Final Report Deep Learning

Alejandro Gilson Campillo
Imperial College London
CID: 01112712
alejandro.gilson15@imperial.ac.uk
https://drive.google.com/open?id=1tJFSihYBFWFycxXSm--rU7TIUZcadmcU

Abstract

Self organizing maps are of great importance in the area of unsupervised learning. They learn a descriptor space to which the input is mapped. This process generates clusters that aid in the task of data classification. However, finding a suitable transformation is a challenging endeavor, especially when the number of classes is large. In this experiment, several neural network models are constructed for this purpose.

1. Introduction

1.1. Dataset description

The dataset that will be used for this experiment is the N-HPatches [1]. This is a noisy version of HPatches [2] with three levels of noise: easy, hard and tough. It contains 116 sequences of images where 57 and 59 of them contain images with photometric and geometric transformations, respectively. Each of the sequences is composed of one original image and 4 different transformations. The strength of these transformations is determined by the same three levels of noise added to the HPatches dataset.

1.2. Problem Formulation

The goal of this experiment is to generate patch descriptors that minimize the euclidean distance between descriptors with similar patches and maximize the distance between the dissimilar ones. The performance of the descriptors will be evaluated by three tasks: verification, matching and retrieval. These determine the ability of the descriptors of size 1 by 128 to classify the transformed patches to the original one, identify correspondences in two images, and assign a query patch to a pool of extracted patches. As a baseline, a simple model that first removes the noise from the 32x32x1 patches and then trains the descriptors will be implemented. Improvements to this algorithm will be proposed and evaluated. The loss function that will be used for training the denoiser is the mean absolute error as defined in Eq. 1. For the descriptor, the triplet loss, which consists of the error between positive, anchor and negative images. The euclidean distance between the first two is minimized while the distance between the last two is maximized as defined in Eq. 2

$$MAE = \frac{\sum_{i=1}^{N}(y_i - x_i)}{N} \tag{1}$$

$$\mathcal{L} = \max\left(\|\delta_+\|^2 - \|\delta_-\|^2 + \alpha, 0\right) \tag{2}$$

where δ_+ is the difference between the positive and anchor images, and δ_- is the difference between the negative and anchor images. α is the margin parameter which is set equal to 1 to avoid a trivial solution.

2. Baseline model

2.1. Denoise model

Eliminating the noise present in the patches is critical in order to have a good performing descriptor. For this reason a shallow U-Net [3] is trained to output a denoised image that minimizes the MAE with respect to the clean image. This is achieved by employing a 5-layer convolutional neural network that takes a 32x32x1 patch as an input and where the first and third layers are concatenated. The number of filters in each of the convolutional layers is 16, 32, 64, 64 and 1, in order. The kernel size of all the layers is of 3 by 3, except that of the third layer, whose size is 2 by 2. The strides are of 1 by 1 in all layers and the padding is set to *same*. The activation function for all layers of this network is *ReLu*. A *max-pooling* function is applied to the network just after the first layer in order to compress the information. Each of the layers is initialized with *he-normal*, a random initializer whose shape is a normal distribution with $\mu = 0$ and $\sigma = \sqrt{2/\text{number of inputs}}$. This architecture allows the output to be of the same size as the input. One special characteristic of this neural network is that it is an autoencoder. It compresses the input from size 32x32 to 16x16 and decompresses it back to its original size. The model is trained to minimize the reconstruction error and therefore it must learn to acquire the most important features of the patch. This is achieved by reducing the noise. A schematic of the resulting network can be seen in figure 1.

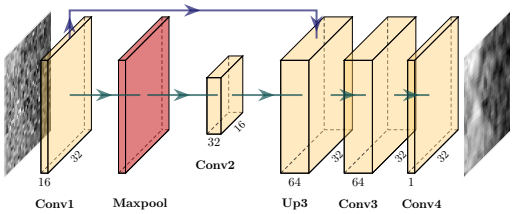

Figure 1. Network schematic of the baseline denoiser model. Generated using [4].

2.2. Descriptor model

The second stage of the baseline model consists of a 7-layer convolutional neural network that generates the descriptors.

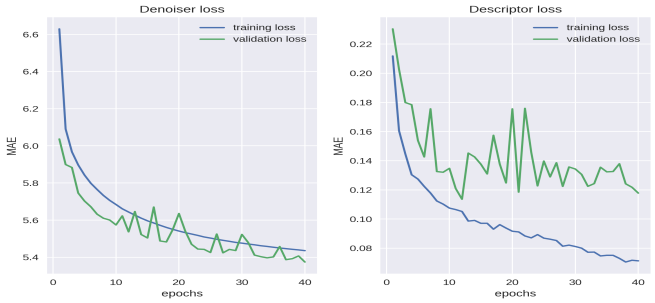

Figure 2. Training and validation error of the denoiser and descriptor baseline models.

These take as an input the 32x32x1 patches generated by the denoiser model and converts them to descriptors of size 128x1 that minimize the triplet loss. The number of filters in each of the layers is 32, 32, 64, 64, 128, 128 and 128, in order. All the convlution layers employ a kernel of size 3 by 3, have a padding of *same* and are followed by batch normalization and a *ReLu* activation function, except the final layer, which uses a kernel size of 8 by 8 and a padding of *valid*. Finally, the output is reshaped so that it is of size 1x128. Just as with the denoiser model, the initialization of the weights is done using *he-normal*. A schematic of the resulting network is shown in figure 3.

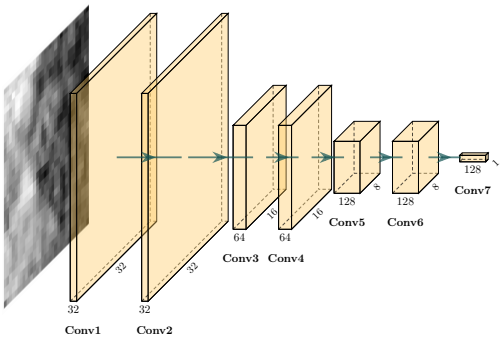

Figure 3. Network schematic of the baseline descriptor model. Generated using [4].

2.3. Results baseline

The baseline model provides a benchmark with which the rest of the models will be compared. The training and validation errors are shown in figure 2. It can be observed how the training and validation loss of both models go down with the number of epochs. Only 40 epochs were computed but if the network had been left to run for longer, the validation loss would have begun to increase and an optimal training point would have been found. Surprisingly, for the given data, the validation loss is lower than the training loss for the denoiser model. Some of the reasons for this could be that the model generalizes very well or that the split of validation data taken is very easy. The test results for validation, matching and retrieval with 40 epochs of training for the denoiser and descriptor models and using mAP are 0.841989, 0.260636 and 0.563411, respectively.

3. Improving the denoiser

3.1. The DnCNN denoiser

Image denoising has been an active topic for many years. Because of this many state of the art algorithms have arised such as BMRD [5], LSSC [6] and NCSR [7]. For this experiment, the DnCNN [8] will be used. This is a very deep CNN with many layers that is effective at capturing the characteristics of images [9]. It contains many batch normalization [10] layers and ReLU [11] activation functions which speed up the learning process and improve the denoising performance. The noisy input patch y is modelled by $y = x + u$, where x is the clean patch and u is the noise. The DnCNN is designed to estimate the noise in the patch and then substract it from the noisy patch.

The architecture of the model is composed of 17 convolutional layers with 64 filters each, a kernel size of 3 by 3 and a stride of 1 by 1. It takes as an input the 32x32x1 patch and outputs a denoised patch of the same size. All layers are followed by a batch normalization operation, except the first and last layers. After this operation and, except for the final layer the ReLU activation function is employed. The model is trained to estimate the noise, therefore, a subtraction layer is used at the end to eliminate the noise from the original input patch. Just as before, the MAE is used as the loss function. The weights are initialized using an orthogonal random matrix with a gain of 1. This initialization method is very effective in avoiding exploding or vanishing gradients since the eigenvalues of the matrix are equal to one, and thus, will make the gradients backpropagate more effectively. The resulting architecture is displayed in figure 4.

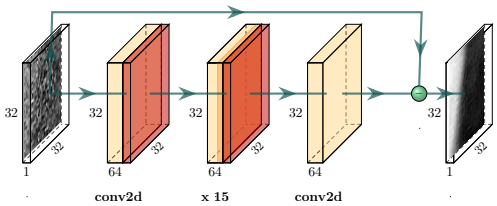

Figure 4. Network schematic of the DnCNN denoiser model. Generated using [4].

The selection of an appropriate optimizer and learning rate is crucial for a well performing network. If the learning rate is too large, the loss may never converge or the learnt weights may not be optimal. On the other hand, if the learning rate is too low, it will take longer to converge. However, testing a wide range of values is very time consuming. For this reason, only two optimizers are tested: Adam and SGD with learning rates of 0.001 and 0.00001, respectively. Due to the depth of the network, both models are only trained for 6 epochs. The results are shown in figure 5. It can be seen that the performance of the DnCNN with the Adam optimizer is better than the baseline denoiser model. Moreover, from a qualitative point of view, the image patches seen in figure 6 from this model seem clearer. The optimal training point is achieved with 5 epochs, after which, the model over-fits. On the other hand, the DnCNN network with the SGD optimizer has a poor per-

formance and an optimal training epoch is not found with just 6 epochs.

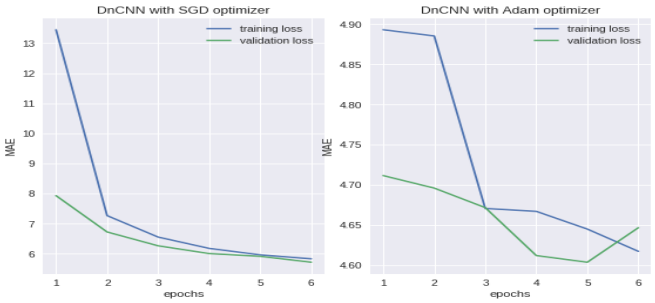

Figure 5. Training and validation loss of DnCNN denoiser with SGD and ADAM optimizers

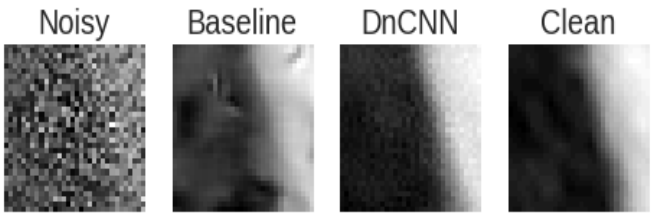

Figure 6. Noisy, clean and denoised patches by the baseline and the DnCNN with adam optimizer models trained for 40 epochs.

3.2. Changing the loss function

The loss function used for the baseline model was the MAE. This function gives the same linear penalty to datapoints from the denoised patch that deviate by different amounts from the clean image patch. However, this loss function may not be optimal, better performance could be achieved if the datapoints further away from the ground truth have harsher penalties. This can be achieved by the use of the MSE defined in Eq. 3.

$$MSE = \frac{\sum_{i=1}^{N}(y_i - x_i)^2}{N} \tag{3}$$

Since this is equivalent to the square of MAE, datapoints close to the ground truth are less penalized than those further away. The training and validation error for this loss function is displayed in figure 7. The evaluation metric is MAE for comparison purposes. After 5 epochs the model over-fits. Overall, the loss is worse than when using MAE, but, since the model was trained to minimize a different loss function, this does not mean that it has a lower performance.

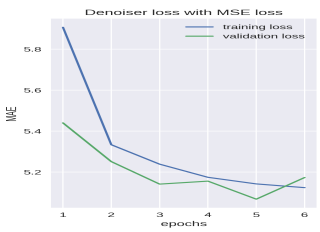

Figure 7. Training and validation loss of the baseline denoiser with MSE loss

4. Improving the descriptor

4.1. Updated baseline

The architecture used in the baseline for the descriptor model is well thought and appropriate for the problem at hand. For this reason, no mayor structural changes are done to the neural network. However, minor improvements could be achieved by modifying the initialization of the weights and adding some dropout layers.

As explained in section 3, exploding and vanishing gradients are a recurring problem in neural networks. This could is solved by initializing the weights with a random orthogonal matrix, where the eigenvalues are all one.

Empirical evidence shows that Dropout layers increase the resilience of models to over-fitting [12]. For this reason, for each convolutional layer, a dropout with probability of 0.3 is added after the activation layer. The overall performance of the model is shown in figure 8 (left). The results show that there is no significant improvement over the baseline model.

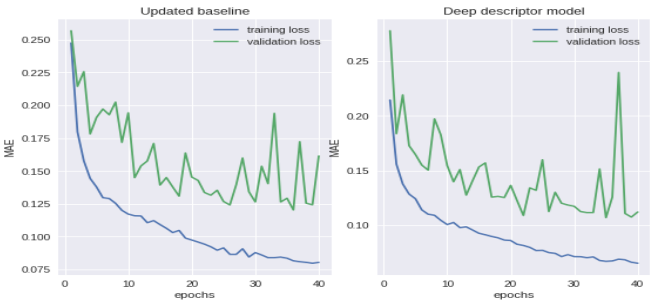

Figure 8. Training and validation loss of the updated descriptor model.

4.2. Deep baseline

Due to the lack of improvement with the previous approach. A variation of the architecture is undertaken whereby the number of layers is increased. If the baseline descriptor layers consisted of 2D convolutional layers with 32, 32, 64, 64, 128, 128 and 128 number of filters; the new model contains the same blocks but of the form 32, 32, 32, 64, 64, 64, 128, 128 and 128. In order to not alter the shape of the output, the number of strides of the newly added layers is kept to 1 by 1. Finally, as with the baseline model, batch normalization and ReLU activation functions follow the convolutional layers. The performance of this new model can be seen in figure 8 (right). Again, not much improvement is achieved.

4.3. Data augmentation

It is of interest to analyse whether the best performing model can improve its generalization and made more robust by training on a new set of data. However, since we are limited by the dataset in HPatches, new data must be generated by data augmentation. The pre-trained model on the real dataset is retrained with flipped patches. The model might interpret this as a new set of data and improve its performance. The results for the baseline descriptor trained for 40 epochs and retrained with the augmented dataset are displayed in figure 9.

It can be observed that the validation loss is slightly reduced which could be a sign that the model has improved. However,

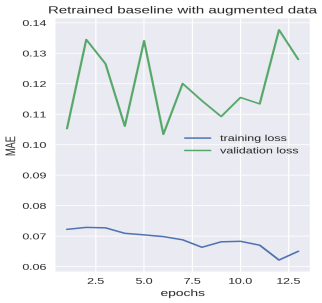

Figure 9. Baseline descriptor retrained with flipped patches

Denoiser model	Descriptor model	Epochs	Epochs
Baseline	Baseline	40	40
DnCNN	Baseline	5	40
DnCNN	Updated baseline	5	40
DnCNN	Deep baseline	5	40
Baseline	Retrained baseline	40	13
DnCNN	Retrained baseline	5	13

Table 1. Epochs used for evaluation, matching and retrieval.

a final result with the matching, evaluation and retrieval performances will be shown in section 5.

4.4. Hard triplet loss

The triplet loss employed throughout this experiment fails to take into account the distance $\delta_{-}^{'} = f(P) - f(N)$, where P and N are the positive and negative images, and, for this reason, the hard triplet loss [13] is introduced. Following this approach, Eq. 2 is changed to:

$$\mathcal{L} = \max\left(\|\delta_{+}\|^2 - \|\delta_{*}\|^2 + \alpha, 0\right) \tag{4}$$

where $\delta_{*} = \min\left(\delta_{-}, \delta_{-}^{'}\right)$. This ensures that the hardest negative inside the triplet is used for backpropagation. The training and validation losses are shown in figure 10.

Figure 10. Baseline descriptor with flipped patches

5. Conclusion

This experiment has studied different approaches for generating descriptors that map noisy input patches to a descriptor space. The first task was to try and improve the denoising performance. This was done with the use of the DnCNN, a deep CNN that achieves state of the art results. Then, it was studied whether a change from MAE to MSE loss would yield a lower loss.

On the other hand, the baseline descriptor model was updated with the addition of dropout and with a different initialization method. Furthermore, a deeper baseline architecture was implemented in order to try and achieve better results. Then, data augmentation was used. The baseline model was retrained with the original dataset transformed with a rotation of 180 degrees. Finally, a change of the descriptor triplet loss was performed, this time, the distance between the negative and positive patches was taken into account. The results are displayed in table 2. After many attempts, the baseline model is still the best performing one.

Denoiser model	Descriptor model	Verification	Matching	Retrieval
Baseline	Baseline	0.841989	0.260636	0.563411
DnCNN	Baseline	0.831896	0.257823	0.546754
DnCNN	Updated baseline	0.76265	0.185319	0.481561
DnCNN	Deep baseline	0.837206	0.250349	0.542044
Baseline	Retrained baseline	0.816596	0.240827	0.544785
DnCNN	Retrained baseline	0.812047	0.243429	0.535071

Table 2. Final performance results for all the models.

Appendices

A. Extra figures

References

[1] *Keras triplet descriptor*. URL: https://github.com/MatchLab-Imperial/keras_triplet_descriptor.

[2] Vassileios Balntas et al. "HPatches: A benchmark and evaluation of handcrafted and learned local descriptors". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2017, pp. 5173–5182.

[3] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. "U-net: Convolutional networks for biomedical image segmentation". In: *International Conference on Medical image computing and computer-assisted intervention*. Springer. 2015, pp. 234–241.

[4] Haris Iqbal. *HarisIqbal88/PlotNeuralNet v1.0.0*. Dec. 2018. DOI: 10.5281/zenodo.2526396. URL: https://doi.org/10.5281/zenodo.2526396.

[5] Kostadin Dabov, Alessandro Foi, and Karen Egiazarian. "Video denoising by sparse 3D transform-domain collaborative filtering". In: *2007 15th European Signal Processing Conference*. IEEE. 2007, pp. 145–149.

[6] Julien Mairal et al. "Non-local sparse models for image restoration". In: *2009 IEEE 12th International Conference on Computer Vision (ICCV)*. IEEE. 2009, pp. 2272–2279.

[7] Weisheng Dong et al. "Nonlocally centralized sparse representation for image restoration". In: *IEEE Transactions on Image Processing* 22.4 (2013), pp. 1620–1630.

[8] Kai Zhang et al. "Beyond a gaussian denoiser: Residual learning of deep cnn for image denoising". In: *IEEE Transactions on Image Processing* 26.7 (2017), pp. 3142–3155.

[9] Karen Simonyan and Andrew Zisserman. "Very deep convolutional networks for large-scale image recognition". In: *arXiv preprint arXiv:1409.1556* (2014).

[10] Sergey Ioffe and Christian Szegedy. "Batch normalization: Accelerating deep network training by reducing internal covariate shift". In: *arXiv preprint arXiv:1502.03167* (2015).

[11] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. "Imagenet classification with deep convolutional neural networks". In: *Advances in neural information processing systems*. 2012, pp. 1097–1105.

[12] Nitish Srivastava et al. "Dropout: a simple way to prevent neural networks from overfitting". In: *The Journal of Machine Learning Research* 15.1 (2014), pp. 1929–1958.

[13] Vassileios Balntas et al. "Learning local feature descriptors with triplets and shallow convolutional neural networks." In: *BMVC*. Vol. 1. 2. 2016, p. 3.