TP1 Deep learning for image restoration and synthesis

Imane SI SALAH

January 27, 2024

1 DCT denoiser

We consider a real random variable X whose distribution is drawn form $e^{-|x|}$ and the obeservation $Y = X + B$ where B is zero-mean Gaussian noise with variance σ^2

1. We show that if the observation is known, $(Y = y)$ then the most probable value of $X = x$ is the one that minimizes
$$\arg\min_x \left(|x| + \frac{1}{2\sigma^2}(x-y)^2 \right)$$
we use MAP estimation
given that x is a realisation of the rv X with prior $p_X(x)$ and y is the realization of rv Y, the posterior probability can be written as
$$p_{X|Y}(x/y) = \frac{p_{Y|X}(y/x)p_X(x)}{p_Y(y)}$$
then we note that maximizing the posterior probability is equivalent to maximizing $p_{Y|X}(y/x)p_X(x)$ since $p_Y(y)$ is fixed given the data.
then we have $Y = X + B$ also $P(B) = e^{-\frac{B^2}{2\sigma^2}}$
therefore: $p_{Y|X}(y/x) = e^{-\frac{(y-x)^2}{2\sigma^2}}$
then we get $p_{X|Y}(x/y) = e^{-\frac{(y-x)^2}{2\sigma^2}} e^{-|x|}$
we get the log likelihood as follows:
$$\log(p_{X|Y}(x/y)) = -\frac{(y-x)^2}{2\sigma^2} - |x|$$
however we kow that maximizing the log likelihood is equivalent to minimizing the negative log likelihood, we obtain the objective function is:
$$\arg\min_x \left(|x| + \frac{1}{2\sigma^2}(x-y)^2 \right)$$
we solve this optimization problem by setting the first derivative of the objective to zero, we just study the case where $y > 0$:

- if $x \geq 0$:
$f'(x) = 1 + \frac{(x-y)}{\sigma^2} = 0$ gives $x = y - \sigma^2$, to ensure positivity of x we need $y \geq \sigma^2$
- if $x < 0$:
$f'(x) = -1 + \frac{(x-y)}{\sigma^2} = 0$ gives $x = y + \sigma^2$ to ensure negativity of x we need $y < -\sigma^2$ however this is not valid since $y > 0$, therefore x is set to 0

so to summarize:
$$x = \begin{cases} y - \sigma^2 & \text{if } y > \sigma^2, \\ 0 & \text{otherwise.} \end{cases}$$

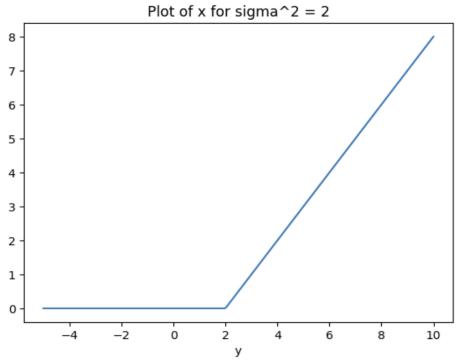

Figure 1: x as a function of y

2. The network in DCT_denoise involves two convolutional layers, the first is initialized with weights **W1** from generated IDCT basis in **vects**, it converts the image to the DCT domain, and the second layer performs the inverse process to recover the image from the DCT domain. in addition it accounts for the border effect due to the convolution.

3. Yes, the threshold can be learned by gradient descent, by including to the overall graph of computation in the initialization, computing the a loss and performing backward propagation.

4. from the first convolution layer we have kernel sizes of $(N \times N)$ and the number pf channels is N^2 so at the application of each filter, a pixel will undergo N^2 multiplications and $N^2 - 1$ additions so for this layer the number of operations is approximately $2N^4 - N^2$, then for the second convolution layer we have kernel size of $(7 \times 7)$the number of output channels is 1, so the number of multiplications and additions per pixel is $49N^2 + 48N^2$ therefore the total number of operations per pixel can be approximated to $\mathbf{2N^4 + 96N^2}$. for $N = 7$ we get 7105 operations per pixel

5. The best threshold for a noise level of 25 is 20.32
the residual error versus the ratio is plotted as follows:

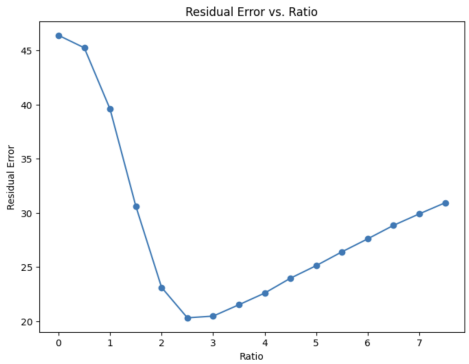

Figure 2: residual error versus the ratio

and the result of the denoiser at ratio 2.5 is:

2 FFDNET

6. result of the FFDNET model: we clearly see that the ouptut image of the FFDnet model is better than the DCT denoiser one, in fact there is less blur and the edges seem sharper

7. number of operations per pixel:from the paper,we have in the FFDnet there are 15 convolutional layers, 64 feature maps for each and kernel size is (3×3) similar to the analysis we performed

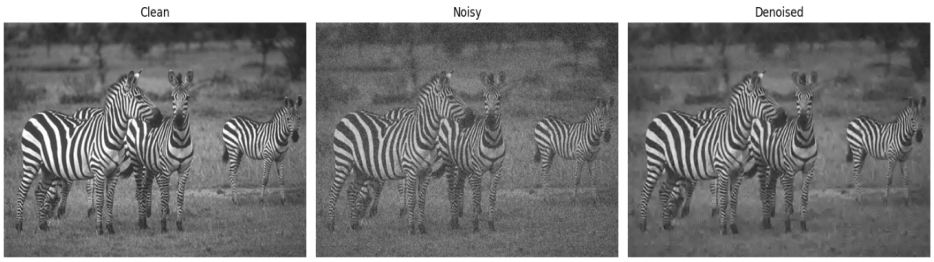

Figure 3: clean, noisy and denoised image fro DCT denoiser (nl=25)

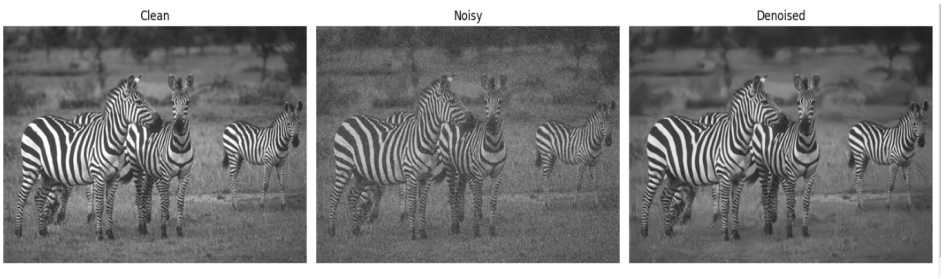

Figure 4: clean, noisy and denoised image for FFDNET (nl=25)

above, at the application of one filter to the image, we have 9 multiplication and 8 additions at the level of each pixel, resulting in 17 operations, so for a single layer we have $64 \times 17 = 1,088$ operations, therefore for the whole model we get: $1,088 \times 15 = 16,320$, this much more than the number of operations obtained for the DCT denoiser.

3 DRN

8. architecture of the super resolution network: the residual dense network consists of four main parts, first a Shallow feature extraction which uses two convolutional layers to extract features from an input low resolution image, then set of residual dense blocks (RDB) where each RDB contains a set of densely connected convolutional layers ie. each layer receives feature maps from all previous layers, then the Dense Feature Fusion part where all local features obtained from the RDBs are fused to form global features, finally the upsamling Net (UPNet) where the network upsamles the feature maps to the desired high resolution size

9. number of operation for execution:

   - shallow feature extraction step: for the first layer we have $H \times W \times 3 \times 3 \times 1 \times 64$ (we consider one input channel for gray images) and for the second layer we have $H \times W \times 3 \times 3 \times 64 \times 64$
   - RDBs: each RDB has G filters and ther are D RDBs so the number of operations can be computed by: $H \times W \times 3 \times 3 \times G \times (C \times D + 64)$ wher C is the number of conv layers in one RDB.
   - RDBs: Each RDB has G filters, and there are D RDBs, so the number of operations for the convolutions inside each RDB can be computed by summing up the operations across all layers. For the i-th layer in an RDB, the number of input feature maps is $64 + (i - 1) \times G$, assuming the input feature maps from the SFENet plus the output feature maps of all preceding layers within the RDB are concatenated. Therefore, the total number of operations for one RDB is:

$$\sum_{i=1}^{C} H \times W \times 3 \times 3 \times G \times (64 + (i-1) \times G)$$

and for all RDBs, it would be:

$$D \times \left(\sum_{i=1}^{C} H \times W \times 3 \times 3 \times G \times (64 + (i-1) \times G) \right)$$

where C is the number of convolutional layers in one RDB.

- Dense feature fusion (DFF) and up-sampling net (UPNet

10. zoom-out kernel used during the network's learning phase: In the laboratoty work, in the apply_model() function, three downsampling methods are mentionned to test the network, bicubic, decimation and local mean.

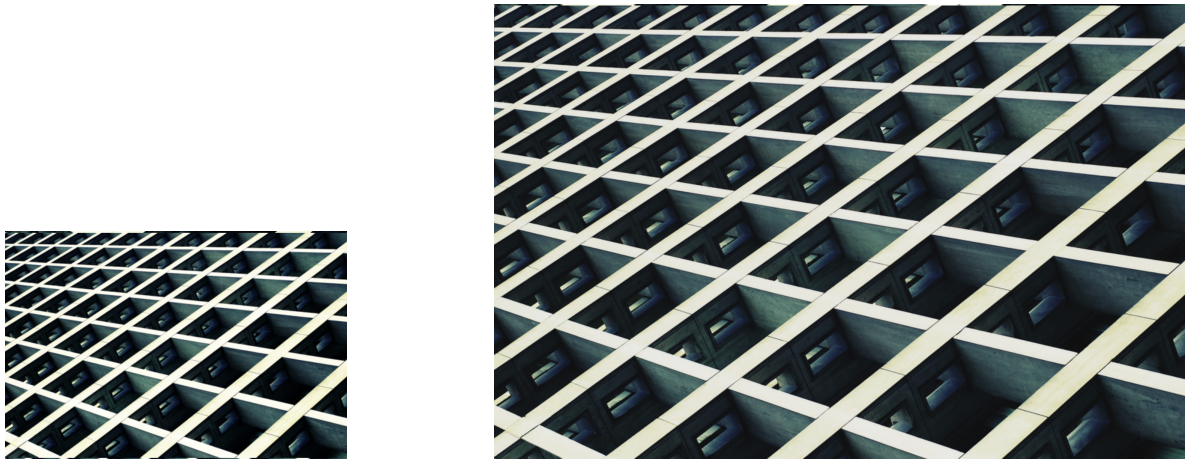

Figure 5: Left:Downsampled image: Bicubic. Right: Zoomed Image from model.

Figure 6: Left:Downsampled image: local mean. Right: Zoomed Image from model.

we see that the best result is obtained when using the bicubic zoom-out method, it gives a better visual result. this method is based on bicubic interpolation, this leads to smoother transitions in color and intensity, on the other hand we can explain the bad rendering of the decimation method to the fact that it may lead to aliasing since it just discards pixels, finally the local mean does better since it filters out noise keeping only essential features.

11. The zoom-out kernel used: we can see this from the apply_model() function, for the decimation downsamplign method, it takes one pixel each 'scale' number of pixels, and the local mean method takes the average of blocks of 'scale x scales' pixels so we can deduce that the kernel for this method is 'scale x scale'

Figure 7: Left:Downsampled image: ldecim. Right: Zoomed Image from model.