COMPUTER VISION - ASSIGNMENT 2 - REPORT

2) MEAN SHIFT ALGORITHM

REMARK:

For all the functions, I have employed methods belonging to the **NUMPY** library. I have preferred to use NUMPY instead of TORCH as I have observed, after several attempts, that NUMPY performs better in terms of speed. Moreover, for the sake of consistency, I have used NUMPY both in the naive and the accelerated version. In that way, I can fairly compare the performances of the two versions.

2.1) In the distance function, I have calculated a vector which keep track of the distances between x and X, using the definition of euclidean norm. Using the numpy.linalg.norm function would have been more elegant, but I have observed that the manual implementation performs slightly better.

2.2) In the gaussian function, I have computed each weight according to a “Gaussian Kernel” with bandwidth 2.5. It is worth noticing that assigning the weights in this way penalizes points further away from x.

2.3) In the update function, I have updated x with the weighted averages of X. The weights used were of course those computed by the previous function. Once again, using functions like numpy.dot and numpy.average would have been smoother and more elegant, but I have observed that manual manipulation of the tensors performs better in terms of speed.

I have carefully avoided using for loops in the functions, as the performances steeply decline. In order to manually perform the distance and the update, I have often used a syntax like weight[:,None], X[None,:] which is useful to add an extra dimension to the tensors.

2.4) In the accelerated version of the Mean-Shift algorithm, I had to redefine the distance and the update function as well as the meanshift_step_batch. In this function, I have defined the constants BATCH_SIZE = 49, BATCH_NUMBER = 75. Basically the mean shift algorithm now runs one batch at a time. It is worth noticing that the constants BATCH_SIZE, BATCH_NUMBER are not independent from each other. In fact, BATCH_SIZE*BATCH_NUMBER = 3675 which equals the total number of pixels. I have run the script with several values of BATCH_SIZE and BATCH_NUMBER, finding out that 49 and 75 are optimal values in terms of speed.

After several attempts the average running time of both implementations are:

RUNNING TIME (CPU) of **NAIVE IMPLEMENTATION**: **12.9 / 13 seconds**
RUNNING TIME (CPU) of **ACCELERATED IMPLEMENTATION**: **3.8/ 4 seconds**

3) SEGNET

After having completed the model of the Segnet, I have trained and validated it. I have obtained a validation score of 0.871.