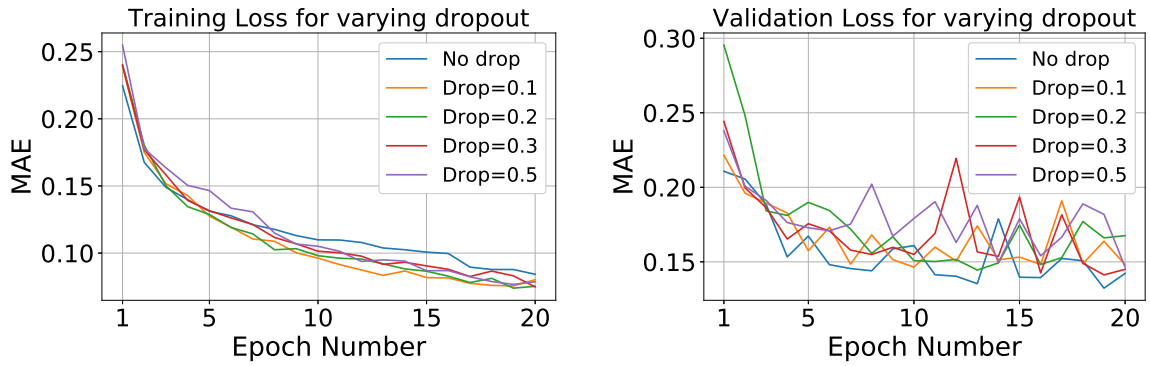
Table 6: Performance for varying α

Under the same baseline training conditions, Table 6 shows how setting $\alpha = 0$ makes performance collapse, but that too

high alpha also affects overall performance. Setting $\alpha = 1,2$ provides the perfect trade-off between class separation and hard triplet separation, suggesting the baseline approach already had the optimum α value.

Dropout

As mentioned in Section 2, the descriptor network tends to overfit the data leading to unstable training and the resulting validation oscillations shown in Figure 2b. Adding dropout layers to the descriptor network helps stabilise training by randomly ignoring (dropping) certain layer output making an apparent different layer structure. As a result, generalisation is improved by providing a different *view* during training.



(a) Train loss varying dropout

(b) Val loss varying dropout

Figure 5: Train and Validation loss for varying dropout

Under the same baseline training conditions, Figures 5a and 5b show how dropout can lead to lower training error, however do not improve network performance. For varying dropout no significant stability gain is observed and the lowest validation loss is actually obtained for no dropout. 0.2 and 0.3 outperform 0.1 and 0.5 which can be seen as too little or too much dropout, however 0 dropout seems to be the most stable and give the lowest validation loss, suggesting for this task dropout doesn't improve performance.

Triplet regularisation according to angle

The angle between the anchor-positive vector and the anchor-negative vector describes how colinear the positive and negative are with respect to the anchor. The more colinear they are the more relevance as the distance directly shows how well patches are being separated. If the angle is large however, the anchor is effectively in between the positive and negative and so the distance difference doesn't entail as much information about patch separation, Figure 6. Hence by performing the inner product of both vectors we are effectively calculating $\cos\theta$, which varies between $-1,1$, giving a measure of spatial relation. θ is basically the inner product of the normalised distance vectors, Equation 4.

$$\theta = (\mathbf{x_i} - \mathbf{x_j}) \bullet (\mathbf{x_i} - \mathbf{x_k}) \tag{4}$$

As shown in Figure 6, the figure on the left shows a triplet were the positive and negative are highly colinear, small θ, whilst the one on the right shows a triplet with low positive-negative colinearity. Even though the distanced are similar, the left situation provides much more information about the separation, $\theta_1 < \theta_2$, and so performance and so is weighed accordingly.

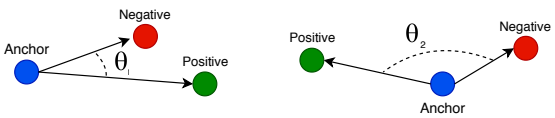


Figure 6: Comparison of high and low information according to angle θ

The procedure involves the calculation of the distance vectors followed by normalisation. The inner product between them is then performed to obtain our *angle* indicator between $-1,1$. If $\theta = 0°$ the inner product is 1, whilst if $\theta = 180°$ the inner product is -1, thus giving the perfect relation of angle and amount to regularise. This indicator is then used as the regularisation factor weighed by the hyperparameter β. This results in Equation 5

$$\text{Loss} = \sum_{\substack{i,j,k \\ y_i = y_j \neq y_k}} \Big[\alpha + d_{\mathbf{A}}(\mathbf{x_i}, \mathbf{x_j}) - d_{\mathbf{A}}(\mathbf{x_i}, \mathbf{x_k}) + \beta\theta\Big]_+ \tag{5}$$

Figures

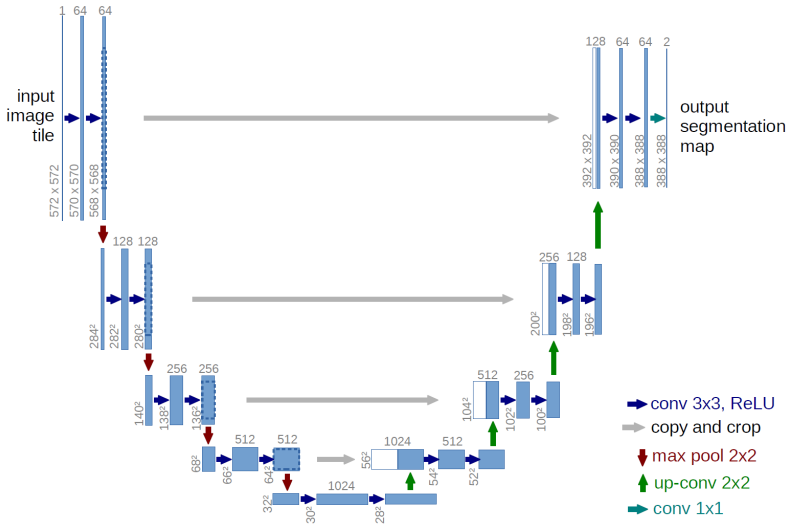
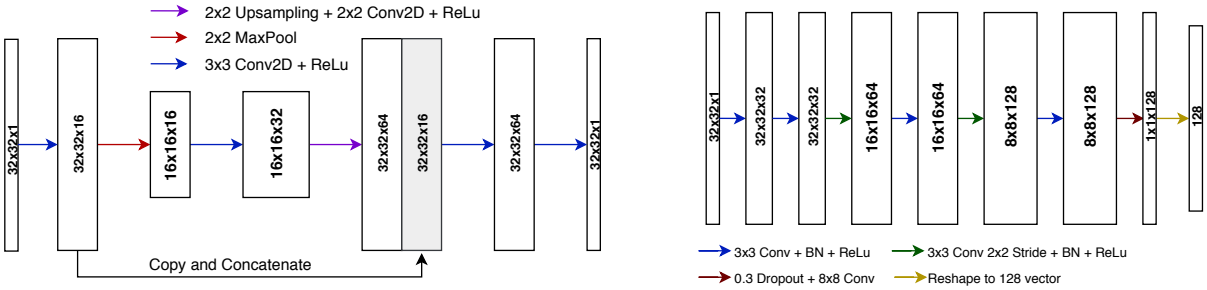


Figure 7: UNet original architecture [8]



(a) Denoise Net structure (for visualisation)

(b) Descriptor Net structure (for visualisation)

Figure 8: Denoiser and Descriptor Networks

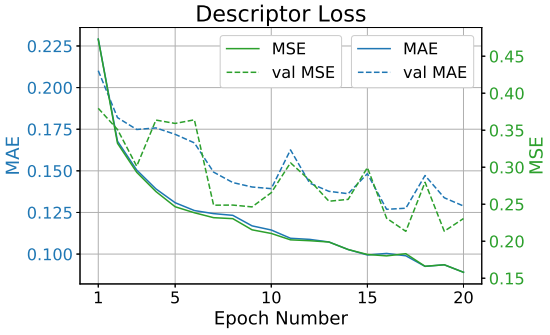


Figure 9: Baseline descriptor loss on noisy patches

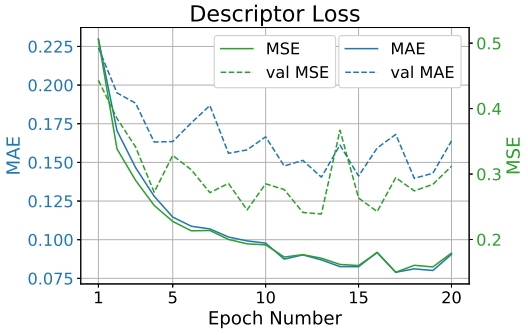


Figure 10: Baseline descriptor loss with *Adam* optimiser

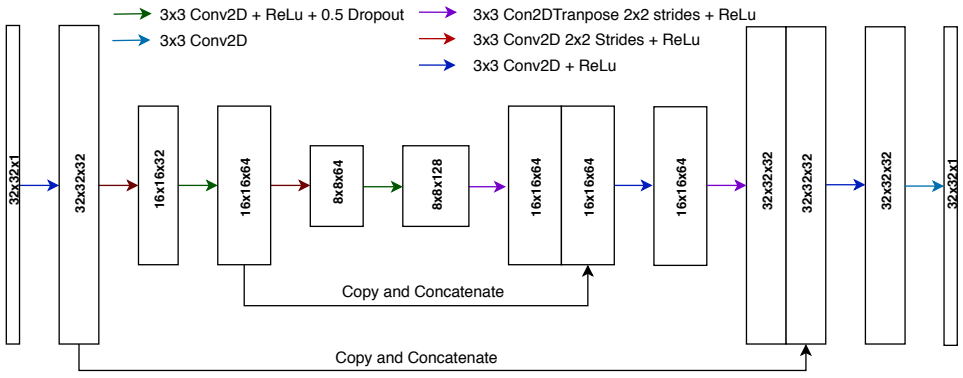
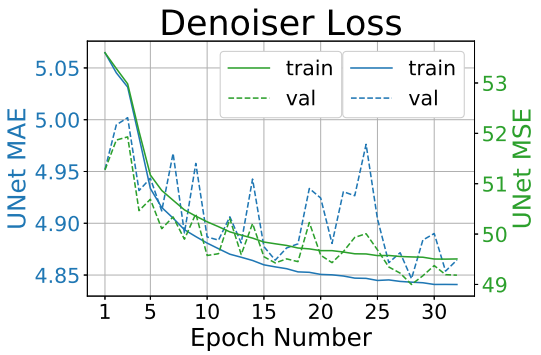
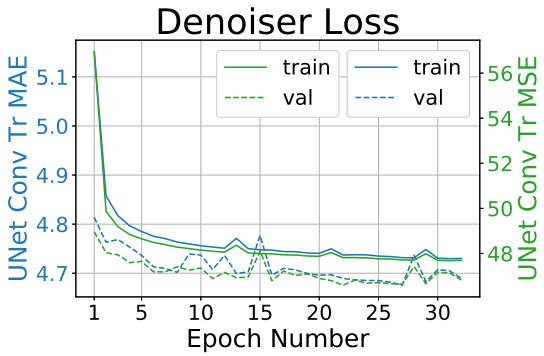


Figure 11: UNet with Transposed convolution implemented architecture

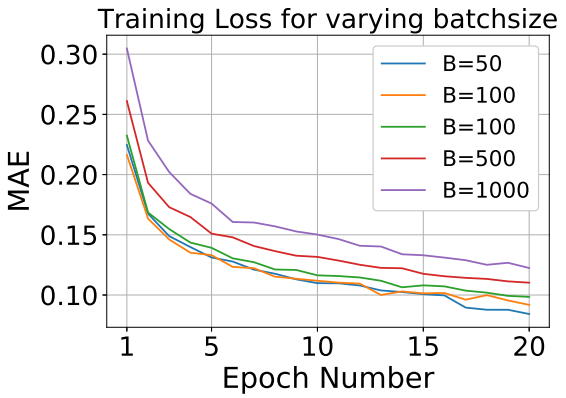


(a) Implemented UNet Loss

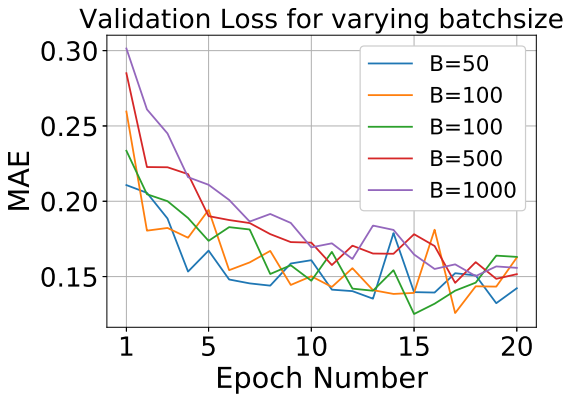


(b) Implemented UNet with Transposed convolution Loss

Figure 12: Losses for implemented UNets

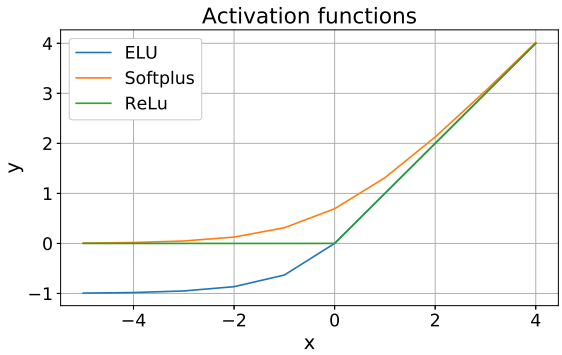


(a) Train loss varying batchsize



(b) Validation loss varying batchsize

Figure 13: Train and Validation loss for varying batchsize



(a) Activation functions

(b) In-Triplet Hard Negative Mining

Figure 14

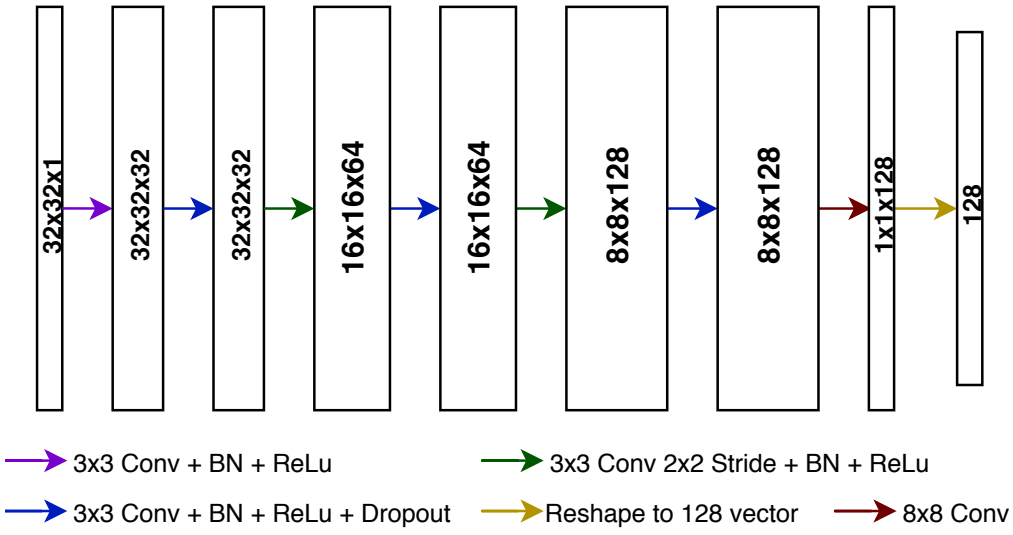


Figure 15: Descriptor structure with Dropout layers

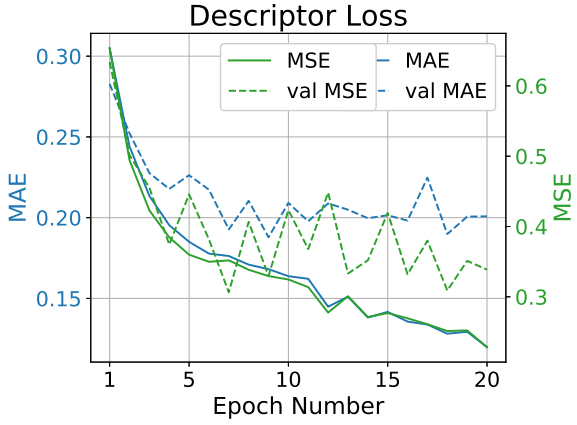
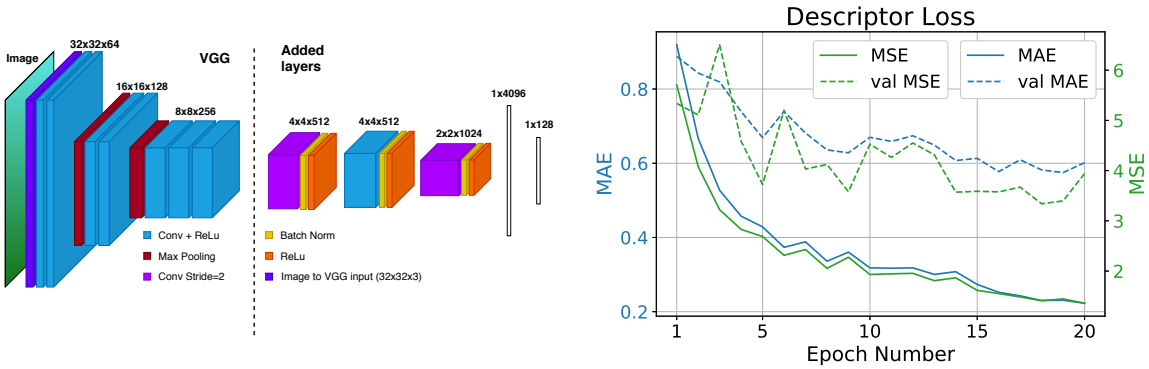


Figure 16: Training and Validation loss using data augmentation



(a) Implemented VGG network

(b) Training and Validation loss for implemented VGG

Figure 17: VGG architecture and Loss

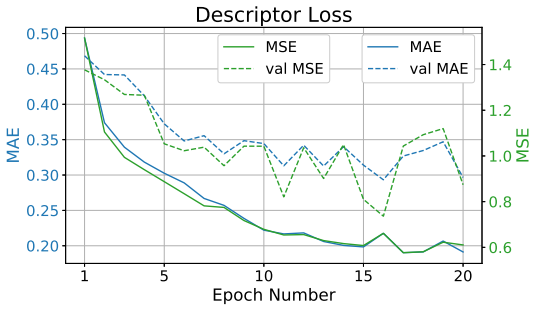


Figure 18: Training and Validation loss for proposed improved approach

Setting	Verification (2 d.p.)	Matching (2 d.p.)	Retrieval (2 d.p.)	Average (3 d.p.)
Baseline	0.76	0.17	0.47	0.471
Descriptor on noisy patches	0,76	0,16	0,44	0,452
Adam	0.77	0.16	0.46	0.473
Unet + base descriptor	0.79	0.17	0.48	0.480
UNet Transpose Conv + base descriptor	0.78	0.19	0.48	0.483
$\alpha = 0$	0.54	0.07	0.31	0.310
$\alpha = 0.5$	0.76	0.13	0.41	0.432
$\alpha = 1$ (base)	0.76	0.17	0.47	0.471
$\alpha = 2$	0.79	0.17	0.45	0.471
$\alpha = 3$	0.78	0.16	0.45	0.465
batchsize=50 (base)	0.76	0.17	0.47	0.471
batchsize=100	0.79	0.19	0.49	0.493
batchsize=200	0.78	0.19	0.48	0.485
batchsize=500	0.79	0.18	0.46	0.476
batchsize=1000	0.77	0.15	0.43	0.447
Dropout=0 (base)	0.76	0.17	0.47	0.471
Dropout=0.1	0.76	0.18	0.47	0.471
Dropout=0.2	0.78	0.18	0.46	0.472
Dropout=0.3	0.79	0.18	0.47	0.481
Dropout=0.5	0.77	0.16	0.46	0.462
ReLu (base)	0.76	0.17	0.47	0.471
ELU	0.78	0.17	0.47	0.478
Softplus	0.82	0.22	0.52	0.520
Hard Negative Mine ReLu	0.83	0.24	0.55	0.540
Hard Negative Mine ELU	0.81	0.21	0.52	0.510
Hard Negative Mine Softplus	0.84	0.26	0.55	0.548
$\beta = 0.01$	0.83	0.24	0.55	0.540
$\beta = 0.05$	0.82	0.22	0.53	0.52
$\beta = 0.1$	0.84	0.25	0.56	**0.551**
$\beta = 0.5$	0.82	0.23	0.53	0.525
$\beta = 1$	0.83	0.23	0.53	0.530
$\beta = 2$	0.0.83	0.24	0.54	0.535
Baseline with data augmentation	0.74	0.08	0.36	0.397
VGG	0.81	0.13	0.45	0.466

Table 7: Performance for all different implementations

Code Instructions

Baseline Code Colab link
Improved Approach Code Colab link
Zip File

For all files, specify root directory when mounting drive so that file are saved/loaded properly. When executing denoiser/descriptor training two cells are present before.

If training an already executed (and saved) model please specify name of file to be loaded (set default as name='model'+*executedEpochs*'.h5), if new model being trained execute cell with *loadedEpochs*=0 and epochs set to how many epochs wished.

For the The zip file consists of 3 files: baseline, improved and readdata file:

- Baseline: This file contains the baseline code untouched, with only slight modifications for saving files, and specific areas indicating were to change parameters, alpha, batchsize and activation function.
- Improved: This file contains all implementations of all different variations with option to vary alpha, batchsize and activations. Models included are
  - baseline denoiser
  - UNet deeper implementation
  - UNet deeper implementation with strided and transposed convolutions
  - baseline descriptor
  - baseline descriptor with dropout
  - VGG implementation
  - base triplet loss
  - triplet loss with in-triplet hard negative mining implemented
  - triplet loss with colinear approach implemented
  - The file also has an option to perform data augmentation. When cloning the repository uncomment my personal repository and then specify training generator transform to True. Indicated in code
- readData file : just as a check up of the data augmentation modifications to the original repository, this file basically has the transform implemented and should be included in the repository for operation.

For a follow through of the code the baseline file contains full comments (text cells) on the significance of each section, the improved file does not include such text but follows the same general structure.