Final Report

Victor Sanchez
vas116
01119900
vas116@ic.ac.uk

Abstract

This paper proposes a method to generate high-performance learned image representations to achieve noisy patches matching. We will be using learned local descriptors as image representation, which are known for high-performance in Euclidian space. The proposed method considers and trains a baseline model which is a pipeline of two convolutional neural networks, trained on the N-HPatches dataset - noisy version of HPatches. It then attempts to improve the baseline model via architectural modifications, networks pre-training and hyperparameter optimisation. Performance of the learned descriptor will be assessed in terms of patch verification, image matching and patch retrieval.

1. Learning Problem

1.1. Dataset

Our dataset - N-HPatches - is a noisy version of the HPatches dataset [1]. It contains 116 sequences, 59 based on geometric changes, 57 on photometric deformations. Each sequence contains a reference patch and 5 deformations (see figure. 1). N-Hpatches are 32x32 dimensions grayscale images.

1.2. Target function: local descriptors

Our goal is to create an efficient image representation of N-HPatches. Our image representation will be learned local descriptors of dimensions 128. Local image descriptors are used in image matching to map similar features between images which represent a similar situation. Local features are extracted from images and matched to one another via their descriptor. A successful descriptor will therefore allow us to identify whether two images depict the same item or situation, and eliminate those who are dissimilar. Descriptors will be assessed on patch verification, image matching and patch retrieval. More formally, We can describe descriptors as target functions $f(\boldsymbol{x}) \in \mathbb{R}^{128} : f : \mathbb{R}^{32x32x1} -> \mathbb{R}^{128}$.

1.3. Learning Task

First, our model will need to denoise the input patches. It will then generate the 128 dimensional descriptor from the denoised patches, using regression. The last task of our model will be to evaluate the descriptors by verification, matching, and retrieval.

1.4. Evaluation of the descriptor

We follow the evaluation metrics proposed by Balntas et al. [1]. We define the following precision measures. Let $\boldsymbol{y_i} \in \{-1, 0, 1\}^n$, $i = 1...n$. indicating positive, neutral and negative patches from a patch query. Let precision be $P_i(\boldsymbol{y}) = \sum_{k=1}^{i}[y_k]_+ / \sum_{k=1}^{i}[y_k]$. Let average precision be $A(\boldsymbol{y}) = \sum_{k:y_k=+1}[P_k(\boldsymbol{y})] / \sum_{k=1}^{N}[y_k]_+$.

1.4.1 Patch verification

Verification classifies two patches on whether they are similar or not from their descriptors. Evaluation takes a list of positive and negative patches $L = (\boldsymbol{x}_i, \boldsymbol{x}'_i, \boldsymbol{y})$, $i : 1, ..., n$, $\boldsymbol{x}_i, \boldsymbol{x}'_i \in \mathbb{R}^{32x32x1}$ patches, label $\boldsymbol{y} = \pm 1$. We evaluate the verification in terms of average precision $A(y_{\pi_i})$, π_i permutation.

1.4.2 Matching

Matching takes a query and measure how well descriptor can identify similar images from a small gallery with difficult distractors. Evaluation considers pairs of images $P = (I_0, I_1)$, I_0 reference and I_1 target. It then computes the best matching patch with associated confidence score and computes average precision based on these metrics.

1.4.3 Retrieval

Retrieval takes a query and measure how well descriptor can identify similar images from a large gallery with easy distractors. It considers a reference patch taken from an image and all patches taken from all corresponding images in the sequence. Patch $\boldsymbol{x}_i$ is then given label ± 1 if it matches or

not query patch $\boldsymbol{x}_0$. Retrieval is then estimated on average precision on the task.

2. Baseline Architecture

The baseline architecture consists of a pipeline made of two cascading networks: a shallow UNet which denoises input patches, and a L2-Net which takes the denoised patch and generates its descriptor (see fig.2).

2.1. Shallow UNet for denoising

The first network is a simplified version of the "UNet" [3], a convolutional neural network built from the "fully convolutional network"[2] by replacing pooling operations by upsampling operations. We can compare the full network to the baseline simplified version (see figures. 4 versus 3). It is trained on mean absolute error as loss function between output and its reference cleaned version. Denoising is achieved as the shallow UNet starts by decreasing image resolution, keeping only "important features". The upsampling part then increases the resolution of the kept features, in theory removing unwanted noise. Mathematically, it is performing principal component analysis, projecting vectors on a lower subspace before re-sizing them to their original dimension.

2.2. L2-Net for descriptor generation

The second network is an L2-Net [4] (see Figure 5), which converts 32x32 input patches to the objective 128 dimensional descriptor by regression. Its loss function is a triplet error, which takes three inputs: the trained descriptor (anchor), a positive descriptor (target) and a negative descriptor. Its goal is to minimise the distance between the anchor $\boldsymbol{a}$ and the target $\boldsymbol{t}$: $min\{mean\{(a-t)^2\}\}$ while maximising the distance from the negative example n: $max\{mean\{(a-n)^2\}\}$.

3. Training - Baseline Model

We train the baseline approach with base parameters set during interim report (see Tables.1 and 2). We tune the number of epochs for shallow UNet and L2-Net so that we obtain the best trade-off between training error and validation error: we aim for the minimum validation error before the model starts overfitting (see fig.7). Learning measures and evaluation are presented in tables 1 and 2, section 5.

While we scored an overall improvement, the model performs badly on both matching and retrieval, and has decent results on verification. Although L2-Net performs extremely well on HPatches dataset [4], it was designed to be trained on clean data which is hard to provide from the oversimplistic U-Net of the baseline model (see Figure6).

4. Improvements

We implement the following changes to the baseline approach.

4.1. Architecture

The most important improvement brought to the model is switching the shallow UNet for its original, full version (fig.4), adapted to support single channel images instead of originally expect three-channel RGB. The full UNet is much more complex and has more than 15 million trainable parameters, which made it extremely long to train. Further decisions were almost all taken based on this fact.

4.2. Pre-training: UNET on MNIST

As stated in the previous section, the full UNet is an extremly complex convolutinal neural network which on average took 5400s to train per epoch on full dataset, with very slow converging learning curves (regardless of the optimisers used). It was then very beneficial to pre-train the network on a simplified task, that is Gaussian noised MNIST dataset (fig.8). Three levels of noise were used (see table.5).

4.3. Pre-training: L2Net on clean patches

A similar approach was taken to pre-train the L2Net, which was trained on 20 epochs on the clean version of the HPatches, then repeated 20 epochs with learning rate divided by 10 as recommended in the L2Net paper [4].

4.4. Parameters tuning

Modifications were brought to the parameters as well. As the full version of the UNet was now in use, Adam optimiser seemed preferable to the baseline stochastic gradient descent. Adam has the particularity of storing exponentially decaying average of past squared gradients v_t and non-squared gradients m_t: $m_t = \beta_1 m_{t-1} + (1 - \beta_1 g_t)$
$v_t = \beta_1 v_{t-1} + (1 - \beta_1 g_t^2)$
The update rule is then $\theta_{t+1} = \theta_t - \frac{\eta}{\sqrt{\frac{v_t}{1-\beta_2^t}+\epsilon}} \frac{m_t}{1-\beta_1^t}$ Adam is therefore preferable for more complex networks to prevent overfitting and adapt learning rate to avoid a stall. UNet was trained following parameters described in table.6 L2Net was tuned using hyperparameters preconised in the research paper 5, see table.7.

5. Results

Results for the improved method were quite disappointing. The full UNet training had to be stopped after the 6th epoch because it would start overfitting (see learning curves fig.10). Descriptor training suffered from the unability to properly denoise the data with UNet since no real improvements were made compared to interim report (see learning curves fig11). The overall results were slightly better than

the ones obtained for the interim report, although they are not significant especially regarding how much what seems like improvement was brought to the model.

6. Challenges and further improvements

Training the improved model proved to be a challenging and arduous task. Switching straight from the shallow UNet to the full Unet was a mistake. Indeed, going from baseline UNet to the original UNet increased the network complexity by an impressive factor. Training time was therefore expanded (from a few hundred seconds to about 5400 seconds per epoch on the full dataset). It therefore proved to be almost impossible to train the denoise model on more than 10 epochs consecutively based on the computation time. Overfitting was also occurring after the second epoch due to the high-complexity of the network and the difficulty to find appropriate hyperparameters. This is when pre-training on less-noisy data proved to be useful as it increased the size of the training data set and allowed a faster training of the UNet since weights were pre-tuned to the task. A more appropriate approach would have been to add progressively more layers to the denoising network in order to avoid the previously stated issues. Training should also integrate the full data set progressively. I focused too early on training the networks on the full data sets instead of selecting random subsets in the data to test parameters and models before implementing them. This also explains why network was so time and memory consuming, which can be penalising on Colab.

Ideas for further improvements is to investigate deeper into pre-training of the networks. More types of noise can be used to pre-train the UNet: Gaussian Noise is only a first step that could be completed with uniform noise, pink noise... for a more comprehensive approach. Data augmentation was also no attempted: it may have been one of the reasons the network overfitted so quickly.

7. Figures and tables

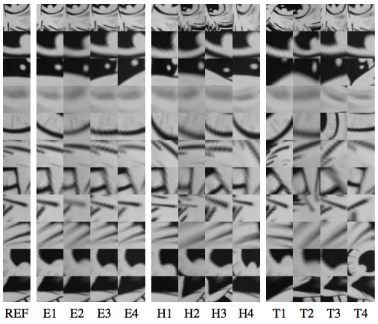

Figure 1. HPatches, example of Easy, Hard and Tough distributions. From [1]

Figure 2. Baseline architecture summary, from specifications GitHub repository

Figure 3. Baseline shallow UNet block diagram

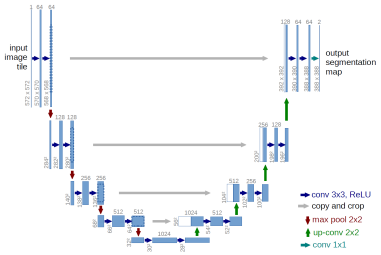

Figure 4. U-Net as proposed by Ronneberger et al. (2015) [3].

Figure 5. L2-Net as proposed by Tian et al. (2017) [4].

Table 1. Interim training parameters: UNet

Epochs	Optimizer	LR	rho	Epsilon	Decay
4	RMSprop	0.001	0.9	none	0.0005

Table 2. Interim training parameters: L2-Net

Epochs	Optimizer	LR
7	SGD	0.1

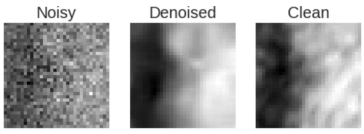

Figure 6. Interim denoise performance

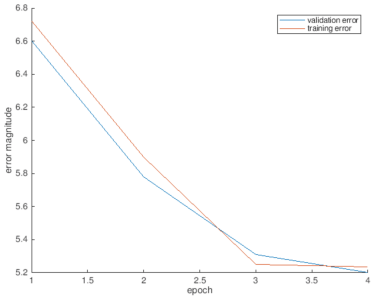

Figure 7. Interim learning curves for UNet (shallow)

Table 3. Interim model: learning metrics

	Loss	Validation Loss
U-Net	5.2360	5.2023
L2-Net	0.1226	0.1357

Table 4. Interim model: evaluation results

Verification	Matching	Retrieval
0.7888	0.1912	0.4120

Table 5. MNIST: noise levels

	Noise scale	σ^2	μ	epochs
Level 1	0.5	1	0	28
Level 2	1	2	0	8
Level 3	2	4	0	12

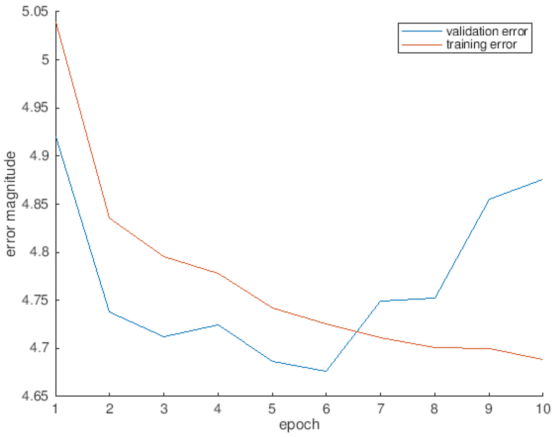

Figure 10. Learning curves for the full UNet - Mean Absolute Error

Figure 8. MNIST dataset: noisy versus clean

Table 6. Improved training parameters: U-Net

Phase	Epochs	Optimizer	LR	β_1	β_2	decay
HPatches	6	Adam	0.001	0.9	0.999	0
Denoise1	28	Adam	0.001	0.9	0.999	0
Denoise2	8	Adam	0.001	0.9	0.999	0
Denoise3	12	Adam	0.001	0.9	0.999	0

Table 7. Improved training parameters: L2-Net

Epochs	Optimizer	LR
7	SGD	0.1

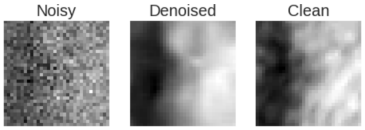

Figure 9. Improved denoise performance

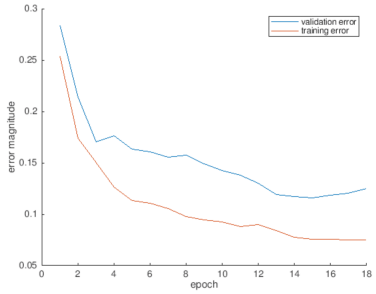

Figure 11. Learning curves for the full Descriptor - Mean Absolute Error

Table 8. Improved model: evaluation results

Verification	Matching	Retrieval
0.82	0.21	0.42

References

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

[2] J. Long, E. Shelhamer, and T. Darrell. Fully convolutional networks for semantic segmentation. 2014.

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

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