EE3-25 Deep Learning Interim Report for Denoising and Descriptor Network Optimisation

Archit Sharma
AS10216
01199766
as10216@ic.ac.uk

Abstract

This report investigates using deep learning to allow computers to represent images. There are two stages to this: denoising the images and then generating a descriptor from the denoised image. A baseline approach used a CNN for each task, specifically U-Net [6] for denoising and HardNet [4] for generating descriptors. It was tested by matching noisy 32x32 patches from HPatches [1]. Using the HPatches benchmark, it scored 0.194049, 0.336016 and 0.663329 in matching, retrieval and verification respectively. By using DnCNN [11] for denoising and altering the HardNet implementation, the scores were improved to 0.226501, 0.516589 and 0.811292 for matching, retrieval and verification.

1. Introduction

Task: Denoising a patch and generating a descriptor to create a ranking of matches by $\mathcal{L}_2$ loss. Similar patches should have a short Euclidean distance between their descriptors and otherwise for dissimilar patches. The Euclidean distance between two descriptors a and b is given by Eq. (1).

$$||\mathbf{a}-\mathbf{b}|| = \sqrt{||\mathbf{a}||^2 + ||\mathbf{b}||^2 - 2\mathbf{a}\cdot\mathbf{b}} \tag{1}$$

Data Set Characteristics: $32\times32\times1$ tensor of elements x where $x < 256$. The input data set is `HPatches` [1], containing 116 sequences. Each sequence has a reference patch and 5 transformed patches based on the reference. 57 of the sequences are photometric transformations and the other 59 are geometric. Resulting descriptors have 128 parameters.

Testing: There are three evaluation tasks given by `HPatches-benchmark` [1]:

- **Verfication:** classifying if two images are similar or dissimilar
- **Matching:** finding similar patches to a query from a small gallery with difficult distractors
- **Retrieval:** finding similar patches to a query from a large gallery

The proposed model is then evaluated against these tasks by the metric mean average precision (mAP) with labels for each patch in the returned list to a query given by $\mathbf{y} = (y_1, \ldots, y_n)$, where $y_i \in \{-1, 0, +1\}^n$. This differs from standard mAP because entries of the list can be ignored if $y_i = 0$.

2. Base Implementation

2.1. Network Architecture

The base implementation consists of two networks back-to-back, a denoiser network the output from which feeds into a descriptor network. The denoiser is of a shallow U-Net [6] structure (Figure 2) and uses an SGD optimizer with $\eta = 0.00001$, $\alpha = 0.9$ and Nesterov momentum is applied. The loss function used is Mean Absolute Error (MAE), ReLU is the activation function and the weight initialisation is He Normal.

The baseline descriptor network is a HardNet [4] based on L2-Net [8] based on (Figure 1) using triplet loss along with MAE to generate an image descriptor with 128 parameters. Triplet loss [7] means the network takes in three inputs: an anchor (A), a positive (P) similar to A and a negative (N) dissimilar to both A and P. It then minimises the Euclidean distance between A and P while maximising the distance from A to N. The loss function is shown in Eq. (2), where $\mathrm{f}(X)$ denotes a descriptor of the image X and α is a margin. The optimizer used is SGD with $\eta = 0.1$, the activation function is ReLU and the weight initialisation used is He Normal. Batch normalisation is applied at every convolutional layer except the full-connected layer.

$$\mathcal{L}(A,P,N) = \max(||\mathrm{f}(A)-\mathrm{f}(P)||^2 - ||\mathrm{f}(A)-\mathrm{f}(N)||^2 + \alpha, 0) \tag{2}$$

2.2. Results & Evaluation

The baseline was trained over 10 epochs with a batch size of 50 from a subset of the full training data, with a training/validation split of 3/1 for the denoiser and 10/1 for the descriptor. The denoiser and descriptor networks were trained separately and then evaluated on matching, retrieval and verification, the results of which were 0.134049, 0.366016 and 0.693329 respectively and achievin a total score of 1.193394. When compared to results found in `HPatches`[1], retrieval and verification were about average, while matching was below average. Positively, the time taken to train the baseline was 201s.

3. Proposed Improvements

3.1. Architecture

Three alternative denoiser architectures were considered: a deeper U-Net, an autoencoder and DnCNN [11]. The architecture of the new 16-layer U-Net is shown in Fig. 3. The extra layers create more feature maps, allowing the network to better represent objects due to the increased levels of abstraction between the input and the output at the expense of taking more time per epoch to train and needing to train over more epochs. Also tested was a 10-layer autoencoder (Fig. 5. The structure is similar to U-Net, minus the merge layers within U-Net. The final architecture tested was a 17-layer DnCNN [11] [2], which finds a residual map and then subtracts it from the input. (i.e. instead of $y = F(x)$, $y = x + R(x)$).

These three networks were tested with the baseline descriptor network on `HPatches Benchmark` after being trained over 10 epochs. All three networks used the same optimizer, activation function, loss and training as the baseline denoiser, described in Section 2.1 and 2.2. The only difference is DnCNN uses orthogonal weight initialisation while U-Net and the autoencoder use He Normal. The results in Table 1 show DnCNN performing the best with a combined score of 1.24071 so it shall be the architecture explored further.

Architecture	Verification	Matching	Retrieval	Total Score
Baseline	0.663329	0.094049	0.336016	1.093394
16-layer U-Net	0.090627	0.000923	0.999666	1.091216
10-layer autoencoder	0.724425	0.086342	0.337832	1.148599
17-layer DnCNN	0.740663	0.122394	0.377653	1.24071

Table 1. `HPatches benchmark` scores with different denoiser architectures and the baseline descriptor

ResNet-50 [3] was considered to replace the descriptor network as it contains residual blocks to prevent vanishing gradients and allow inputs to be easily passed through a block without being transformed. The implementation used [5] without the final `softmax` layer and was tested against the baseline by training over 10 epochs with a DnCNN denoiser, using the same optimizer, activation function, loss, weight initialisation and training as the baseline descriptor discussed in Section 2.1 and 2.2. The results in 2 show that though the retrieval score was perfect, the comparatively low matching and verification score means that baseline HardNet will be explored further. The model used for these tests are improved incrementally as subsequent tests are carried out. A greedy approach is used instead of an exhaustive approach due to time limitations.

Architecture	Verification	Matching	Retrieval	Total Score
Baseline	0.663329	0.194049	0.336016	1.193394
ResNet-50	0.088392	0.000921	1.000000	1.089313

Table 2. `HPatches benchmark` scores with different descriptor architectures and the baseline denoiser

3.2. Loss Functions

DnCNN was tested with MSE and MAE, while HardNet was tried with triplet loss combined with MAE and MSE. Due to the squaring, MSE has only one solution, at the expense of penalising outliers much more strongly. These were evaluated by looking at the results of `HPatches benchmark` (Table 3), from which MAE and MAE with triplet loss will be used for the denoiser and descriptor networks respectively going forward.

Denoiser & Descriptor Error	Verification	Matching	Retrieval	Total Score
MAE & MAE	0.740663	0.122394	0.377653	1.24071
MSE & MAE	0.675923	0.10385	0.325832	1.105605
MSE & MSE	0.367023	0.084952	0.198352	0.650327
MAE & MSE	0.489893	0.028507	0.201406	0.719806

Table 3. `HPatches benchmark` scores after training with different loss functions

3.3. Batch Size

Batch sizes of 25, 50 and 100 were tested when training the network. A lower batch size would mean updating the weights is less computationally expensive, but the downside is the updates to the weights would deviate from the optimal gradient descent vector more strongly so more epochs are needed to reach a minimum. A larger batch size has the inverse problem, with weight updates being more expensive to compute but more closely following the optimal gradient descent. The batch sizes were evaluated over 10 epochs using `HPatches benchmark` as a metric. The results in Figure 6 show that a batch size of 100 reaches a lower MAE loss of 4.2267 compared to batch size 50 reaching 4.2533, the validation loss of batch size 50 is lower at 4.3023 while batch size 100 is at 4.7324, so a batch size of 50 will continue to be used going forward.

3.4. Optimizers

The DnCNN denoiser network was run with ADAM, RMSProp, AdaDelta and SGD with varying parameters. The learning curves of the optimizers are show in 7 from which it is apparent that ADAM with $\eta = 0.01$ is the most effective.

The same process was applied HardNet with results shown in 8 with SGD with $\eta = 0.1$ being the best.

3.5. Activation Function

Both the networks currently use ReLU activation on their layers, which can cause issues due to vanishing gradients killing neurons in the middle of the network. HardNet and DnCNN were tested with Leaky ReLU activation with a range of gradients α; the results are in Figs. 9 and 10. Using the results, Leaky ReLU with $\alpha = 0.2$ for DnCNN and Leaky ReLU with $\alpha = 0.1$ for HardNet shall be used going forward.

3.6. Weight Initialisation

The initial weights assigned to the networks can cause exploding and vanishing gradient issues. Both networks were tested with Xavier Normal, He Normal and orthogonal weight initialisations; the results are found in Figs. 11 and 12. From this, both networks will use He Normal weight initialisations.

4. Final Model

The final denoiser model is a 17-layer DnCNN with the following parameters:

- **Loss function:** MAE
- **Batch size:** 50
- **Optimizer:** ADAM with $\eta = 0.01$
- **Activation function:** Leaky ReLU with $\alpha = 0.2$
- **Weight initialisation:** He Normal

The final descriptor model is the baseline 7-layer HardNet with the following parameters:

- **Loss function:** Triplet loss with MAE
- **Batch size:** 50
- **Optimizer:** SGD with $\eta = 0.1$
- **Activation function:** Leaky ReLU with $\alpha = 0.1$
- **Weight initialisation:** He Normal

This was then trained over 30 epochs, the denoiser and descriptor network learning curves can be found in Figs. 13 and 14. The `HPatches benchmark` scores were 0.811292, 0.226501, 0.516589 for verification, matching and retrieval respectively.

5. Conclusion

The final proposed model offers some improvements over the baseline, increasing the total `HPatches benchmark` score by 0.360988 when trained under the previously described parameters. Due to time limitations, not all possibilities and combinations of parameters and models have been tested exhaustively.

For future improvements, changes to the network could tested exhaustively to fully optimise the model with the given parameters. There are also new models and loss functions that could be tested, them being:

- **FFDNet denoiser:** deconstructs the input into four sub-images and passes them with a noise map into a DnCNN-like network, and reconstructs the image at the output [12]
- **SSIM loss:** measures the structural similarity between two images as a weighted combination of comparative luminance, contrast and correlation [9]
- **MS-SSIM loss:** uses the same parameters as SSIM but over different scalings of the images [10]

References

[1] V. Balntas, K. Lenc, A. Vedaldi, and K. Mikolajczyk. Hpatches: A benchmark and evaluation of handcrafted and learned local descriptors. *CoRR*, abs/1704.05939, 2017.

[2] cszn. Dncnn implemenation for keras.

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

[4] A. Mishchuk, D. Mishkin, F. Radenovic, and J. Matas. Working hard to know your neighbor's margins: Local descriptor learning loss. *CoRR*, abs/1705.10872, 2017.

[5] raghakot. Resnet implementation for keras.

[6] O. Ronneberger, P. Fischer, and T. Brox. U-net: Convolutional networks for biomedical image segmentation. *CoRR*, abs/1505.04597, 2015.

[7] F. Schroff, D. Kalenichenko, and J. Philbin. Facenet: A unified embedding for face recognition and clustering. *CoRR*, abs/1503.03832, 2015.

[8] 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 2017, Honolulu, HI, USA, July 21-26, 2017*, pages 6128–6136, 2017.

[9] Z. Wang, A. C. Bovik, H. R. Sheikh, and E. P. Simoncelli. Image quality assessment: From error visibility to structural similarity. *IEEE TRANSACTIONS ON IMAGE PROCESSING*, 13(4):600–612, 2004.

[10] Z. Wang, E. Simoncelli, and A. Bovik. Multiscale structural similarity for image quality assessment. *Conference Record of the Asilomar Conference on Signals, Systems and Computers*, 2:1398 – 1402 Vol.2, 12 2003.

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

[12] K. Zhang, W. Zuo, and L. Zhang. Ffdnet: Toward a fast and flexible solution for CNN based image denoising. *CoRR*, abs/1710.04026, 2017.

Appendix

A. Figures

To use the code used for this report, both `get_denoise_model` and `get_descriptor_model` have had arguments added to their functions allowing to easily switch between architectures. Some optimizers have also been assigned to variables. Otherwise the parameters changed in this report were done by finding the relevant code and changing it.

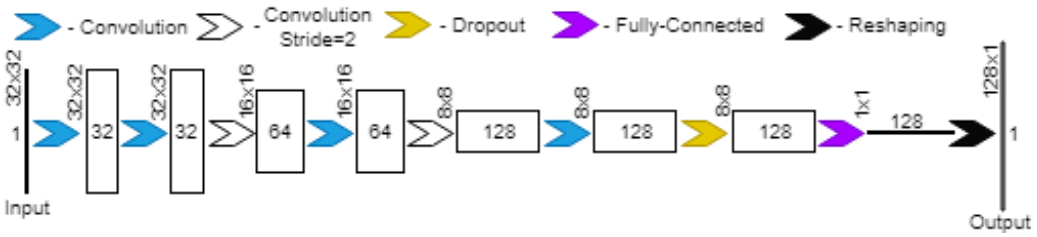

Figure 1. Baseline single descriptor architecture

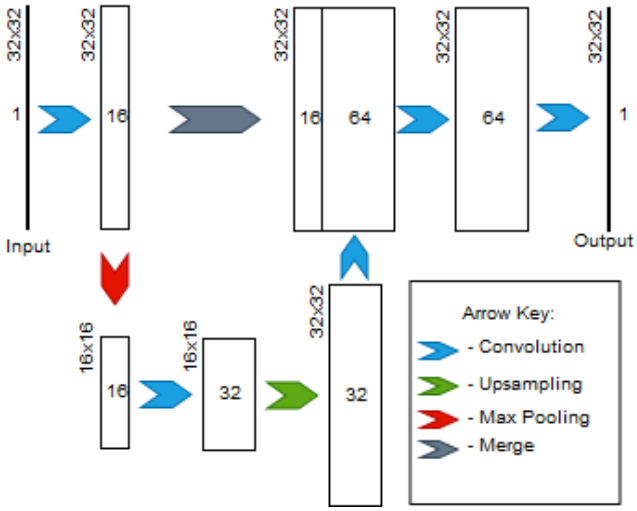

Figure 2. Baseline denoiser architecture

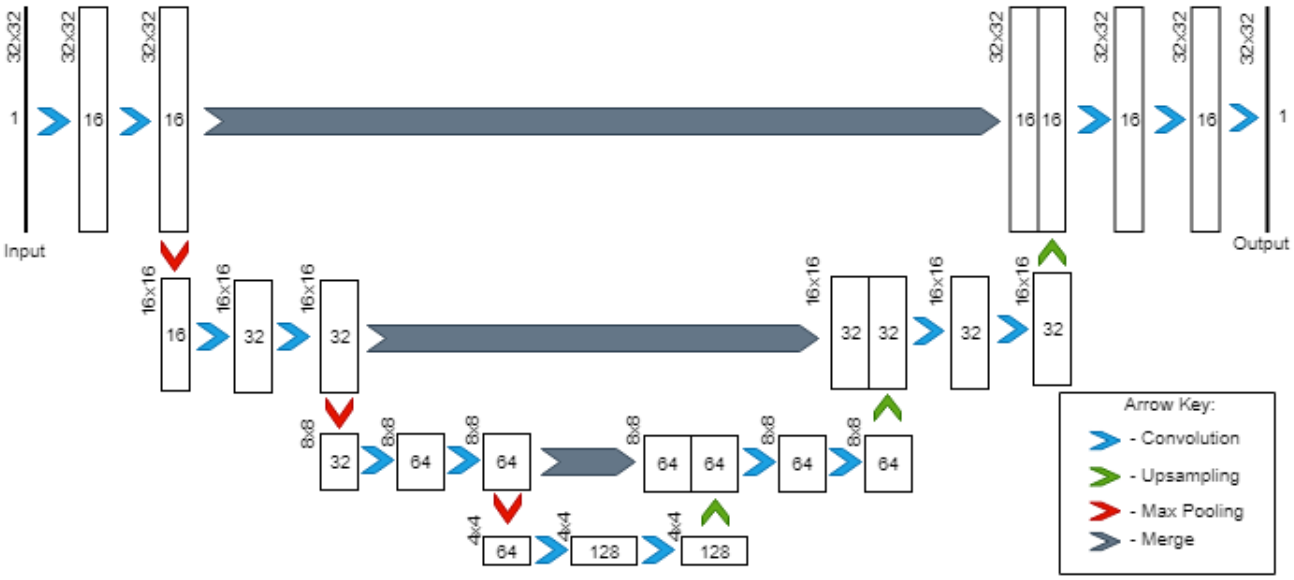

Figure 3. U-Net denoiser architecture

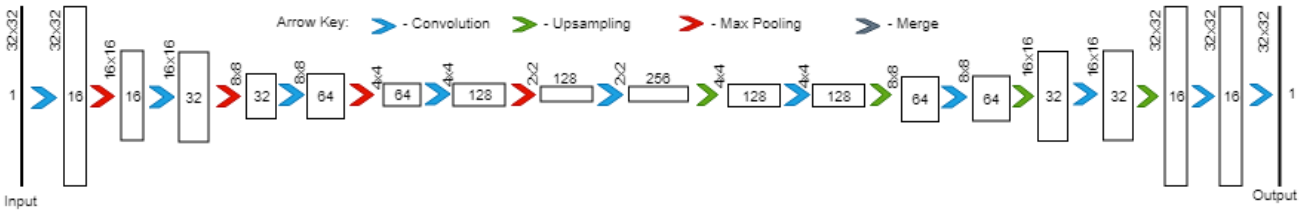

Figure 4. Autoencoder denoiser architecture

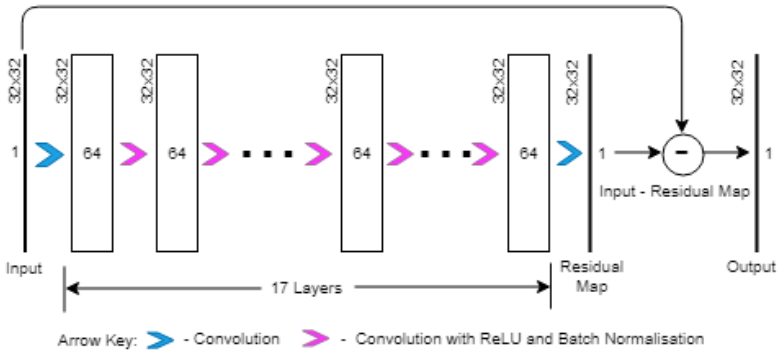

Figure 5. ResNet-50 architecture

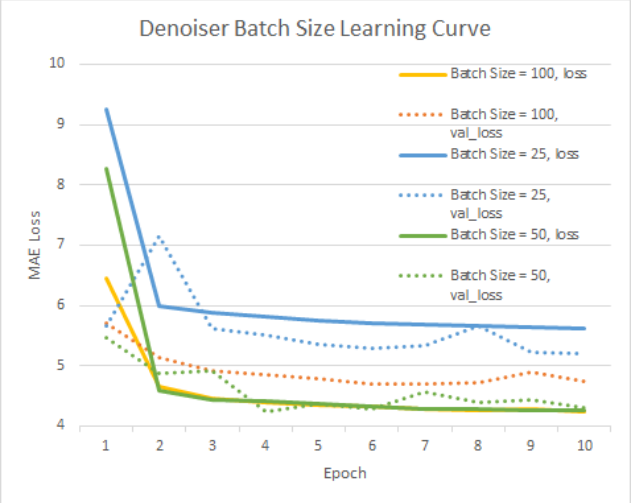

Figure 6. Learning curve of batch sizes over 10 epochs

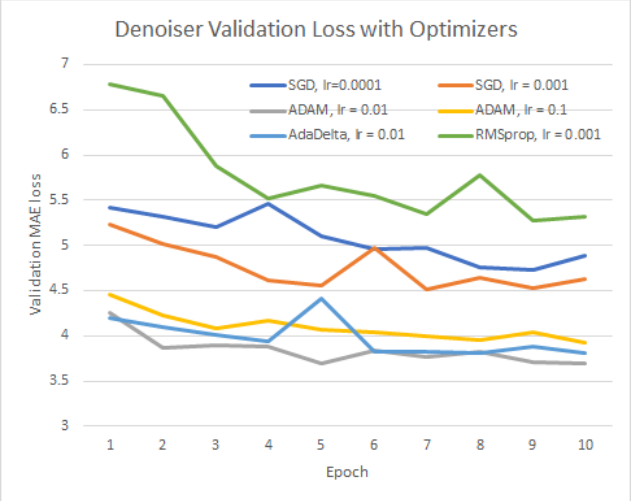

Figure 7. Learning curve of optimizers over 10 epochs

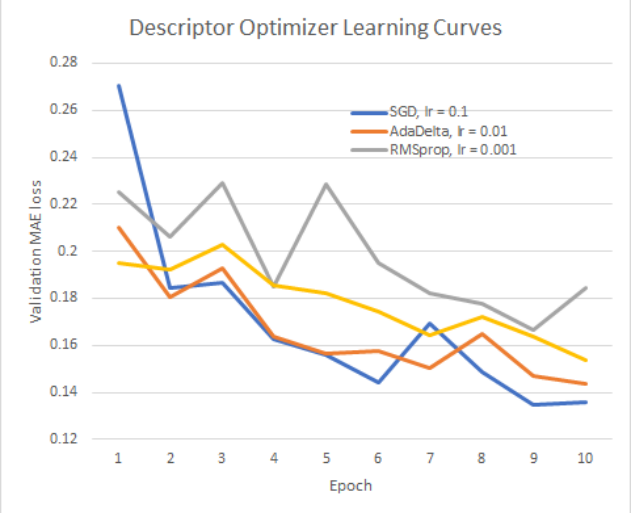

Figure 8. Learning curve of optimizers over 10 epochs

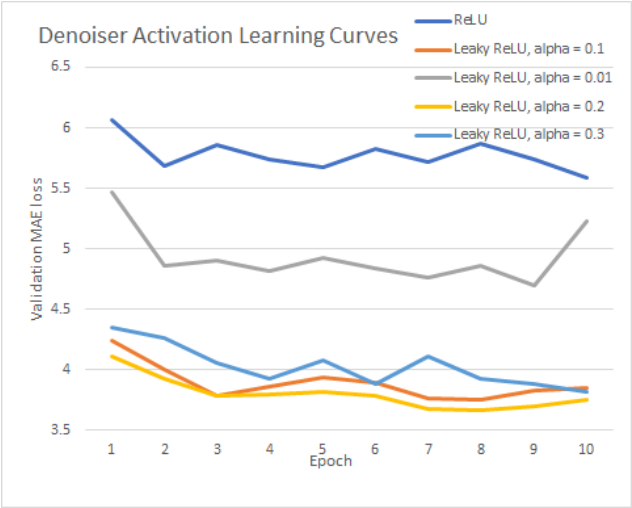

Figure 9. Learning curve of activation functions over 10 epochs

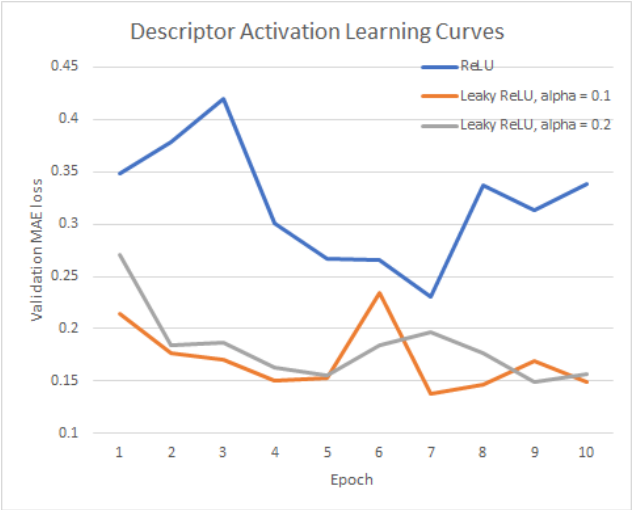

Figure 10. Learning curve of activation functions over 10 epochs

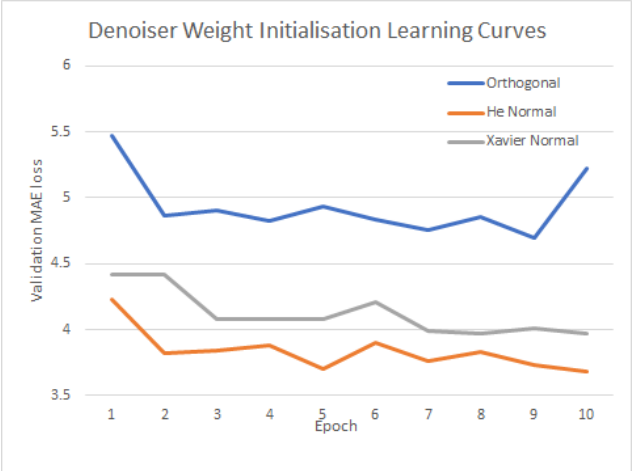

Figure 11. Learning curve of weight initialisations over 10 epochs

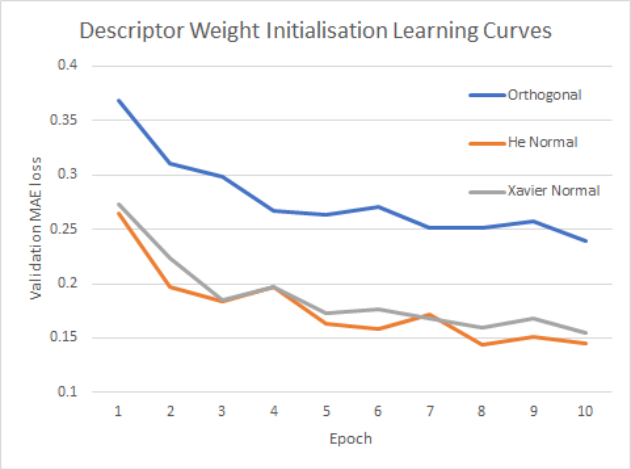

Figure 12. Learning curve of weight initialisations over 10 epochs

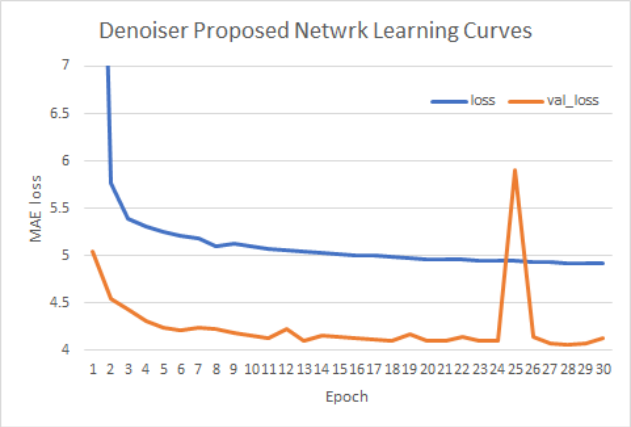

Figure 13. Learning curve of proposed denoiser network over 30 epochs

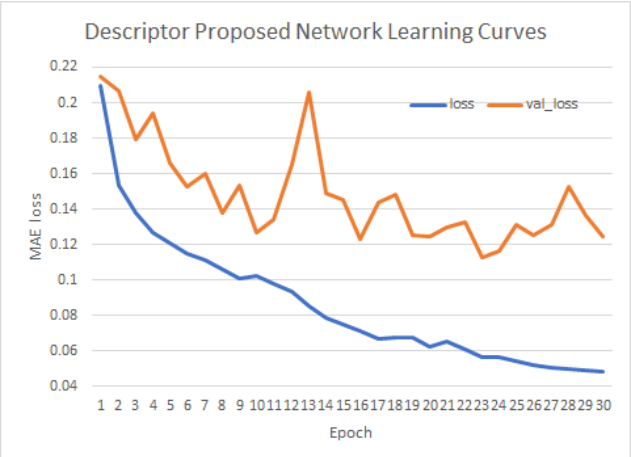

Figure 14. Learning curve of proposed descriptor network over 30 epochs