**Image Restoration with Deep Learning (20pts + 10pts)**

Based on the solutions from HW-4, we will study two problems in deep learning based image restoration/denoising. Consider an image corrupted by both a PSF (blur in this case) and additive noises,

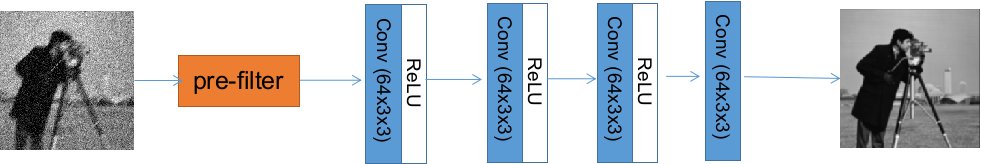
equation

Where x is the clean image, h is the point spread function (PSF) that creates input dependent corruptions, and n is the additive Gaussian noise. An example is illustrated below, with I1 being blurred with angle 45o length 11 motion blur, and I2 blurred with the same motion blur with additive noise with variance 0.004;



To have a fair comparison, we utilize VDSR like residual learning network with fixed depth of 4 blocks of [CONV(64 x 3 x 3) + ReLU] operations, while focusing on the research problems of,

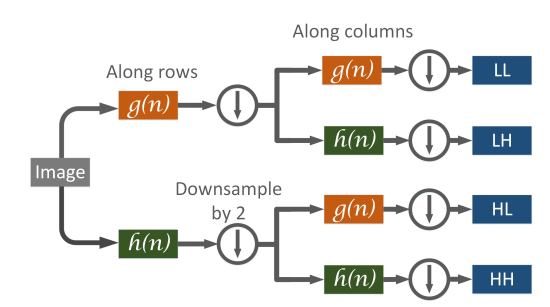
1. Input noisy image pre-filtering [20 pts]:



Develop pre-filtering schemes on the input, before training for the deep residual network denoising, and see if prefiltering can improve the performance. Suggested method: bilateral filtering, try a number of different range and value kernel size and analyze the results.

1. Target image decomposition [10pts]:

To directly learn the target image is kind of difficult. Instead, the ground truth images are decomposed using wavelets into subimages of {LL, LH, HL, and HH} bands:



Use the same 4-block residual learning framework to learn these sub-images from noisy input, and report the PSNR for each subband, as well as overall re-combined images. [Hint: use matlab dwt2() and idwt2(), or Python equivalent, an example matlab code is attached]

|  |
| --- |
| function [s]=getDwtImages(im, wfname, L)  dbg=1;  if dbg  im = im2double(imread('cameraman.tif'));  wfname ='db5';  L=2;  end  % decomposition filters: low pass and high pass  [lpf, hpf]=**wfilters**(wfname, 'd');  % level 1 subimages  [s{1}, s{2}, s{3}, s{4}]=**dwt2**(im, lpf, hpf,'mode','symh');  k=4; L=L-1;  while (L>0)  if dbg  figure(30+L); title(sprintf('level: %d', L));  for j=1:4 subplot(2,2,j); imshow(s{k-4+j}); end  end  im0 = s{k-4+1};  [s{k+1}, s{k+2}, s{k+3}, s{k+4}]=**dwt2**(im0, lpf, hpf,'mode','symh');  k=k+4; L=L-1;  end  % plot last level  if dbg  figure(30+L); title(sprintf('level: %d', L));  for j=1:4 subplot(2,2,j); imshow(s{k-4+j}); end  end  % synthesize  if dbg  % reconstruction filters: low pass and high pass  [lpf\_r, hpf\_r]=**wfilters**(wfname, 'r');  im1 = **idwt2**(s{1}, s{2}, s{3}, s{4}, lpf\_r, hpf\_r);  figure(41); imshow(im1);    figure(42);  hold on; grid on; plot(lpf, '.-b'); plot(hpf, '.-k'); plot(lpf\_r, '.-r'); plot(hpf\_r, '.-m');  legend('lpf', 'hpf', 'lpf\_r', 'hpf\_r');  end  return; |

The data set to work on are two aerial remote sensing data set:

1. AID data set:https://captain-whu.github.io/AID/



1. NWPU RESISC 45: <http://www.escience.cn/people/JunweiHan/NWPU-RESISC45.html>

