CS 663 - Blind Super Resolution with Iterative Kernel Correction

Course Project

Jay Sawant
Priyanka Bagade
Harshit Shrivastava
Kaustubh Bhargao

Blind Super-Resolution With Iterative Kernel Correction

- Most of Deep Learning Blind Super Resolution methods assume that the blur kernel during downsampling is predefined/known (usually Bicubic)
- This paper implements an Iterative Kernel Correction (IKC) method for blur kernel estimation in Blind SR problem, where the blur kernels are unknown.
- Kernel Mismatch causes artifacts due to over-smoothening/sharpening and this information can be used to correct the inaccurate blur kernel
- The authors have further proposed an architecture using Spatial Feature Transform (SFT) layers to take care of multiple blur kernels. (SFTMD)
- Experimentally it was seen that IKC + SFTMD provide good SR Results and state of the art performance.

Components present in network architecture

- SFTMD
- Corrector Network
- Predictor Network

SFTMD Architecture

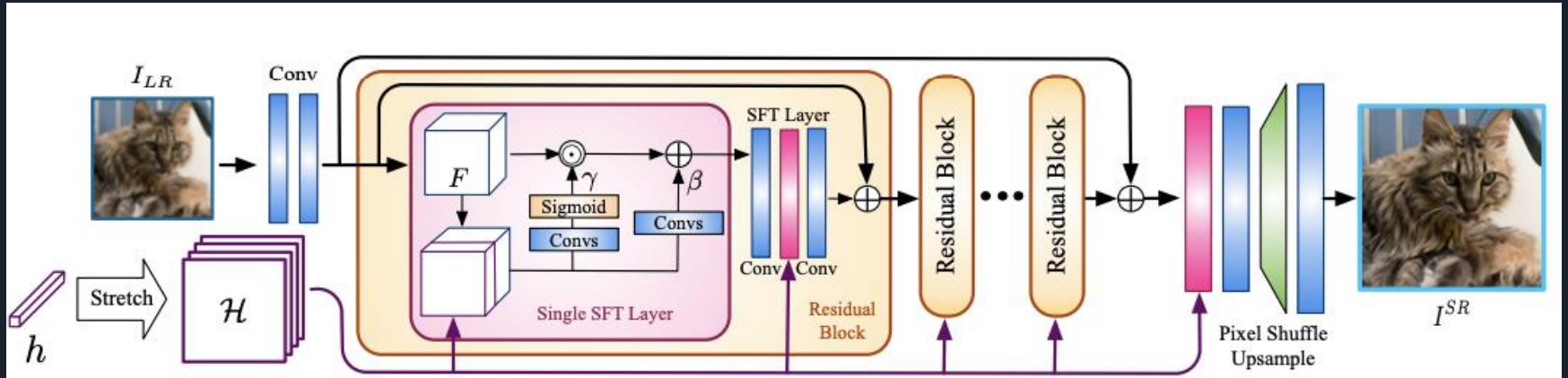


Figure 4. The architecture of the proposed SFTMD network. The design of the proposed SFT layer is shown in pink box.

SFTMD Network

1. The SFTMD network takes a low resolution image and a dimensionally reduced kernel and outputs the Super resolution image given the scale factor.
2. We have used 16 residual blocks of 2 SFT and 2 Conv layers and depth of the feature maps is chosen to be 64
3. The Kernels used for SFTMD training (to create low resolution images) are 21x21 isotropic Gaussian Kernels with their standard deviations (sigma) lying in the interval [0.2,4]
4. The scale factor used for training and testing is 4.
5. The dataset used for training and testing is **DIVerse2K** (https://data.vision.ee.ethz.ch/cvl/DIV2K/) which contains 800 training 2K images and 100 2K validation images

Training of SFTMD

1. Each 2K image was read and a random patch of 256x256 was selected from the image, it was blurred selecting a random gaussian kernel of standard deviation between 0.2 and 4 and downsampled by a factor of 4 using Bicubic interpolation along with addition of some gaussian noise to obtain the Low resolution image
2. This Low resolution image was passed through the model along with dimensionally reduced kernel (which was used for blurring) and loss function used was MSE
3. The SFTMD model was trained for only 1000 epochs instead of 500000 epochs due to limited resources. We were able to clearly see the difference between the low resolution image and the output from 100th epoch onwards

Predictor Network

1. We take the Low Resolution Image I^{LR} as the input and try to predict the kernel ‘h’
2. We use 4 Convolutional layers with a Leaky ReLU activation function and a global average pooling layer
3. The convolution layers give the estimation of the kernel h spatially and form the estimation maps
4. Then the global average pooling layer gives the global estimation by taking the mean value spatially
5. The output kernel from here is then passed on to the corrector network for further corrections and bring it as close as possible to the actual kernel

Corrector Network

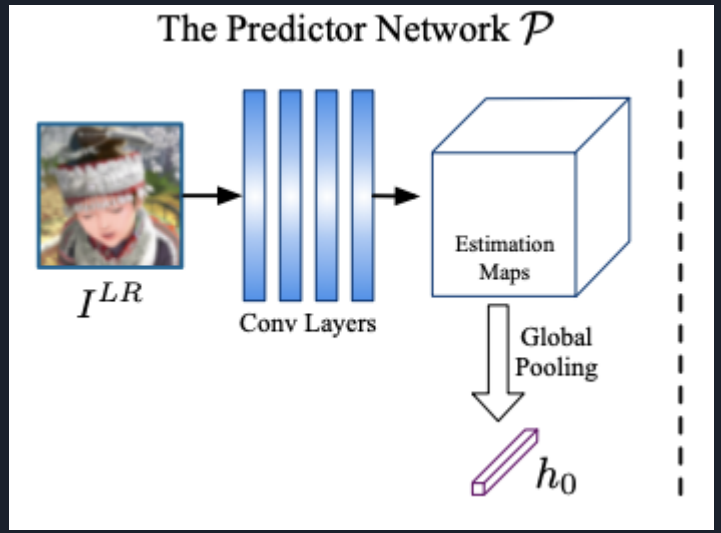
1. We take the Super Resolution Image I^{SR} and the kernel estimated by the predictor as Inputs
2. The new kernel can be obtained by solving the following optimization problem, which to put in simple terms is to predict corrections or Δh which can be added to the original kernel h so that it is one step closer to the ground truth kernel

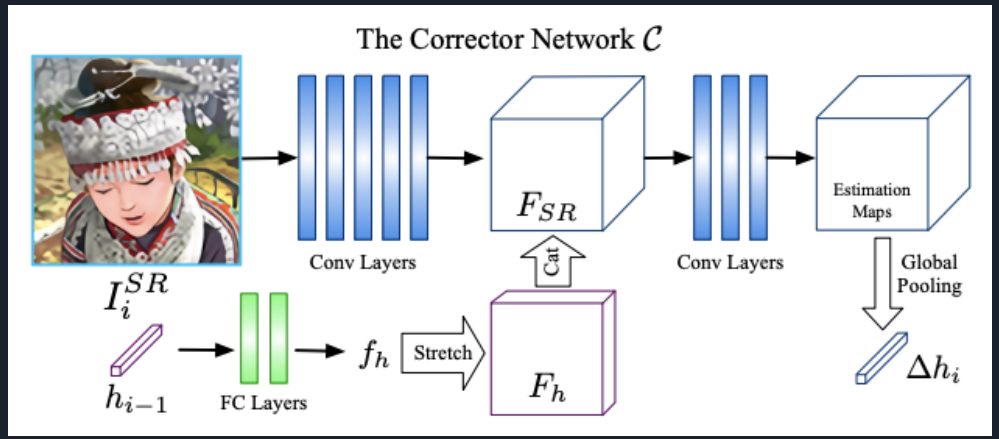
$$\theta_C = \arg\min_{\theta_C} \|k - (C(I^{SR}, h; \theta_C) + k')\|_2^2.$$

1. The high resolution image obtained through sftmd is passed through 5 convolutional layers further to generate F_sr
2. The previously estimated kernel is passed through two fully connected layers. We then stretch the output vector f_h to F_h such that it has the same H and W as F(sr).
3. Finally, the F_h and F_{SR} are concatenated and this is used to predict the Δh_i (Spatially)

4. A global pooling operation gives us the global estimation of Δh

Predictor and Corrector Network Architectures





Training of Predictor

1. We used the 800 images from the DIVerse2K dataset for training purpose
2. Each 2K image was read and a random patch of 256x256 was selected from the image, it was blurred selecting a random gaussian kernel of standard deviation between 0.2 and 4 and downsampled by a factor of 4 using Cubic interpolation along with addition of some gaussian noise to obtain the Low resolution image (same as sftmd)
3. This Low resolution image was passed through the model to predict the kernel used.
4. The loss function used was MSE

Training of Corrector

1. We used the 800 images from the DIVerse2K dataset for training purpose
2. The kernel predicted by the predictor was passed through the the corrector as an input along with the high resolution image obtained using sftmd (and the predicted kernel).
3. The step of passing the HR image obtained from sftmd and the predicted kernel through the corrector was carried out iteratively (Iterative Kernel Correction step) 7 times
4. The loss function used was Mean Squared loss

Algorithm of Iterative Kernel Correction

Algorithm 1 Iterative Kernel Correction

Require: the LR image I^{LR}

Require: the max iteration number t

$h_0 \leftarrow \mathcal{P}(I^{LR})$ (Initialize the kernel estimation)

$I_0^{SR} \leftarrow \mathcal{F}(I^{LR}, h_0)$ (The initial SR result)

$i \leftarrow 0$ (Initialize counter)

while $i < t$ **do**

    $i \leftarrow i + 1$

    $\Delta h_i \leftarrow \mathcal{C}(I_{i-1}^{SR}, h_{i-1})$ (Estimate the kernel error using the intermediate SR results)

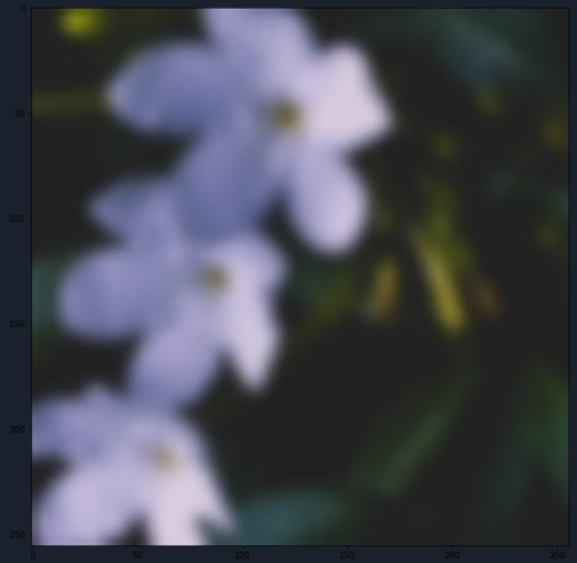
    $h_i \leftarrow h_{i-1} + \Delta h_i$ (Update kernel estimation)

    $I_i^{SR} \leftarrow \mathcal{F}(I^{LR}, h_i)$ (Update the SR result)

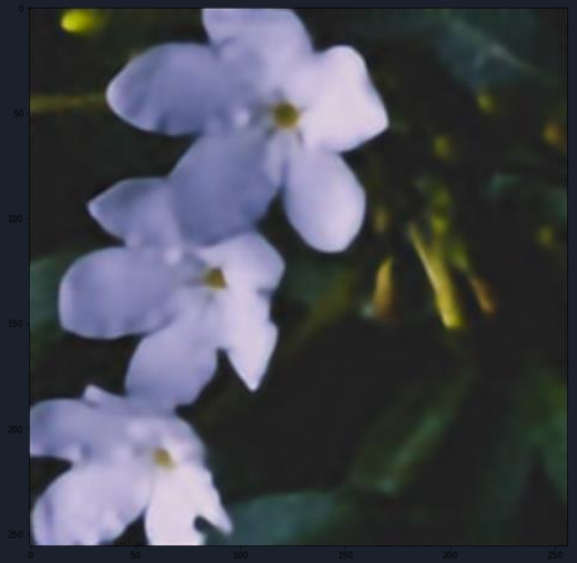
return I_t^{SR} (Output the final SR result)

Sample output from SFTMD

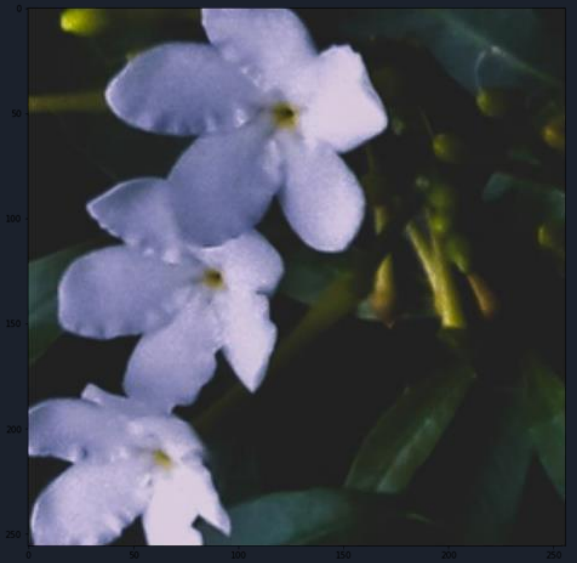
Low Resolution Image



Predicted Image

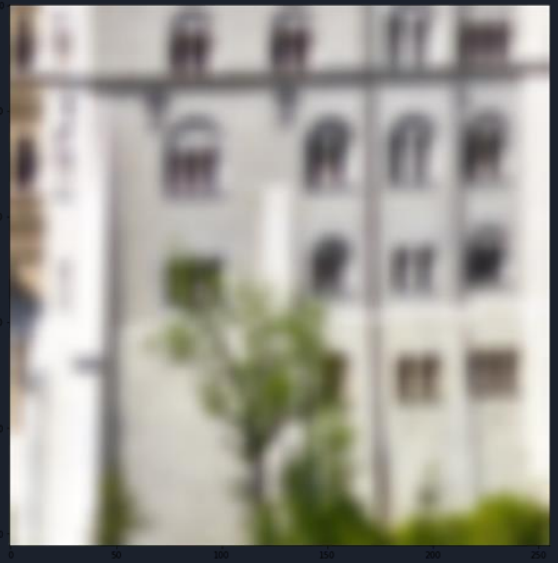


Actual Image

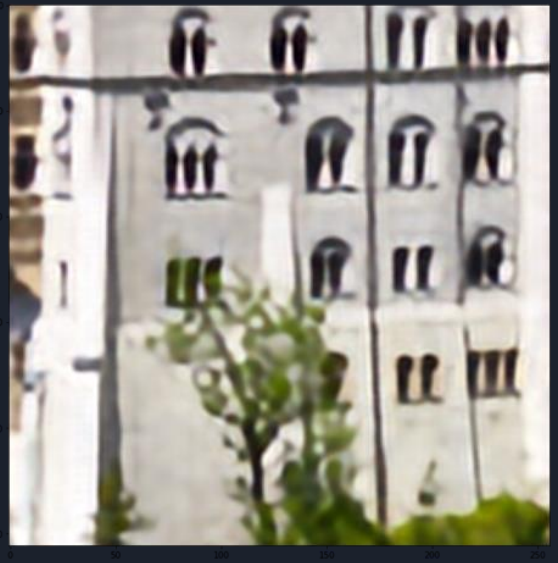


Sample output from SFTMD

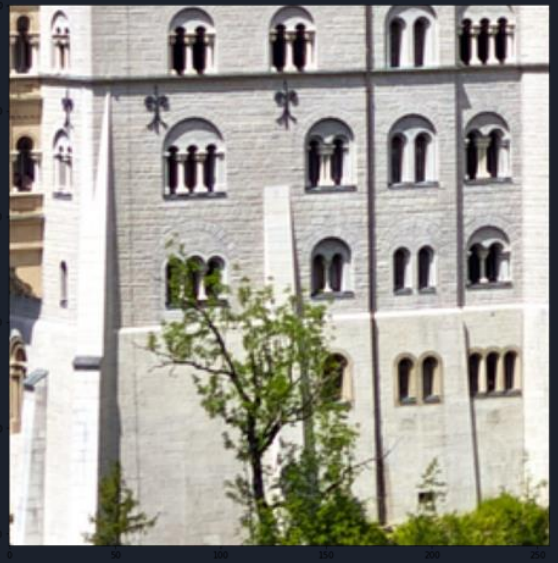
Low Resolution Image



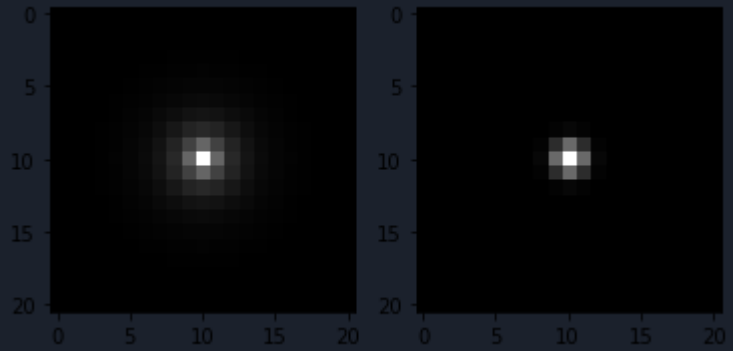
Predicted Image



Actual Image

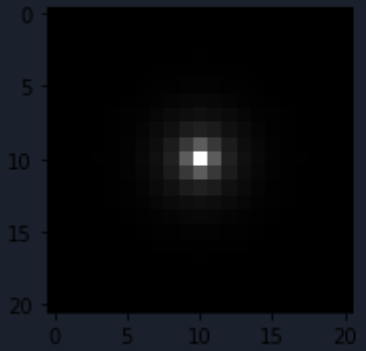


Sample output from Predictor

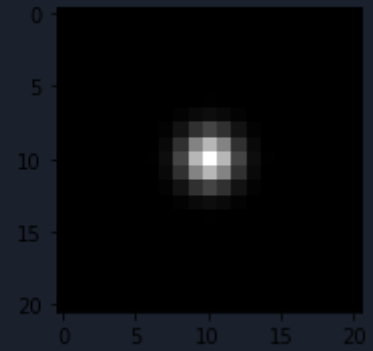


Predicted Kernel

Actual Kernel

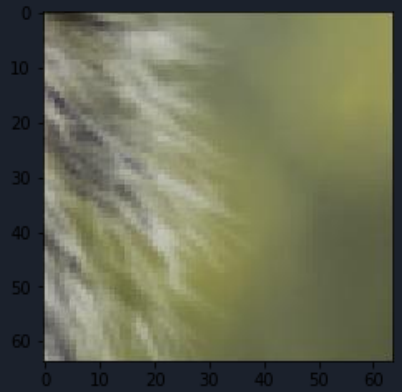


Predicted Kernel

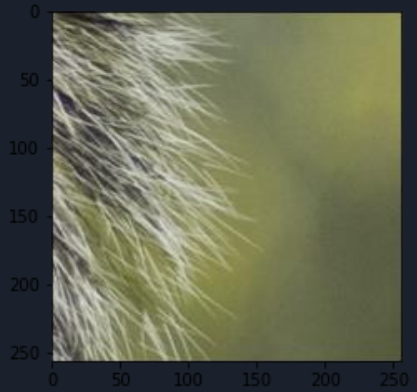


Actual Kernel

Ground Truth Images

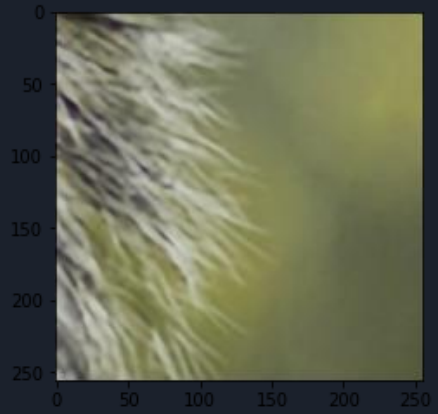


Given LR Image

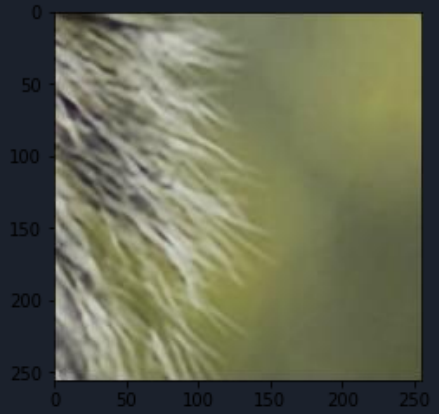


Given HR Image

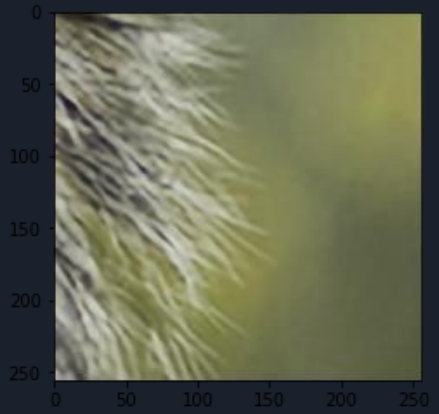
Sample output from Corrector



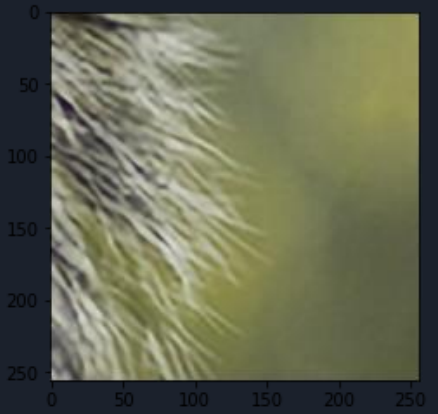
Iteration = 1



Iteration = 2



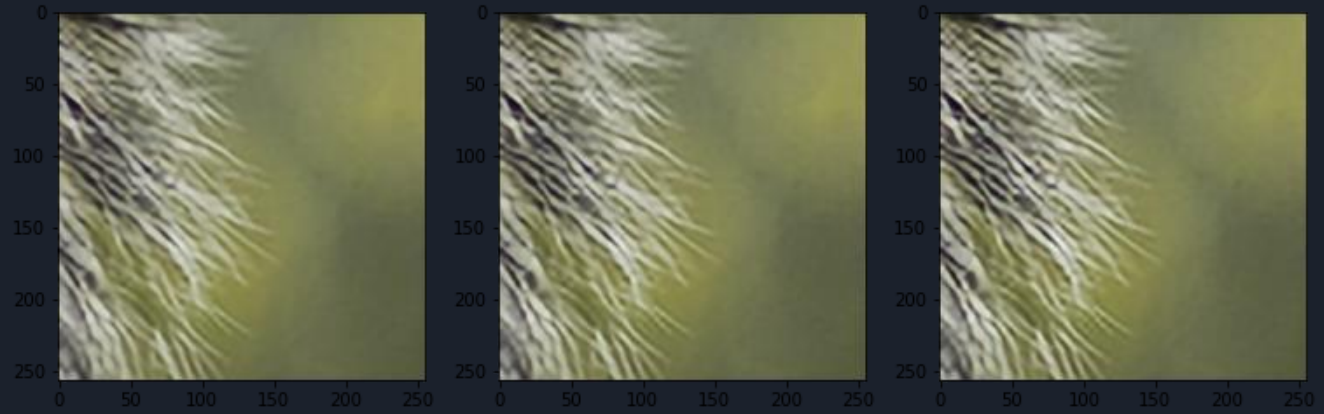
Iteration = 3



Iteration = 4

Images obtained after successive iterative kernel correction

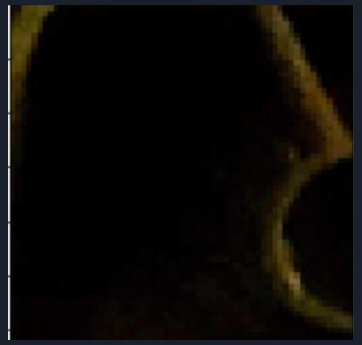
Sample output from Corrector



Iteration = 5 Iteration = 6 Iteration = 7

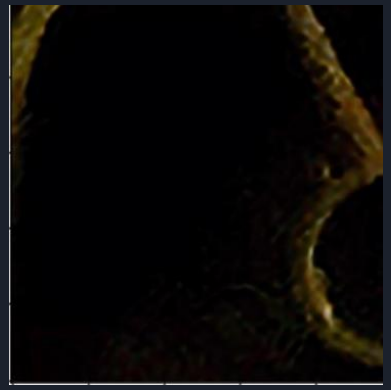
Images obtained after successive iterative kernel correction

Test on Real images

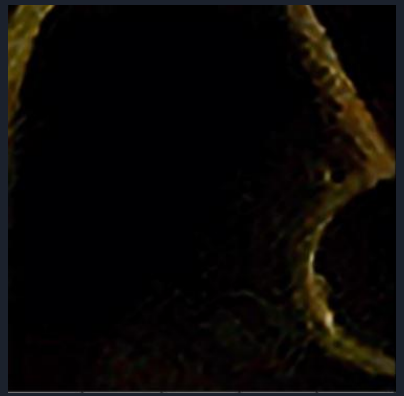


← Original Low Resolution image 64x64

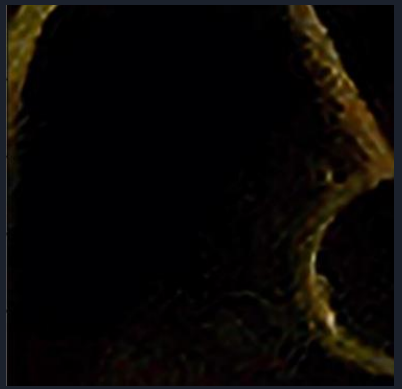
Output images → 256x256



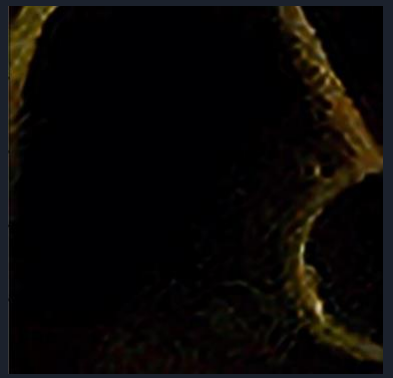
Iteration 1



Iteration 2

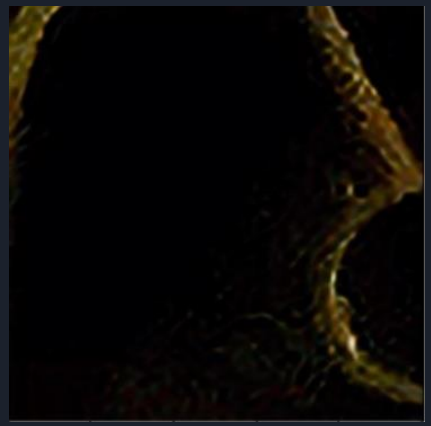


Iteration 3

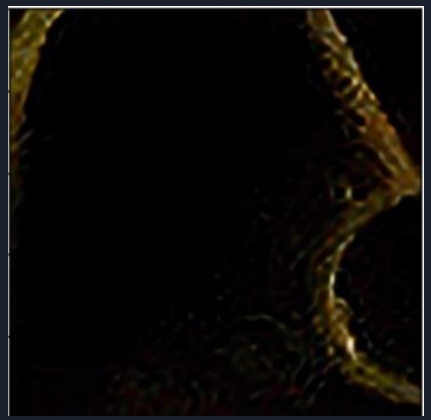


Iteration 4

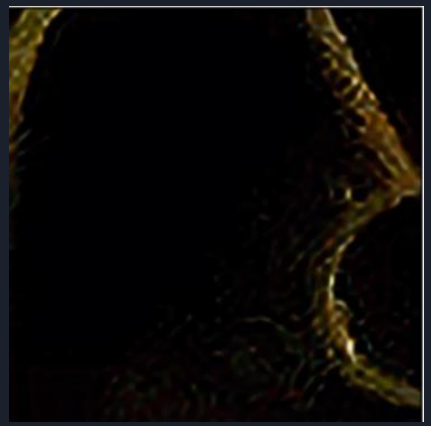
Original Low Resolution image



Iteration 5



Iteration 6



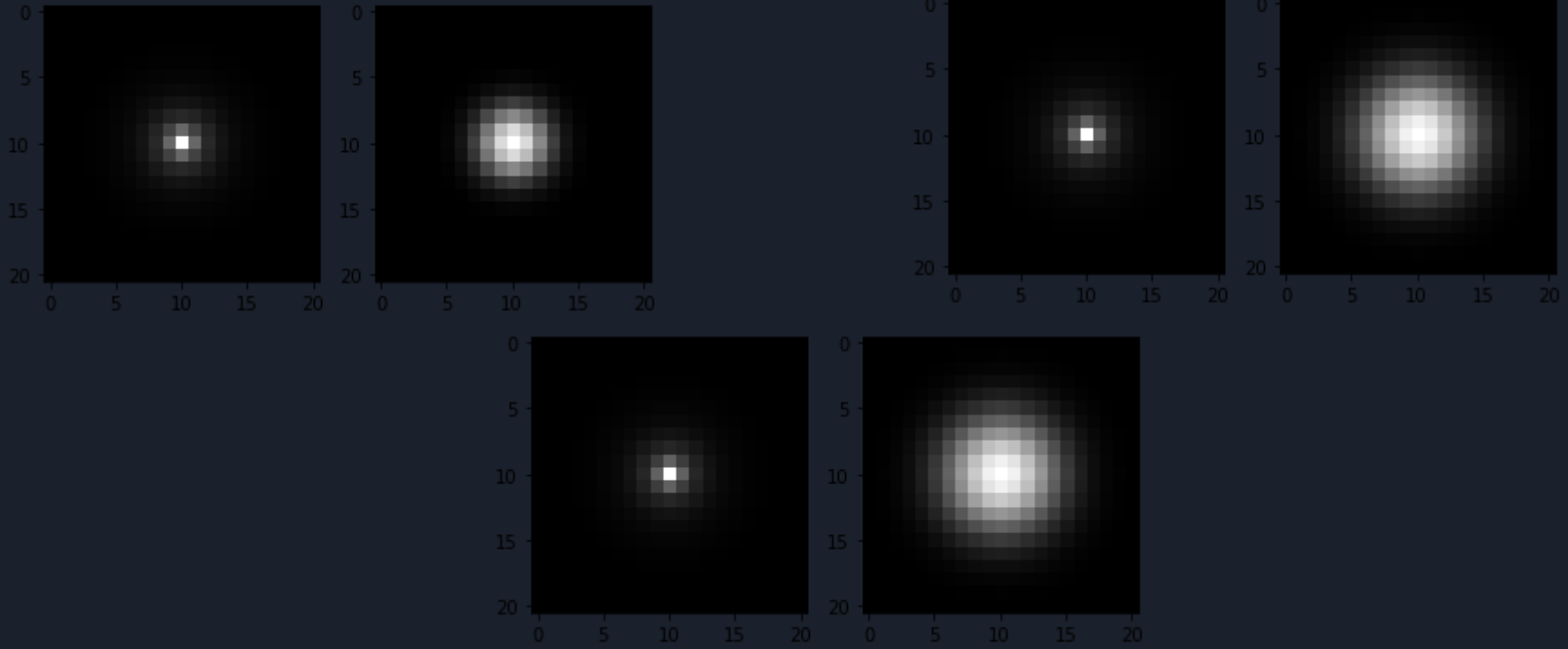
Iteration 7

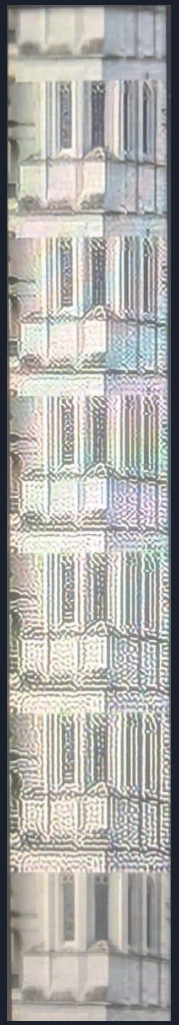
Observations

1. Based on similar lines, the predictor kernel could not be used alone as the kernels predicted using only predictor were significantly different from the ground truth kernels

1. During the initial epochs, when the predictor kernel had been corrected iteratively just once or twice, the features in the high resolution images generated through SFTMD were over emphasized. This happens because the kernels predicted have higher sigma than that of the ground truth sigma. Since this happened in all the validation images used, the need of a corrector was realized more fully

Right (Predicted Kernels) | Left (Ground truth Kernels)





Over-emphasis of image caused to the predictor kernel having a higher sigma initially than the ground truth kernel for 0th epoch

Images obtained through IKC after 60th epoch

Thank You!