Deep Learning Interim Coursework

Omar Muttawa
om816
CID: 01204056
om816@ic.ac.uk

Abstract

This report evaluates the use of different Deep Neural Network architectures in Image representation. This is done by evaluating different model architectures ability to describe noisy patches from the NH-Patches data-set into 128x1 description vectors which are used to represent the patch and differentiate itself from other patches. Firstly a baseline model using two separate CNNs for denoising and describing is analyzed for its performance on the three defined bench marks included with NH-Patches; Verification , Matching and Retrieval , after which a proposed approach formulated by the analysis and improvement of said baseline is outlined and subsequently evaluated.

1. Formulation of Machine Learning problem

1.1. Introduction

The NH-Patches data set contains a large set of patches from real images that have been noised and then augmented randomly five times for three different levels of augmentation (Easy, hard and tough) The task at hand requires a model that takes:

- Input with shape = $\mathbb{R}^{32x32x1}$ A patch from a sequence of a noisy image from NHPatches.
- Output with shape = $\mathbb{R}^{128x1}$ A feature vector for the input patch.

This feature vector is used to represent the patch and is evaluated by three bench marks introduced in the paper 'HPatches: A benchmark and evaluation of handcrafted and learned local descriptors' [1]. Patch verification , Image matching and Patch retrieval summarized below:

1.2. Patch Verification:

This benchmark takes a list $\mathcal{P}$ containing $(\mathbf{x}_i, \mathbf{x'}_i, y_i)$ where $\mathbf{x}_i, \mathbf{x'}_i \in^{32x32x1}$ and $y = 1$ when the patches are augmented versions of the same patch and $y = -1$ when they correspond to negative patches. The models output descriptors for each patch are used to evaluate a algorithm that matches patches. The mean average precision (mAP) is calculated as the mean of 6 sets of $AP(y_{\pi 1}, ..., y_{\pi N}) = \sum_{k:y_k=+} \frac{P_k}{K}$ (1) where $P_i(\mathbf{y}) = \sum_{k=1}^{i} [y_k]_+ / \sum_{k=1}^{N} |y_k|$ and K the total number of positives.

1.3. Image Matching:

This benchmark takes a collection of N patches $L_k = (\mathbf{x}_{ik}, i = 1, ..., N)$ corresponding to two images L_0 and L_1 where L_0 is a reference image and L_1 a target. A algorithm is evaluated comparing these pairs corresponding patch descriptions to determine given a reference patch $x_{i0} \in L_0$ the index $\sigma_i \in 1, ..., N$ of the best matching pair $x_{\sigma_i 1} \in L_1$. These σ_i produce labels $y_i = 2[\sigma \stackrel{?}{=} i] - 1$ which are hence 1 when σ_i corresponds to a matching patch and -1 when it does not. These outputs are fed into (1) and averaged to produce a mAP score.

1.4. Patch retrieval

Takes a collection $\mathcal{P} = (\mathbf{x}_0, (\mathbf{x}_i, y_i), i = 1, ..., N)$ where x_0 is a query patch from a reference image L_0 and $(\mathbf{x}_i, y_i)$ a collection of patches from the same sequence as well as from other images entirely. The protocol uses the descriptors to determine whether a patch x_i corresponds to the query patch $(y_i = 1)$ or does not $(y_i = -1)$. Patches that do not correspond to the query patch but belong to the matching image L_k are ignored $(y_i = 0)$ and these labels are passed to AP to produce a mAP score.

2. Baseline Architecture

The Baseline Architecture splits the Machine Learning problem into two sections. Denoising the patches and Describing them into feature vectors. A pipeline diagram of the top-level architecture is illustrated in Figure-1.

2.1. UNet Denoising Model

The baseline shallow UNet denoise model is a shallow autoencoder architecture that through encoding(a contract-

Figure 1: Top-level base-line model pipeline

ing path consisting of convolution and max pooling operations with stride 2) and symmetric decoding (a expansive path consisting of up-sampled feature maps, up scaled 2x2 convolutions and 3x3 convolutions) [2] Such a structure will identify stable patterns within the image and avoid the non-smooth perturbation produced by the noise and can hence be used as a denoiser.

More specifically for the baseline model:

- Input **x**= A noisy image $\mathbb{R}^{32x32x1}$
- Output **y**= A denoised image $\mathbb{R}^{32x32x1}$, The clean images are labelled **f** and are used as when evaluating loss as per below.
- The base line model uses a Mean absolute Error loss where $\frac{1}{32^2}\sum_{i=1}^{32}\sum_{j=1}^{32}|f_{ij}-y_{ij}|$
- Stochastic gradient descent (sgd) is used to optimize this model with a learning rate of 0.00001 and a momentum of 0.9

The specific baseline shallow UNet denoising model architecture summary can be found in Figure 12 of the appendices as well as an input, output and label example trained for 50 epochs:

2.2. Descriptor Architecture

The L2-Net architecture used in the baseline code for describing the denoised patches is an implementation of the L2-Net described from the paper: ' L2-Net: Deep Learning of Decriminative Patch Descriptor in Euclidean Space'. The architecture consists of a series of convolutional layers down sampling using stride 2 convolution and introducing Batch normalization and ReLU non-linearity after every layer.

The L2Net Descriptor model's architecture can also be found in figure 13 of the appendices but has the following specifications: [3]

The base line model uses a Tripletloss $= \frac{1}{N}\sum_{1}^{N} max(0, 1 + d(a,p) - d(a,n))$ where the euclidean distance between descriptors of the anchor and positive patch is minimized and the distance between the anchor and negative patch is maximized.

- Input x= A denoised patch$\mathbb{R}^{32x32x1}$
- Output y= A feature vector $\mathbb{R}^{128x1}$
- A dropout at the final layer of 0.3
- Once more stochastic gradient descent is used for optimization with learning rate of 0.1:

3. Evaluation of Baseline

To fully evaluate the performance of the baseline both the Denoiser and Descriptor were trained for 50 epochs on the full training data set, such that to provide a viable reference when making improvements to the model.

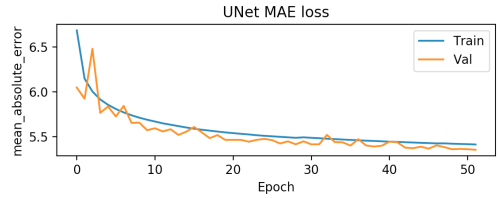

Figure 2: MAE loss of Baseline UNET

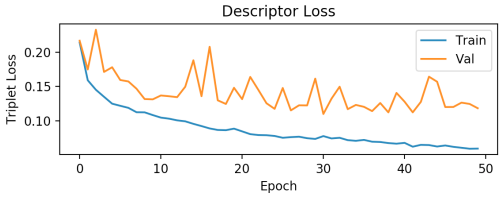

Figure 3: Triplet loss of L2Net

Figures 2 and 3 above show the learning curves after training both the UNet and L2Net respectively. Even with a very small learning rate ($l_r = 0.00001$) it is evident that the UNet's loss converges after around 25 epochs however the validation curve also converges and does not rise. This suggests the UNet is not over-fitting and given the number of epochs trained will never over fit. This is most likely due to how shallow the baseline UNet is suggesting that performance may increase if the model is deepened.

The L2Net also does not over-fit which suggests that reducing its current dropout component on its final layer may also improve performance. To provide an upper and lower bound on the performance of the L2Net, Once trained the model was evaluated against the test bench using clean and noisy images without being denoised.

Metric	Noisy	Denoised	Clean
Patch Verification	0.727	0.830	0.863
Image Matching	0.179	0.241	0.336
Patch Retrieval	0.425	0.549	0.549

Figure 4: Table of Base line test bench results

The results in Figure 4 although impressive emphasize the current trend to design descriptors that achieve high verification scores without considering the models ability to image match or patch retrieve.

4. Proposed improved approach

4.1. Denoising Model

4.1.1 Full UNet

As mentioned in the evaluation of the baseline all evidence suggests that the current base line UNet architecture is too shallow and should be deepened. Hence the first improvement made was deepening said model to include 4 convolutional encoders and 4 convolutional decoders symmetrically as well as a skip connection implemented such that to recover information from the first layers into posterior layers information as well as helping to mitigate vanishing gradients for a deeper architecture. This recovers gradient information lost during backpropagation by providing a direct path between encoder and decoder layers.

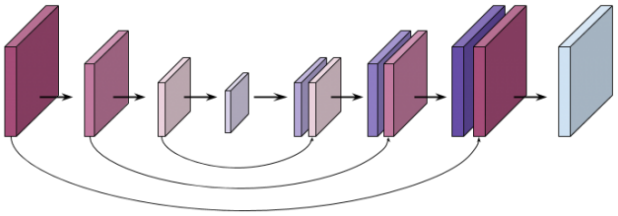

Figure 5: Diagram Illustrating Depth and Skip Connections implemented[2]

Model	Verification	Matching	Retrieval
Shallow UNet	0.788113	0.178835	0.460335
UNET	0.791656	0.180659	0.469842
DnCNN+ Transfer	0.793068	0.185465	0.478052

Figure 6: Test Bench Results for different denoisers and constant descriptors

This improved full UNet was implemented from a model provided in the tutorials and trained for 30 epochs on a subset of the data set (1500 patches). This was compared to the Shallow UNet from the baseline using the same L2Net descriptor trained for 10000 training triplets (1000 validation) for 50 epochs. As can be seen in Figure 6 this increases all three test bench scores slightly confirming the hypothesis from section 3. However, this slight increase in performance for the same number of epochs comes at the trade-off computational complexity as the new model requires significantly longer to train for a single epoch.

4.1.2 DnCNN + Transfer Learning

Instead a DnCNN model [4] has been proposed for the final approach. A DnCNN model assumes the input to the

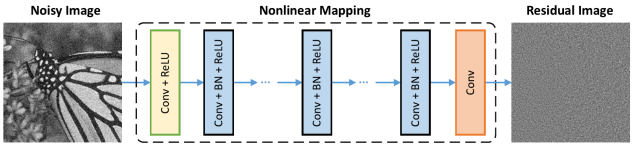

Figure 7: DnCNN Model Architecture [4]

model is a noisy patch $\mathbf{y} = \mathbf{x} + \mathbf{n}$ and attempts to train a residual mapping of the noise such that $R(\mathbf{y}) \approx \mathbf{n}$ such that the output is then $\mathbf{x} = \mathbf{y} - R(\mathbf{y})$. It consists of three separate sections. The first layer, Middle Layer and Last Layer; The first layer comprises of a Convolutional CNN layer with 64 filters $\mathbb{R}^{32x32x1}$ followed by nonlinear ReLU layer layer. The middle layers repeat for a specified depth of D-2 consisting of a CNN with 64 filters of size $\mathbb{R}^{3x3x64}$ and a batch normalization layer in between. The final layer simply contains the final Convolutional layer $\mathbb{R}^{3x3x64}$ to reconstruct the output by subtracting the result by the input.

Through the use of convolutional and ReLU the model can be trained to extract the noise from the image and hence remove it. As the Descriptor Model requires an input of the same size of that inputted into the denoise model denoising such an image can lead to boundary artifacts (anomalies in the image not present before inputted into a model). To reduce this DnCNN pads zeros after convolution ensuring all middle layers have the same size $\mathbb{R}^{32x32x1}$.
No prior Knowledge is provided as to the distribution and type of noise that is inserted into the images from NH-Patches. DnCNN is designed for such scenarios requiring the architecture to be able to form a residual model of the noise instead of a direct mapping. Batch normalization is used in conjunction with a SGD optimizer to to allow for faster and more stable convergence of this residual model.

Attached to the paper was a repository that contained a pre-trained model that was trained denoising the BSD68 dataset which was artificially induced with AWGN with $\sigma \in [0, 25]$. Transfer learning was hence used by implementing this model at the start of my denoiser freezing its trainable layers and adding a smaller Shallow DnCNN of depth 5 to the end of the model which was then trained for 30 epochs on 15000 patches. As per the previous two models this model was then evaluated on the test bench using the same descriptor and the results have been placed in figure 6. Not only are the test bench scores higher in all three tasks but the model requires significantly less time to train then that of the full UNET. The learning Curves for both models over 30 epochs can be found in Figure 14 and Figure 15 in the Appendices.

4.2. Descriptor Model

The baseline L2-Net descriptor emulates the state of the art architecture proposed in [3]. It was hence decided not to alter the architecture of the model as per the denoiser but instead focus on tuning the hyper-parameters and attempt to change triplet generation so that to improve the test bench scores.

4.2.1 Training on Clean Data

The first Improvement proposed came from training the same base line descriptor model using Noisy, Denoised and Clean patches. After which the trained models were evaluated on denoised patches using the trained shallow UNET from the baseline across all three test benches.

Trained on	Verification	Matching	Retrieval
Denoised	0.754740	0.142409	0.420243
Clean	0.791656	0.180659	0.469842
Noisy	0.726405	0.135828	0.405251

Figure 8: Test Bench Results for Different trained Data sets

Figure 8 suggests that the descriptor should be trained on clean data for the best results on denoised images. By training your descriptor on the clean images you provide the model with more distinct edges and features to extract and emphasize upon, which can then in turn be detected easier when the denoised images are received as input. This was hence implemented in the final approach

4.2.2 Triplet Generation

Several Experiments were conducted all changing the way in which triplets were selected for training. The first approach conducted consisted of limiting the selection of positive patches to be only from the subset of Tough augmented images. In theory these will be on average further away from the anchor patch and hence a greater distance between the two same patches will be minimized, speeding up the initial learning process by quickly minimizing the initial large euclidean distance between the descriptors of two patches from the same image.

As one can see using Tough Positive Triplet Generation produces a learning curve that converges below that of regular Triplet Generation. It must be noted that this should not be implemented for the entire training process as by effectively hard mining your positive patches not only do you loose generalization performance by introducing selection bias but you increase the risk of over-fitting due to the large distances that are minimized. For the final proposed approach it is suggested to use Tough Positive Triplet Generation for the start of the model for 10 epochs and regular Triplet Generation afterwards and has been implemented accordingly.

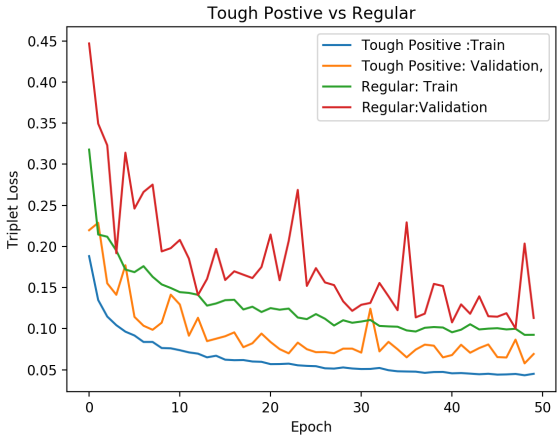

Figure 9: Tough Positive vs Regular Triplet Generation

Triplet Gen	Verification	Matching	Retrieval
Baseline	0.780932	0.190497	0.466388
Tough Positive	0.790728	0.197700	0.485347
Far Negative	0.810313	0.228031	0.519010

Figure 10: Test Bench Results for Different Triplet Generations

The final experiment attempted was purely focused on improving the matching score that has been consistently low through out all experiments. This is because currently the negative patch is not the same patch as the anchor but can be from the same image such that the distance between the descriptors of two patches from the same image are often maximized which reduces the models ability to Image Match as per 1.3. By restricting the negative to be 10000 patches away from the anchor as the patches are loaded sequentially this ensures that the negative patch belongs to another image.

4.3. Results of Full Approach

Final Model	Verification	Matching	Retrieval
Baseline	0.780476	0.168586	0.461546
Proposed	0.800352	0.171257	0.496219

Figure 11: Final Results on Reduced Data Set

The results in Figure 11 follow the proposed approach but on a subset of the data. For the descriptor Tough Positive Triplet generation of clean patches was used for the first 10 epochs followed by regular Triplet generation for 35 epochs and finally 5 epochs of Far Negative Generation.

References

[1] Balntas, V., Lenc, K., Vedaldi, A. and Mikolajczyk, K. (2017). HPatches: A benchmark and evaluation of hand-crafted and learned local descriptors

[2] Ronneberger, O., Fischer, P. and Brox, T. (2015). U-Net: Convolutional Networks for Biomedical Image Segmentation

[3] Tian, Y., Fan, B. and Wu, F. (2017). L2-Net: Deep Learning of Discriminative Patch Descriptor in Euclidean Space

[4] Zhang, K., Zuo W., Chen, Y., Meng, D. and Zhang, L. (2016). Beyond a Gaussian Denoiser: Residual Learning of Deep CNN for Image Denoising

5. Appendices

Project Zip File can be found at: `https://drive.google.com/file/d/1MAZA_kLmsuO1oAnp80u18vY88hLMkd5t/view?usp=sharing` To .pynb file will run all of the tested and train the final model. The following files must be uploaded to working directory udescriptorbaselinecleansmall.h5 model.h5 read_data_farnegative.py read_data_tough_positive.py UNETsmall.h5

Layer (type)	Output Shape	Param #	Connected to
input_4 (InputLayer)	(None, 32, 32, 1)	0	
conv2d_16 (Conv2D)	(None, 32, 32, 16)	160	input_4[0][0]
max_pooling2d_4 (MaxPooling2D)	(None, 16, 16, 16)	0	conv2d_16[0][0]
conv2d_17 (Conv2D)	(None, 16, 16, 32)	4640	max_pooling2d_4[0][0]
up_sampling2d_4 (UpSampling2D)	(None, 32, 32, 32)	0	conv2d_17[0][0]
conv2d_18 (Conv2D)	(None, 32, 32, 64)	8256	up_sampling2d_4[0][0]
concatenate_4 (Concatenate)	(None, 32, 32, 80)	0	conv2d_16[0][0] conv2d_18[0][0]
conv2d_19 (Conv2D)	(None, 32, 32, 64)	46144	concatenate_4[0][0]
conv2d_20 (Conv2D)	(None, 32, 32, 1)	577	conv2d_19[0][0]

Total params: 59,777
Trainable params: 59,777
Non-trainable params: 0

Figure 12: Table of Baseline Denoise Model Architecture

Layer (type)	Output Shape	Param #	Connected to
a (InputLayer)	(None, 32, 32, 1)	0	
p (InputLayer)	(None, 32, 32, 1)	0	
n (InputLayer)	(None, 32, 32, 1)	0	
sequential_1 (Sequential)	(None, 128)	1336928	a[0][0] p[0][0] n[0][0]
lambda_1 (Lambda)	(None, 1)	0	sequential_1[1][0] sequential_1[2][0] sequential_1[3][0]

Total params: 1,336,928
Trainable params: 1,336,032
Non-trainable params: 896

Figure 13: Table of Baseline Descriptor Model Architecture

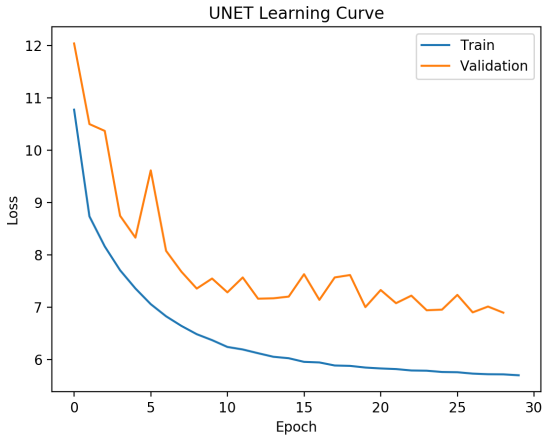

Figure 14: Deep UNET learning Curve

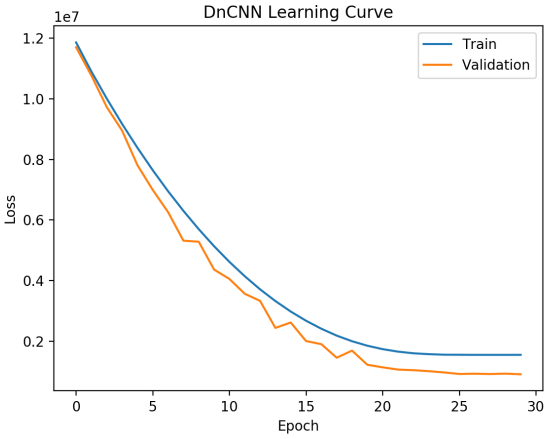

Figure 15: DnCNN learning Curve