Deep Learning Coursework 2019

Desmond Sy
cks116
01210056
cks116@ic.ac.uk

Abstract

The aim of this coursework is to train a descriptor network such that it is able to perform accurately in the evaluation tasks: matching, verification and retrieval, on the clean and noisy versions of image patches. To do this, two CNN architectures are used sequentially. First, a shallow U-Net is used to denoise the patches, followed by an L2-Net that outputs descriptors for the denoised patches. This first half of the report will be dedicated to evaluating the baseline code, and the latter will be on modifications to the baseline code. A final model will also be trained on successful modifications.

1. Problem Formulation

To begin formulating the type of machine learning problem, the dataset will first be discussed. The dataset contains sampled patch sequences from different image sequences, and each image sequence contains a reference image and five homographies of that image. Additionally, a noisy version of each patch sequence is also provided, where the noise added varies from low to high. Two CNN architectures are used as they perform better feature extraction on images. First of all, a shallow U-Net is used to denoise the noisy sequences, with equal input and output dimensions of $1 \times 32 \times 32$. The baseline U-Net uses the architecture shown in figure 1.

This is an example of supervised learning, because the ground truth sequences are used as labels in training the UNet. The denoised patches are then used as input to another CNN, L2-Net, which generates 128-dimensional vectors called descriptors as outputs. The architecture is shown in figure 2.

The similarity is measured by the L2 distance between two descriptor vectors. The L2-Net trained via a triplet loss metric. In the baseline model the triplet loss takes in an anchor patch, a positive patch, and a negative patch. A positive

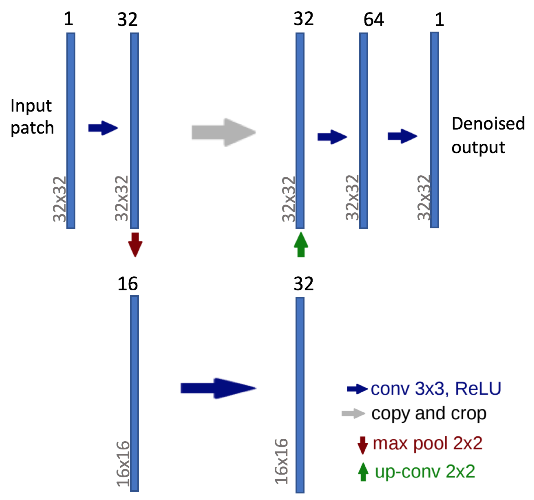

Figure 1. Shallow U-Net Architecture [2]

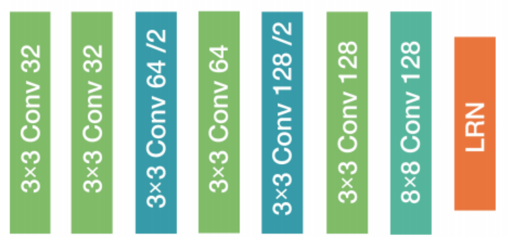

Figure 2. L2-Net Architecture [4]

patch is likely to come from the same sequence as the anchor patch, and therefore the network output descriptors of these two patches ideally should have a small L2 distance, and vice versa for a negative patch. The loss function for this is expressed as:

$$\ell(a,p,n) = max(||f(a) - f(p)||^2 - ||f(a) - f(n)||^2 + \alpha, 0)$$

where $f(a)$, $f(p)$ and $f(n)$ represent the descriptor output for the anchor, positive and negative, and α is a margin preventing the existence of trivial solutions. The weights of the L2-Net are updated in a way such that this loss function is minimized.

Lastly, the networks will be evaluated based on three tasks: Patch Verification, Image Matching and Patch Retrieval. Verification is using the output descriptors to classify whether two images are similar to each other. Retrieval is when the descriptors are used to find similar images in a large gallery, with the majority being dissimilar images. Matching is when the descriptors are used to find similar images in a small gallery, but the dissimilar images in this gallery are harder to be distinguished [1]. The metric used for these tasks is mean Average Precision (mAP).

2. Network Training and Evaluation

Firstly, the shallow U-Net is trained with the noisy patches as inputs, and clean patches as the ground truth labels. In the baseline code, the data is split into a training set, consisting of 76 sequences, and a validation set, consisting of 40 sequences. The U-Net is trained on 3 random image sequences in the training set, and validated on 1 random image sequence in the validation set, along with a batch size of 50. The loss function used is a mean absolute error (MAE) metric, which computes the average of all absolute errors. SGD with Nesterov momentum is used an optimizer, with parameters $\eta = 0.00001$ and $\beta = 0.9$. Training this baseline U-Net on twenty epochs gives the following results:

Epochs	MAE	Validation MAE
1	10.4112	9.9026
20	5.4023	7.8345

Table 1. Baseline U-Net error for 20 epochs

Figure 1 in the appendix shows that the denoised patch is not quite fully representative of the clean patch. This is reflected in the particularly large MAE values, and it is because the network was only trained on 3 random sequences, hence it does not exhibit any convergence.

The L2-Net uses SGD as an optimizer with a learning rate of $\eta = 0.1$. Training this baseline L2-Net on twenty epochs gives a triplet loss of 0.0849 and a validation triplet loss of 0.1467. Figures 2 and 3 in the appendix show plots of loss against epochs for 20 epochs for each network, with a clear decreasing trend in the losses. As the training error is decreasing, the validation error does not change to an increasing trend, which indicates that both CNN's are not overfitting.

The evaluation scores for each task are shown in table 2.

Verification	Matching	Retrieval
0.7865	0.1908	0.4891

Table 2. Baseline L2-Net evaluation scores for 20 epochs

The numbers shown do not come as a surprise. The verification mAP is the most accurate as it is likely that two randomly selected descriptors are not of the same sequence. The matching and retrieval mAP are significantly lower, suggesting that the network is not well trained to accurately find similar patches within a group of patches, especially when the patches are hard (i.e. negative patches that cannot be distinguished easily).

3. Proposed Improved Approaches

In this section, several improvements and comparisons to the baseline code are implemented with the rationale behind doing so. These results are obtained To begin, different optimizers were tested with the baseline code.

3.1. Optimizers

Five optimizers are tested with the original baseline code separately for both the U-Net and the L2-Net. The five optimizers used were: RMSProp, Adamax, Adadelta, Adam and Nadam. Each optimizer was trained on 20 epochs on the baseline code with no other modifications in order to obtain a fair comparison.

Prior to running these tests, it was hypothesized that the Nadam optimizer would yield the best results for the U-Net. Since Adam contains an adaptive learning rate provided by RMSProp as well as SGD with classic momentum, it should theoretically outperform them. Nadam is a direct improvement of Adam as it uses Nesterov momentum [**?**], therefore gradient prediction is even more accurate. The learning curves for Nadam is shown in Appendix A, figure 5.

Optimizers	MAE	Val MAE
Nesterov SGD	5.4023	7.8345
RMSProp	5.9342	5.0569
Adamax	5.1239	4.7734
Adam	5.0137	4.4422
Nadam	5.19	3.9239

Table 3. Comparison between different optimizers and their errors for U-Net

A similar comparison is done for the L2-Net. It is important to note that to ensure fair comparison, the denoised patches fed as input to the L2-Net come from the baseline SGD trained U-Net.

Interestingly enough, both Adam and Nadam perform significantly worse during mAP evaluation than the baseline results. One reason for this is because of the learning process used. The baseline L2-Net is trained using a vanilla SGD with $\eta = 0.1$. Due to unfamiliarity with the problem type, default Adam/Nadam parameters with learning rates of 0.001 and 0.002 were used to generate the values in the above table. Using a smaller learning rate means that the

Optimizers	Matching	Verification	Retrieval
SGD ($\eta = 0.1$)	0.1908	0.7865	0.4891
RMSProp	0.1897	0.7855	0.4512
Adamax	0.2131	0.8097	0.4998
Adam	0.1377	0.7452	0.4181
Nadam	0.1258	0.7300	0.4263

Table 4. Comparison between different optimizers and their evaluation scores for L2-Net

errors take longer to converge and because the number of epochs used is only 20, it is likely that the L2-Net was given an insufficient epoch length to converge on. Because of this, the learning rate is allowed to be tuned higher and exploited to observe what seems to be better evaluation results. It is reasonable to assume that if the deeper U-Nets used the entire patches dataset along with a suitable epoch length (e.g. 500), Nadam will definitely outperform the other optimizers as these may ultimately converge at undesired local minimums.

3.2. Deeper U-Net

For the U-Net, two separate changes were investigated. Due to RAM and time constraints, tests were only performed for two additional layers and for four additional layers relative to the baseline conditions. The challenge of using a shallow network consisting of only 4 layers is the fact that it cannot represent and capture the simple structures and high level patterns of the image patches. An argument for having deeper layers it that images contain some sort of hierarchical structure, and therefore intuitively it makes sense to have increased layers of processing. Theoretically, this will mean that the denoised images will be a more accurate representation of the clean patches. An excessively deep layered CNN will lead to overfitting, where the noise of the data is fitted instead. Furthermore, it may also lead to vanishing gradients and make it impossible for the model to learn. Modern CNNs are best suited to overcome this problem with methods such as skip connections in ResNet or auxiliary classifiers in GoogLeNet. However, the additional layers are altogether still relatively shallow, and hence there is no need to employ such techniques to enhance the model.

There is no structured method to determine the optimal number of layers; it depends largely on the nature and parameters of the problem. The table of results is shown below:

It is clear that having 4 additional layers results in better loss values and this deeper architecture will be used in the final model. The learning curves can be found in Appendix A.

	2 additional layers		4 additional layers	
Epochs	MAE	Val MAE	MAE	Val MAE
1	10.0617	6.2488	9.8688	6.0589
10	7.0368	6.0642	6.3544	5.1031
20	6.3434	5.1044	6.1205	4.9463

Table 5. Comparison of errors between adding 2 and 4 extra layers to the baseline U-Net

3.3. Weight Initialization

Three separate weight initializers tested as proposed changes to the baseline. LeCun uniform, Xavier normal and the orthogonal initializations were tested for both networks. The results are shown in tables 6 and 7 for the U-Net and L2-Net respectively:

Initializer	MAE	Val MAE
He Normal	5.5203	7.9548
LeCun Uniform	6.0254	4.4490
Xavier Normal	6.5736	5.3054
Orthogonal	6.5274	5.2632

Table 6. Comparison of errors between different weight initializers for U-Net

Initializer	Matching	Verification	Retrieval
He Normal	0.1908	0.7865	0.4891
LeCun Uniform	0.1900	0.7937	0.4795
Xavier Normal	0.1977	0.7987	0.4977
Orthogonal	0.1909	0.7687	0.4946

Table 7. Comparison between different weight initializers for evaluation scores for L2-Net

The He Normal initializer is best known for working well with ReLU activation functions, which are used throughout the baseline networks. However, despite performing the best for MAE, its validation MAE is rather poor. In the evaluation tasks, it is surpassed by both the Orthogonal and LeCun uniform initializer.

An orthogonal initialization generates matrices with eigenvalues with an absolute value of 1, and therefore performing repeated multiplications does not lead to exploding or vanishing gradients, allowing much smoother backpropagations. He Normal, LeCun and Xavier initializers are very similar to each other, they serve the purpose of keeping variance similar along the layers. The results seem fairly arbitrary, which may be due to shallow layers not fully utilizing the effect of weight initialization.

3.4. Batch size

Different batch sizes for the U-Net were also tested for 20 epochs to check if it showed any signs of improvement. Table 9 shows the results obtained. Due to the way memory is laid out, batch sizes are trained in powers of 2 as it speeds up training speed.

	Batch Size				
	32	50	128	512	2048
MAE	5.8123	5.5203	6.4137	8.4627	9.1573
Val MAE	4.7206	7.9548	5.2803	6.2543	10.2080

Table 8. Comparison between different batch sizes and errors for U-Net

In training for a fixed number of epochs, using a smaller batch size means more weight updates, as more batches fit into each epoch. The data has a clear trend that highlights the apparent positive correlation between batch size and training and validation error. In theory, using a bigger batch size should give more accurate updates, at the expensive of having less updates per epoch. This is computationally more efficient as the number of SGD iterations decreases. However, similar to the optimizer section, the limiting factor here is not being able to use the entire dataset for training. For the same reason, a lower batch size gives seemingly better results in this test. In that scenario the batch size can be sufficiently large whilst also allowing enough weight updates to be made in a single epoch. It is important to note that there is an inherent performance trade off with batch size selection, and different optimizers will be better suited to a particular range of batch sizes.

3.5. Testing with clean vs denoised patches

The last test will be related to the L2-Net. The baseline code allows the L2-Net to be trained either with clean or denoised images. It is expected that the denoised patches from the U-Net will be more suitable, because these patches are effectively harder to train on, therefore resulting in an L2-Net that is better trained than if clean patches were to be used. This would lead to better mAP scores for each of the evaluation tasks. Testing this proposed hypothesis indeed does yield better evaluation scores when using denoised patches, as shown in the table below:

Patches	Matching	Verification	Retrieval
Denoised	0.2445	0.8087	0.5222
Clean	0.1997	0.7725	0.4792

Table 9. Comparing evaluation scores for using denoised and clean image patches in training the L2-Net

Thus, the final model will utilize denoised image patches from the U-Net as input to the L2-Net.

4. Proposed final model

Based on the above analysis, a final model was trained to see if the benefits of the proposed changes added up together. For the U-Net, 8 layers are used with default Nadam is used as an optimizer and LeCun uniform as the initializer. A batch size of 32 is used. For the L2-Net, denoised patches are used as input, default Adamax is used as an optimizer and Xavier uniform as the initializer. To obtain results the networks were trained on 20 epochs. It is expected that these individual benefits will have an additive effect, generating results better than any of the ones presented so far. The U-Net learning curve and evaluation results are shown below:

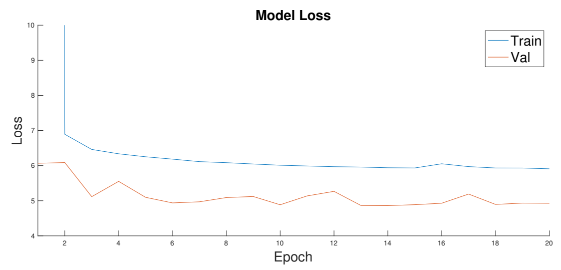

Figure 3. Learning curve for final U-Net

Verification	Matching	Retrieval
0.8322	0.2497	0.5239

Table 10. Final Model evaluation scores for final L2 network

The L2-Net evaluation results look extremely promising, but however the U-Net learning curve appears to be settling at a training error of 5.9 and a validation error of 4.9. As with much of the training done up till this point, a lot of the errors seem to be generated in a somewhat arbitrary manner. Again, this is likely due to the fact that the training and validation sets are small and randomly sampled and tests are only ran on 20 epochs. Therefore, the choice of optimizers, regularizers and other hyperparameters will not methodically affect the training losses. Nonetheless, it is a significant improvement to the baseline code.

5. Modifications

For the networks, different loss functions can be implemented to check which one gives the best accuracy. Upon having performed more trial runs for the problem, hyperparameters should be fine tuned properly. Several U-Net architectures such as U-Net++ and Framing U-Net have Triplet 'anchor swapping' [5] may also be implemented, where if two negatives are selected, one of these negatives becomes the anchor, the other negative becomes a positive, and the original anchor becomes the new negative. Similar ideas can be done for the triplet loss metric to increase its reliability, which would give better scores for the evaluation tasks.

References

[1] Vassileios Balntas and Karel Lenc and Andrea Vedaldi and Krystian Mikolajczyk, *HPatches: A benchmark and evaluation of handcrafted and learned local descriptors*. arXiv: 1704.05939 [cs], Apr. 2017.

[2] Olaf Ronneberger and Philipp Fischer and Thomas Brox, *U-Net: Convolutional Networks for Biomedical Image Segmentation*. arXiv: 1505.04597 [cs], Aug. 2018.

[3] Timothy Dozat, *Incorporating Nesterov Momentum into Adam. In Proceedings of 4th International Conference on Learning Representations, Workshop Track*, 2016.

[4] Yurun Tian, Bin Fan, and Fuchao Wu, *L2-Net: Deep learning of discriminative patch descriptor in Euclidean space. In Proceedings of 4th International In Proc. IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017.

[5] Vassileios Balntas, Edgar Riba, Daniel Ponsa, and Krystian Mikolajczyk, *Learning local feature descriptors with triplets and shallow convolutional neural networks. In British Machine Vision Conference (BMVC)*, 2016.

Appendix A

Colab: `https://colab.research.google.com/drive/1RHpT2li4N0zsXcjvSvKvynfYqraYMU_w`

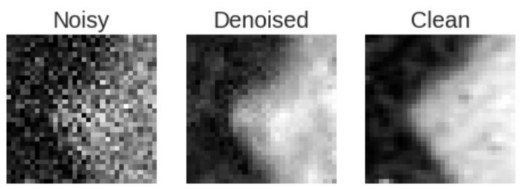

Figure 1. Denoised patch comparison

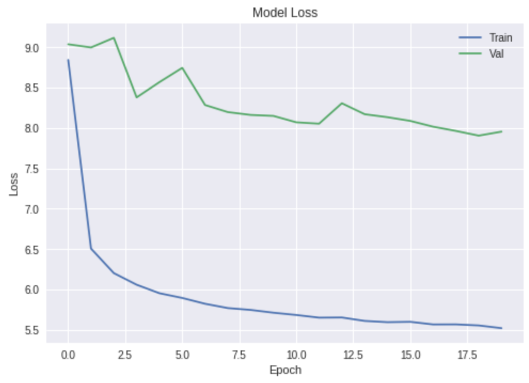

Figure 2. Baseline U-Net Model Loss with 5 epochs

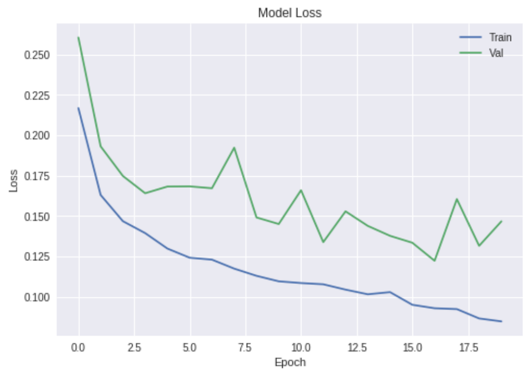

Figure 3. Baseline L2-Net Model Loss with 5 epochs

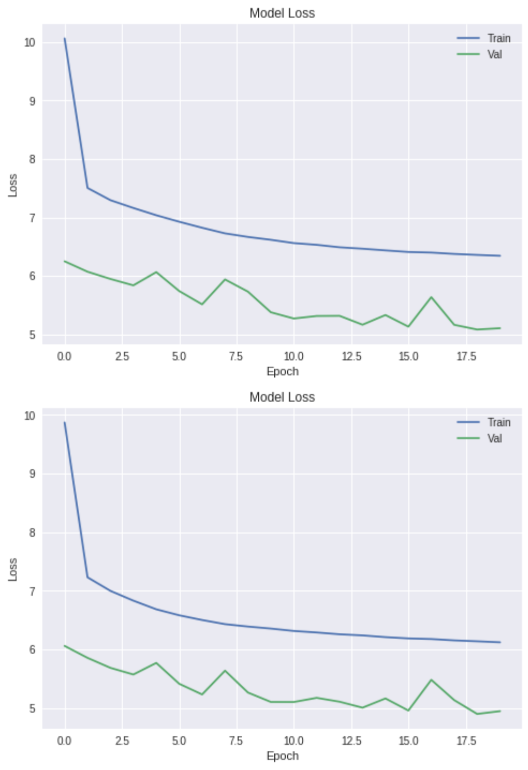

Figure 4. Learning curves for 6 and 8 layered U-Net

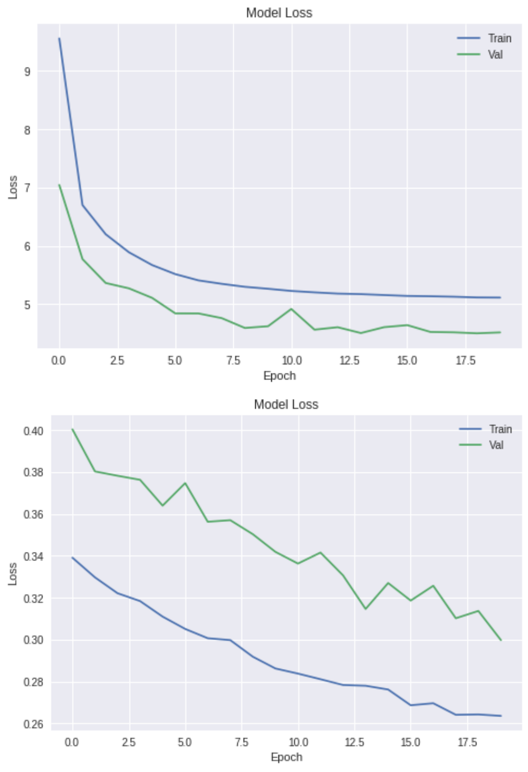

Figure 5. Nadam UNet and L2Net learning curves