HW1

Gabriel ROMON

1 DCT denoiser

1. The conditional density of X given $Y=y$ is given by $f_{X|Y=y}(x) = \dfrac{f_{Y|X=x}(y)f_X(x)}{f_Y(y)}$.
   Since $Y = X+B$, $Y|X=x \sim \mathcal{N}(x,\sigma^2)$ hence $f_{X|Y=y}(x) = \frac{1}{\sqrt{2\pi\sigma^2}}\exp(-\frac{1}{2\sigma^2}(y-x)^2)$. Thus

   $$f_{X|Y=y}(x) \propto \exp(-\frac{1}{2\sigma^2}(y-x)^2 - |x|)$$

   Computing the maximum of $f_{X|Y=y}(x)$ is consequently equivalent to computing

   $$\arg\max_{x\in\mathbb{R}} \exp\left(-\frac{1}{2\sigma^2}(y-x)^2 - |x|\right) = \arg\min_{x\in\mathbb{R}} \frac{1}{2\sigma^2}(y-x)^2 + |x|$$

   Let $\varphi : x \mapsto \frac{1}{2\sigma^2}(y-x)^2 + |x|$. Studying the derivative of φ on $\mathbb{R}^+$ and $\mathbb{R}^-$ shows that if $y \geq \sigma^2$, φ is increasing over $[y-\sigma^2, \infty)$ and decreasing on the complement. If $y \leq -\sigma^2$, φ is decreasing over $(-\infty, y+\sigma^2]$ and increasing on the complement. Therefore, if $y \geq \sigma^2$, the minimum is attained at $y-\sigma^2$, if $y \in (-\sigma^2, \sigma^2)$, the minimum is attained at 0 and if $y \leq -\sigma^2$, the minimum is attained at $y+\sigma^2$. This rewrites more compactly as

   $$x^* = \left(1 - \frac{\sigma^2}{|y|}\right)^+ y$$

   which corresponds to *soft-thresholding* of y with threshold σ^2.

2. The code in `DCT_denoiser` performs denoising on an image corrupted by Gaussian noise with variance σ^2.
   First, a tensor `D` of shape $N\times N\times N^2$ is built such that the `D[:,:,k]` are the inverse discrete cosine transforms of the N^2 elementary matrices of $\mathbb{R}^{N\times N}$. More precisely there is some bijection $\varphi : [\![1,N]\!]^2 \to [\![1,N^2]\!]$ such that $\mathrm{D}[:,:,\varphi(i,j)]$ is the 2D inverse DCT of the elementary matrix E_{ij}. Consequently, $\mathrm{D}[m,n,\varphi(i,j)] = \alpha_i\alpha_j \cos\left(\frac{\pi i}{N}\left(m+\frac{1}{2}\right)\right)\cos\left(\frac{\pi j}{N}\left(n+\frac{1}{2}\right)\right)$ where the α_i are the usual constants defining the type 2 orthonormal DCT.

   For some fixed (i,j), $\mathrm{D}[:,:,\varphi(i,j)]$ is a kernel/filter. For each $N\times N$ patch X of the noisy image, `conv2D` computes the correlation between the filter $\mathrm{D}[:,:,\varphi(i,j)]$ and X:

   $$\sum_{m=0}^{N-1}\sum_{n=0}^{N-1} \mathrm{D}[m,n,\varphi(i,j)]X_{mn} = \alpha_i\alpha_j \sum_{m=0}^{N-1}\sum_{n=0}^{N-1} X_{mn}\cos\left(\frac{\pi i}{N}\left(m+\frac{1}{2}\right)\right)\cos\left(\frac{\pi j}{N}\left(n+\frac{1}{2}\right)\right)$$

   which is precisely coefficient (i,j) in the 2D DCT of the patch X.

   This is done for every (i,j) and the resulting coefficients undergo *hard-thresholding*: all coefficients with absolute value larger than 3σ are set to 0. This is different from the soft-thresholding used in 1. Next, the algorithm returns to the spatial domain by convolving the resulting tensor with the same filters (note that this an actual convolution so filters have to flipped beforehand, hence the command `np.fliplr(np.flipud(D))`).

3. In order to learn the threshold parameter s, one has to define a loss function and get some training data. Regarding the loss, a possible choice is the squared euclidean norm on the flattened images $\ell(y,\hat{y}) = \sum_i (y_i - \hat{y}_i)^2$. However, the relation between ℓ and s is hard to write down explicitely (even as a composition of several simple functions) thus making gradient descent difficult.

4. Computing the inverse DCT of the elementary matrices has total cost $O(N^2 \times N \log N) = O(N^3 \log N)$. For an $m \times n$ input, the correlation operation has cost $O(N^2 \times (m-N)(n-N) \times N^2) = O(mnN^4)$ since for each filter, there are $\sim (m-N)(n-N)$ patches and $O(N^2)$ operations to compute on each. The second convolution has the same cost $O(mnN^4)$, so the number of operations per pixel is $O(N^4)$.

2 DnCNN

5. The architecture used in the practical session is identical to the one presented in the seminal paper on DnCNN [1]: the network has depth 17, the first layer is Conv+ReLU (64 filters of size $3 \times 3 \times 1$), each layer from №2 to №16 is made of Conv+BatchNorm+ReLU (64 filters of size $3 \times 3 \times 64$) and the final layer is convolutional (1 filter of size $3 \times 3 \times 64$).
Layer №1 has $3 \times 3 \times 1 \times 64 + 64$ parameters, while layers from №2 to №16 each have $3 \times 3 \times 64 \times 64 + 64$ parameters and the last has $3 \times 3 \times 64 \times 1 + 1$, for a grand total of ~ 555.000 parameters (weights and biases).

6. For an $m \times n$ input, convolutions in layer №1 take $\sim 64 \times 3^2 \times mn$ operations, convolutions in each of the layer №2 to №16 take $\sim 64^2 \times 3^2 \times mn$ and for the last layer $64 \times 3^2 \times mn$. For the network used in the practical session, the number of operations per pixel is ~ 554.000.

   Let D denote the depth of the network ($D = 17$ in the implementation). Since the network stacks convolutional layers of 3×3 filters, the second layer has a 3×3 view of the first one, hence a 5×5 view of the input, and so on with the third layer which has a 7×7 view of the input. The final layer has a receptive field of size $(2D+1) \times (2D+1)$, which is exactly what [1] states. If one wants to compare DCN denoising with DnCNN, one should thus compare the patch size N with $2D+1$ (and not with 3, the size of the filters).
   Throwing away constants, the total number of operations is thus $O(Dmn)$, which makes $O(D)$ operations per pixel. Given what has been said above, N can be compared with D ($2D+1$ actually but this is harmless) and the complexity has changed from $O(N^4)$ to $O(D)$, a significant improvement, but one should not forget that DnCNN needs to be trained on a large dataset beforehand.

7.

Figure 1: Denoised images obtained with each method

References

[1] Kai Zhang, Wangmeng Zuo, Yunjin Chen, Deyu Meng, and Lei Zhang. Beyond a gaussian denoiser: Residual learning of deep cnn for image denoising. *IEEE Transactions on Image Processing*, 26(7):3142–3155, 2017.