Course project

SEMESTER – 3

Submitted By:

Sasikanth Koti | MT19AIE308

Nikhila Dhulipalla | MT19AIE270

Adhun Thalekkara | MT19AIE205

Link to drive containing all the details(proposal, code and results):

https://drive.google.com/drive/folders/1cjmYezmBrjO5YwhTtCbBos66jHj9vBNB?usp=sharing

Datasets used in the paper:

RESIDE consisting of training and testing datasets.

Here the testing dataset consists of SOTS Indoor and SOTS outdoor.

Introduction

Image dehazing is considered to be a low-level vision task which has got lots of attention in computer vision domain past few decades. Haze, fog, fumes, mist or smoke will greatly reduce the quality of scenery images, and the irradiance received by the camera from the scene point will be attenuated along the line of sight. This leads to image degradation which in turn will lose the contract and color fidelity as the amount of scattering is related to the scene points distance from the camera as the degradation is spatially variant. This also makes the visual tasks such as classification, tracking, object detection and so on much difficult to solve. So, removal of haze from the images was highly significant to increase the visibility of the scene and also the rectify the color shift caused by the air light.

With the advance in technology, people are tending to use portable digital devices such as smartphones extensively for capturing images and image reflection is one such issue commonly faced. Especially for the images captured through glass or mirror there is definitely an unpleasant reflection and it not only damages the image quality but also plays a vital role in the performance of the computer vision tasks like image classification. So, it's highly desirable to be removed by using user-friendly image reflection suppression technique which can be used on smart phones within short span of time and obtain a better result in real-time as per the user's visual perception.

Implementation details

The main aim of our project is to apply CV techniques and develop pipeline for image enhancement via Dehazing and removal of image reflections.

These has been successfully implemented in the below mentioned tasks details:

1. Implementing the pipeline for image enhancement via Dehazing.
2. Implementing image reflection removal optimization techniques to perform image dehazing.
3. Implementing and reproducing existing results of the paper AAAI 2020 paper - FFA-Net: Feature Fusion Attention Network for Single Image Dehazing.
4. Implementing own architecture to obtain possible improvements of the current/near to SOTA.
5. Apply improved pipeline to video data for dehazing.

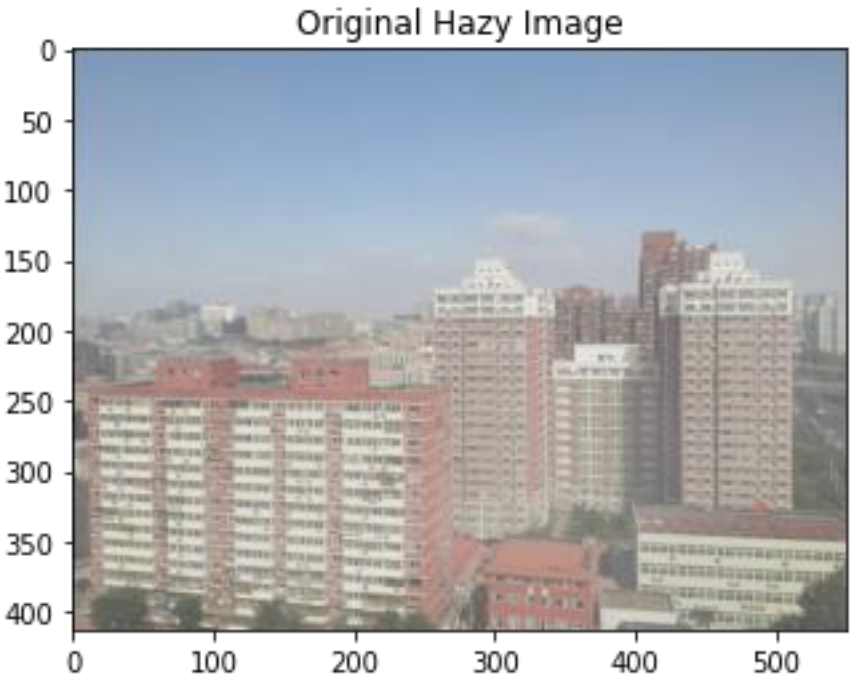
TASK 1: Implementing the pipeline for image enhancement via dehazing.

Dark channel prior (DCP) for single image haze removal was implemented along with this the preprocessing technique is White Balance (WB) and postprocessing technique used are CLAHE and DWT.

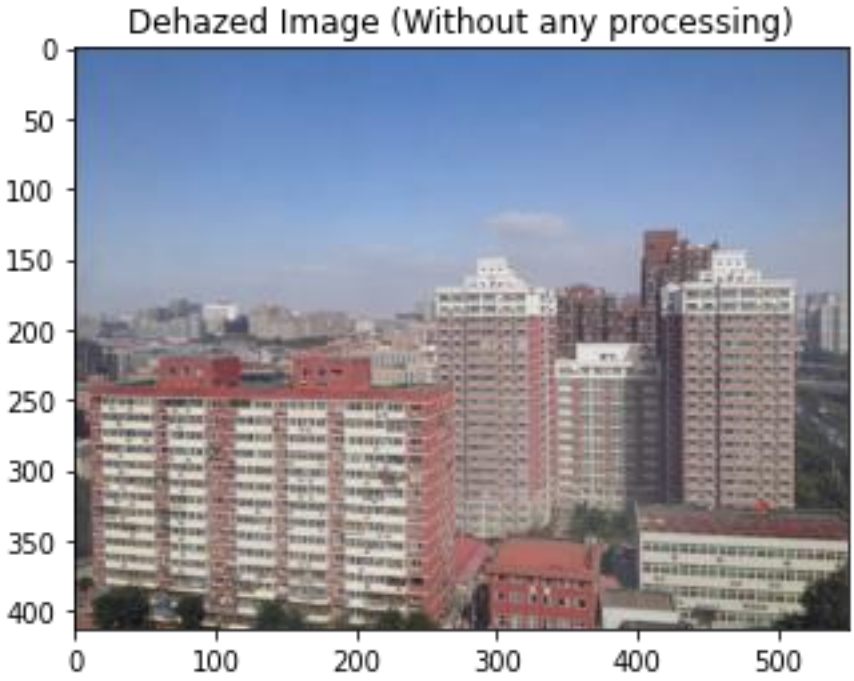
The dark channel prior is applicable on the outdoor haze free images where there is a nonsky region. Here some of pixels might have low intensity in at least one of the RGB color channel and the intensity of hazy images are very low and close to zero in such channels is mainly because of the airlight. So, we can say that – if J is an outdoor haze-free image (except sky region) then the intensity would be $J^{dark} \rightarrow 0$. The factors of low intensity in dark channel a) shadows of the objects/things b) colorful objects or surfaces emitting low reflectance in any colosr channel c) dark objects or surfaces. The hazy images will be brighter than its haze-free images, in other words the hazy images are tend to have higher intensity for the regions with the denser haze. But the images with the sky regions can be handled by using haze imaging model and the dark channel prior together.

Results obtained on original image size:

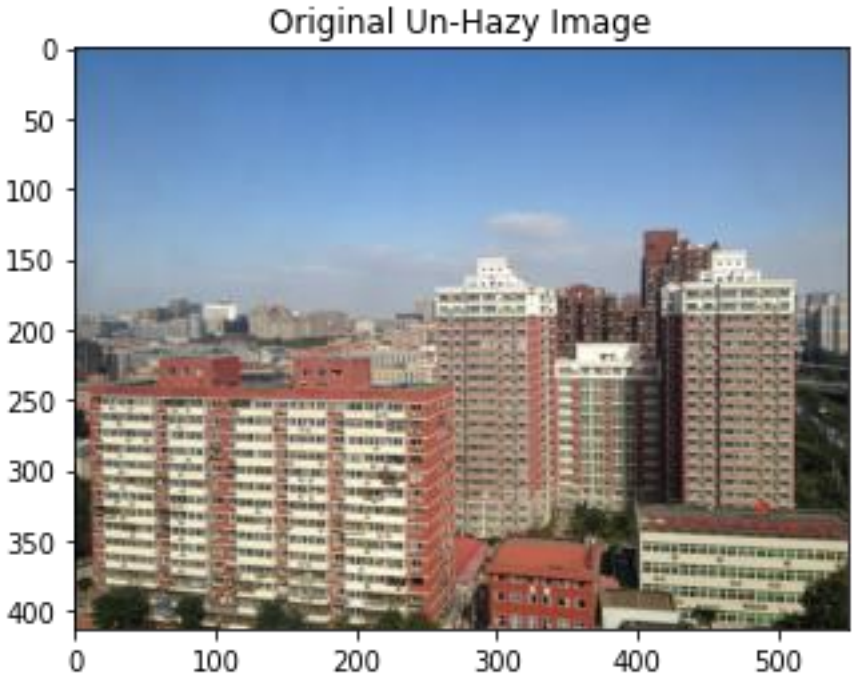
Original Hazy image:



Dehazed Image:



Original Un-hazy Image:



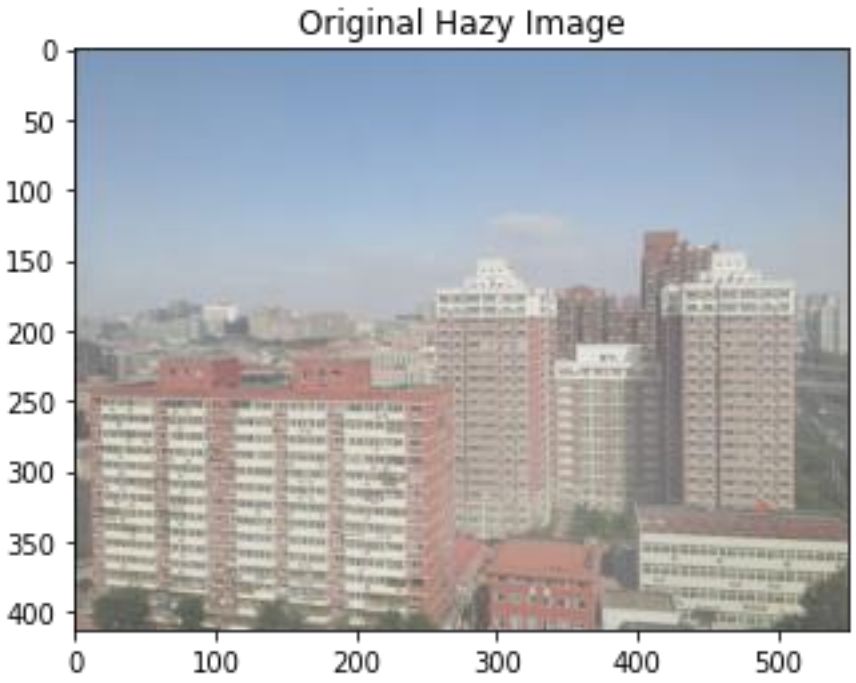
WB is used as the preprocessing technique to remove the unrealistic color. It helps in balancing the color temperature in the image by adding the opposite color to the image so that the color temperature is neutral.

CLAHE – Contrast Limited Adaptive Histogram Equalization is a type of the histogram equalization used as postprocessing technique. It helps in maintaining or limiting the contrast amplification which in turn reduces the noise amplification issue, in simpler words it is used for enhancing the local contrast of an image. Here the vicinity of a given pixel value is given by the slope of the transformation function.

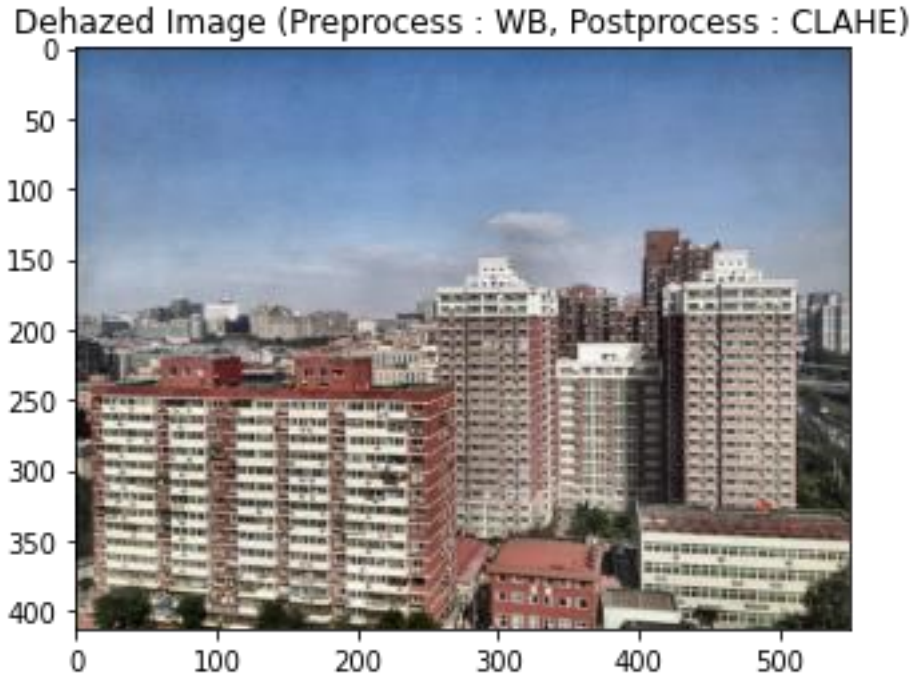
DWT – Discrete wavelet Transform is a technique used for decomposing the signal into multiple subbands in such a way that low frequency subbands will be having a finer frequency resolution. This is used as postprocessing technique.

Results obtained:

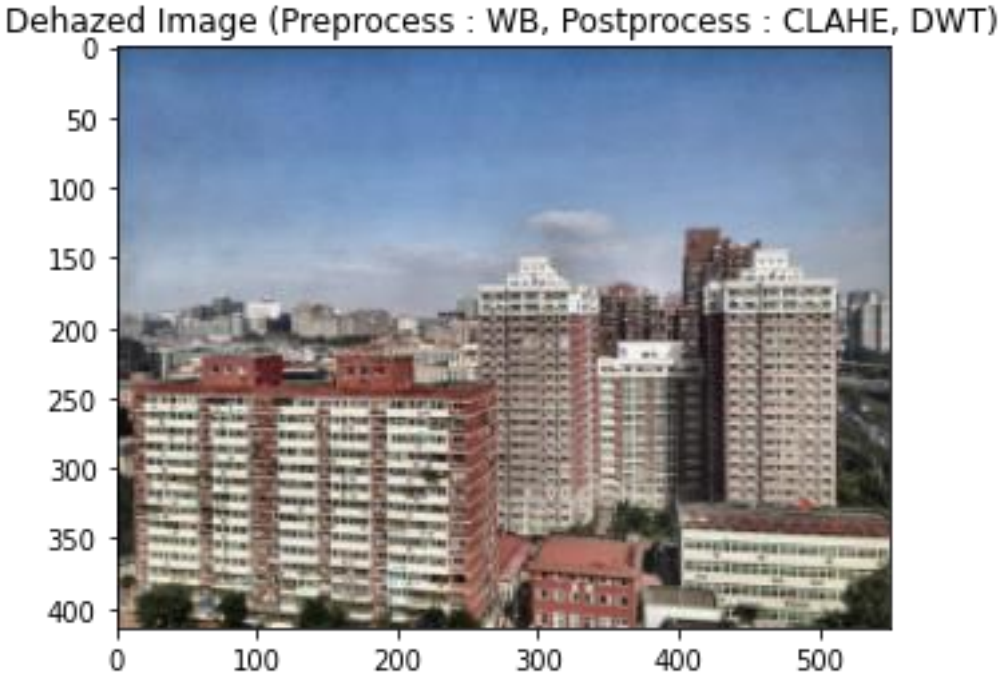
Original Hazy Image:



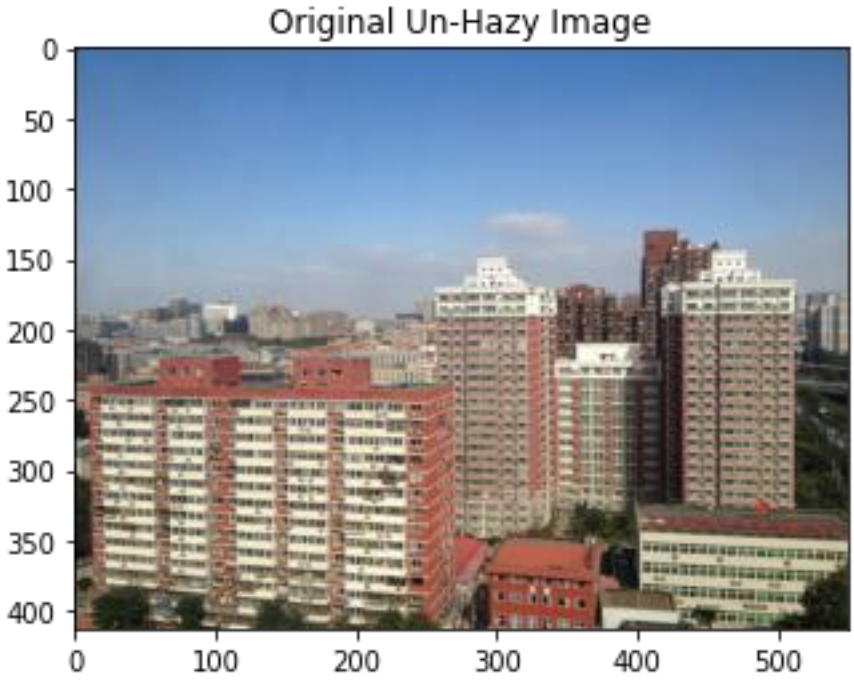
Dehazed Image – preprocessing is WB and postprocessing is CLAHE:



Dehazed Image – preprocessing is WB and postprocessing is CLAHE and DWT:



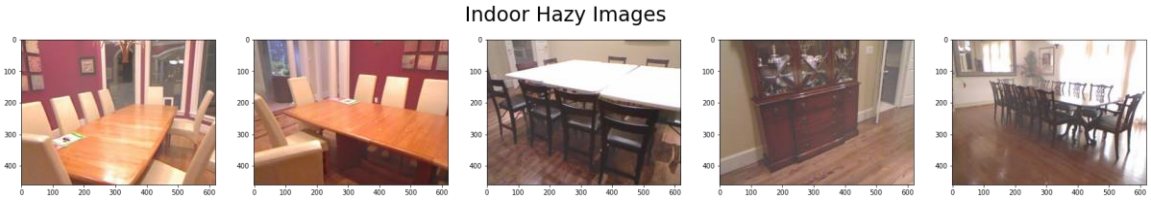
Original Un-hazy Image:



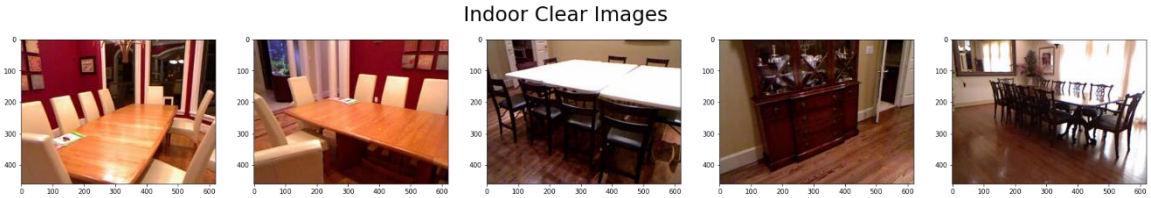
<u>Implementing the above dehazing technique on the images in the dataset:</u>

We have considered the indoor and outdoor images for the implementation.

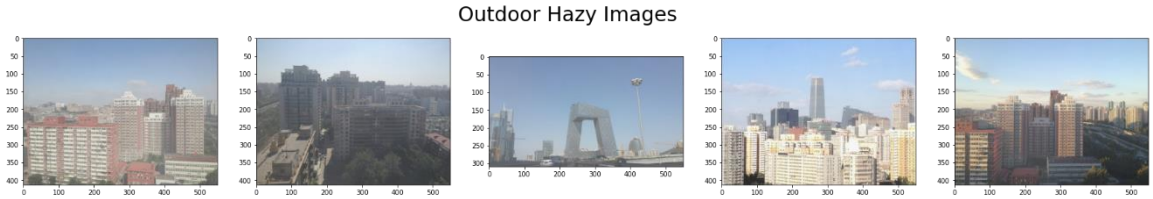
Indoor Hazy images:



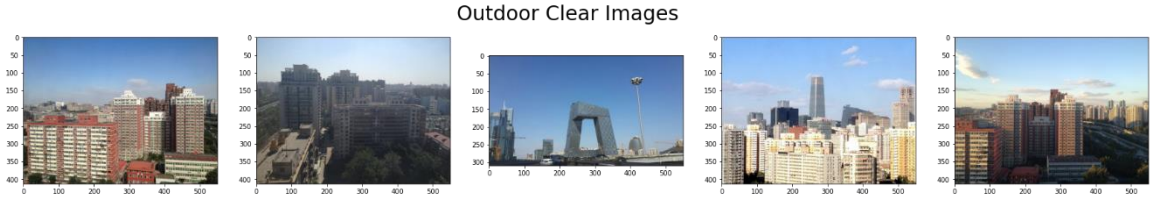
Indoor clear images:



Outdoor haze images:

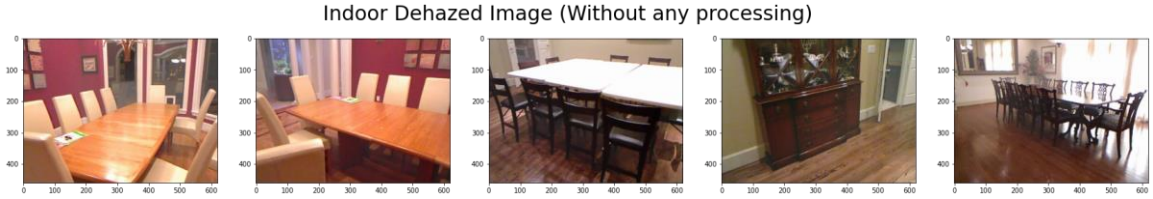


Outdoor clear images:

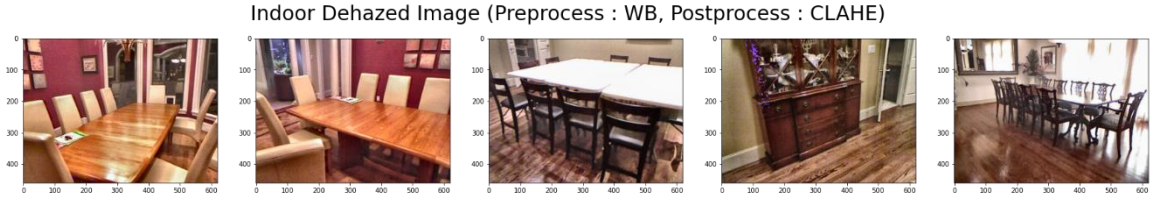


Dehazed images:

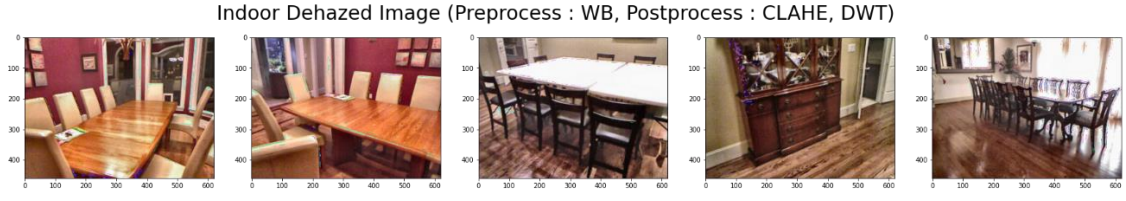
Output of DCP on the indoor images:



Output of DCP on the indoor images with preprocessing as WB and postprocessing as CLAHE:



Output of DCP on the indoor images with preprocessing as WB and postprocessing as CLAHE and DWT:



Output of DCP on the outdoor images:

Output of DCP on the outdoor images with preprocessing as WB and postprocessing as CLAHE:

Output of DCP on the outdoor images with preprocessing as WB and postprocessing as CLAHE and DWT:

Evaluation Metrics – PSNR and SSIM

For indoor images:

Average PSNR for SOTS Indoor Images Dehazed with DCP	**14.912122266196l2**
Average PSNR for SOTS Indoor Images Dehazed with DCP with Preprocessing(Pipeline1)	**15.088614455629532**
Average PSNR for SOTS Indoor Images Dehazed with DCP with both Preprocessing and Postprocessing(Pipeline2)	**14.898261597476749**
Average SSIM for SOTS Indoor Images Dehazed with DCP	**0.7808506442597811**
Average SSIM for SOTS Indoor Images Dehazed with DCP with Preprocessing(Pipeline1)	**0.7487565257993833**
Average SSIM for SOTS Indoor Images Dehazed with DCP with both Preprocessing and Postprocessing(Pipeline2)	**0.7091964054242976**

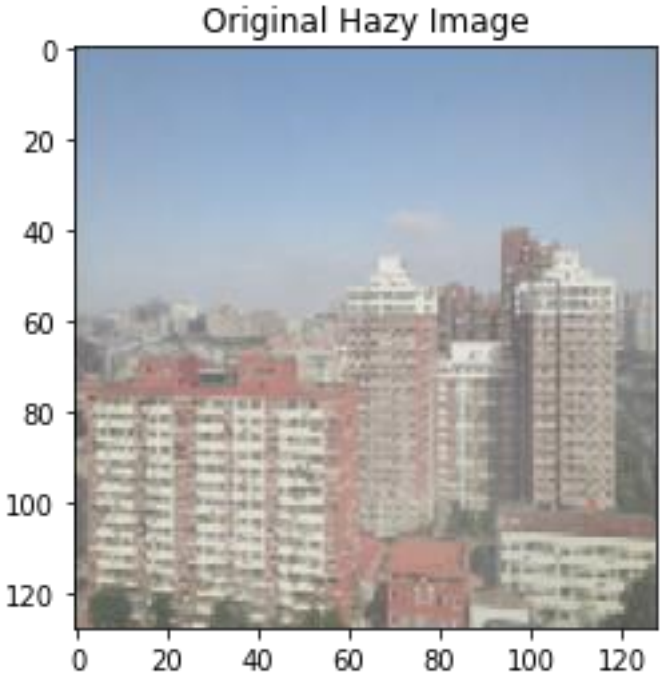
For outdoor images:

Average PSNR for SOTS Outdoor Images Dehazed with DCP	**21.9988773393858**
Average PSNR for SOTS Outdoor Images Dehazed with DCP with Preprocessing(Pipeline1)	**18.558668056069813**
Average PSNR for SOTS Outdoor Images Dehazed with DCP with both Preprocessing and Postprocessing(Pipeline2)	**18.076255938198056**
Average SSIM for SOTS Outdoor Images Dehazed with DCP	**0.9086461333139355**
Average SSIM for SOTS Outdoor Images Dehazed with DCP with Preprocessing(Pipeline1)	**0.8236948048114106**
Average SSIM for SOTS Outdoor Images Dehazed with DCP with both Preprocessing and Postprocessing(Pipeline2)	**0.7580779204529856**

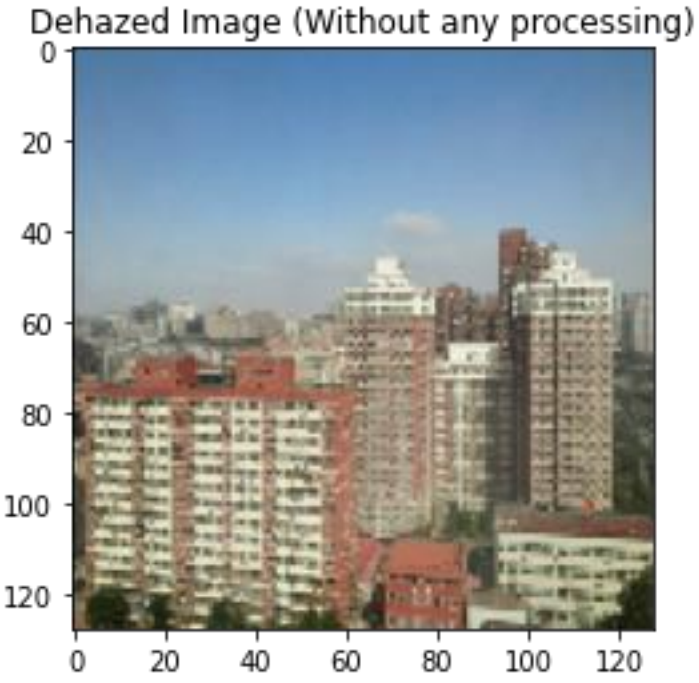
***Here the pipeline 1 means DCP with preprocessing as WB and postprocessing as CLAHE and pipeline 2 means DCP with preprocessing as WB and postprocessing as CLAHE and DWT.*

Results obtained on resized image of 128 x 128:

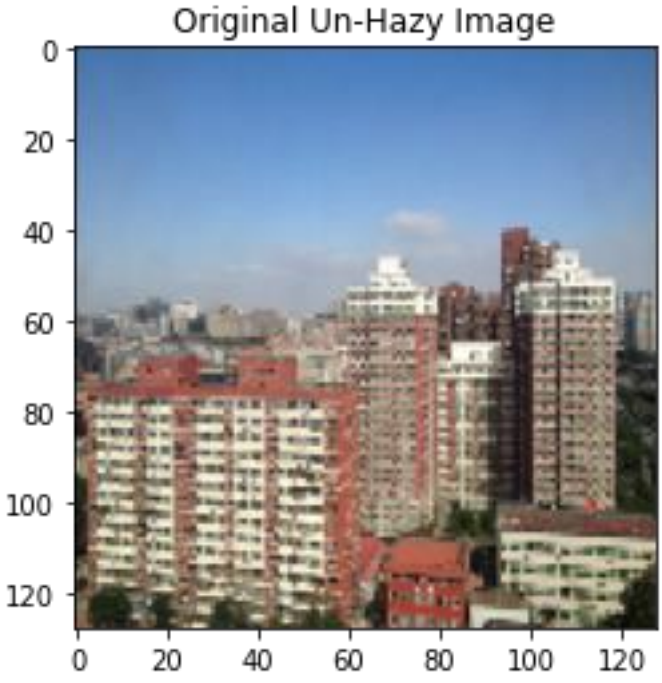
Original Hazy image:



Dehazed Image:

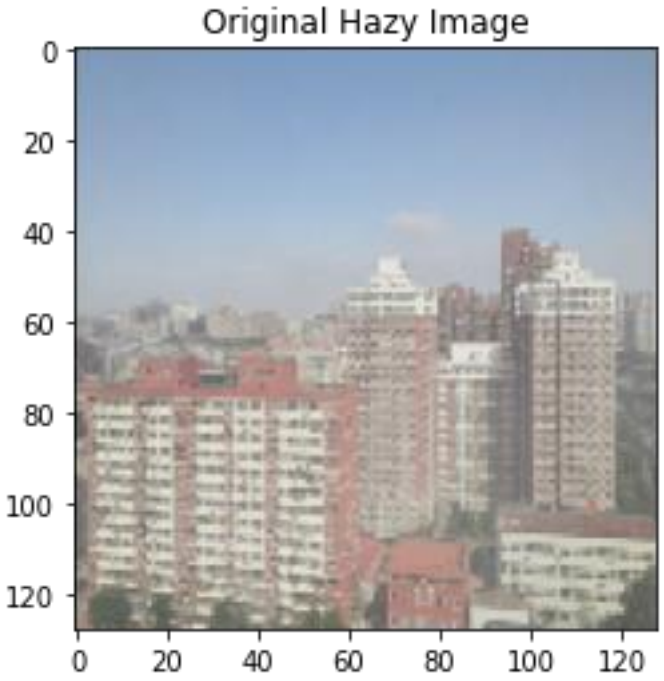


Original Un-hazy Image:

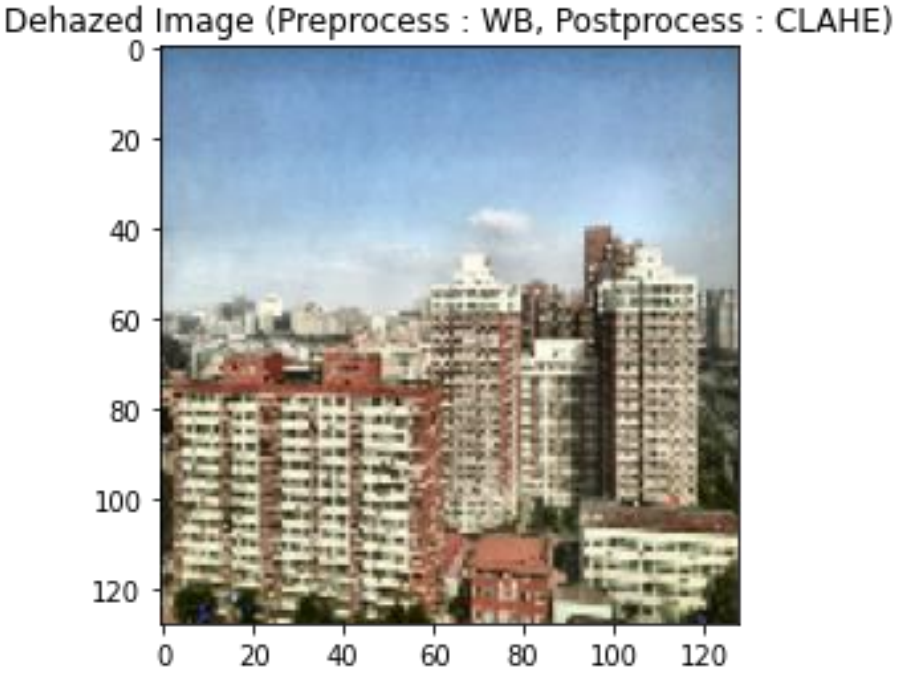


Results obtained by applying preprocessing and postprocessing techniques on resized image of 128 x 128:

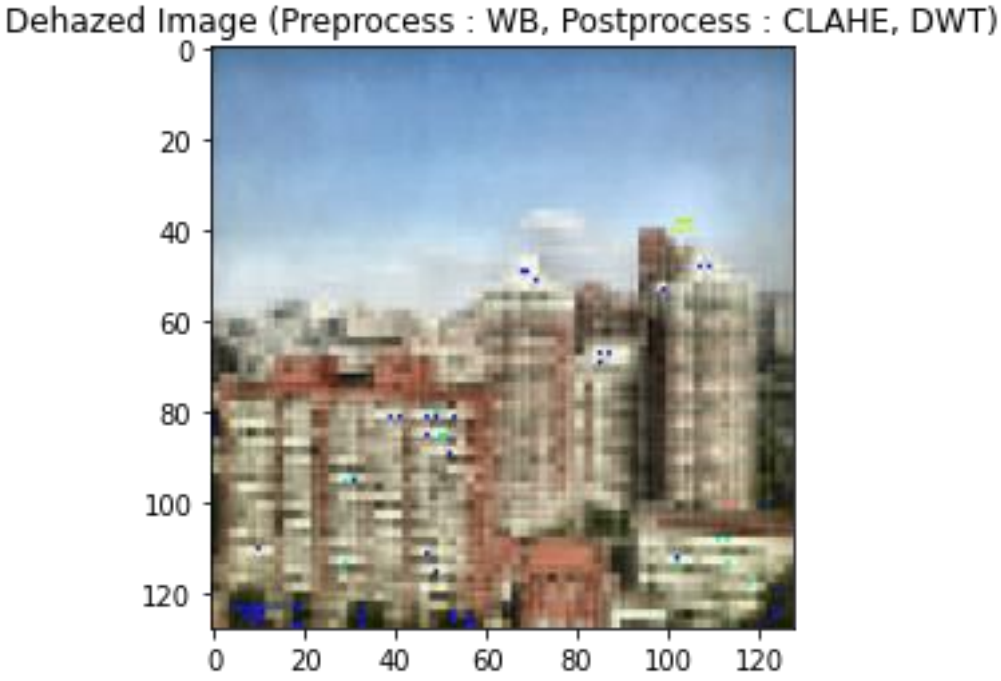
Original Hazy Image:



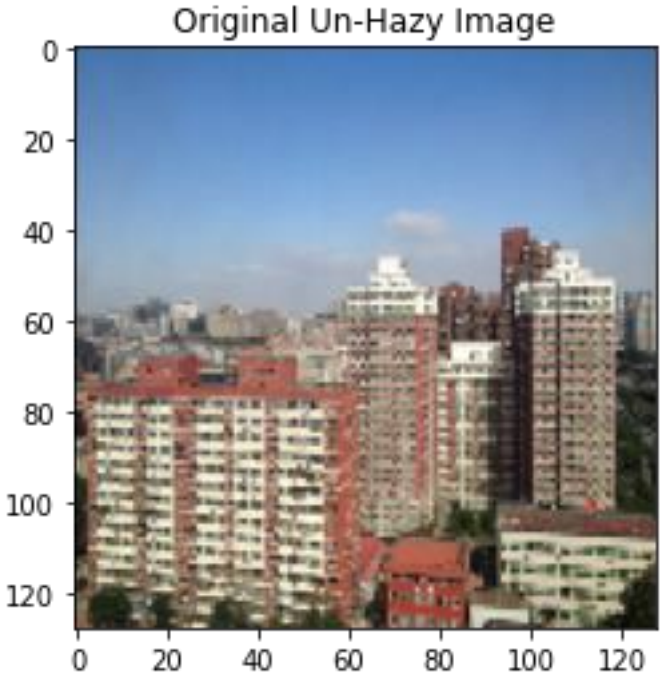
Dehazed Image – preprocessing is WB and postprocessing is CLAHE:



Dehazed Image – preprocessing is WB and postprocessing is CLAHE and DWT:



Original Un-hazy Image:



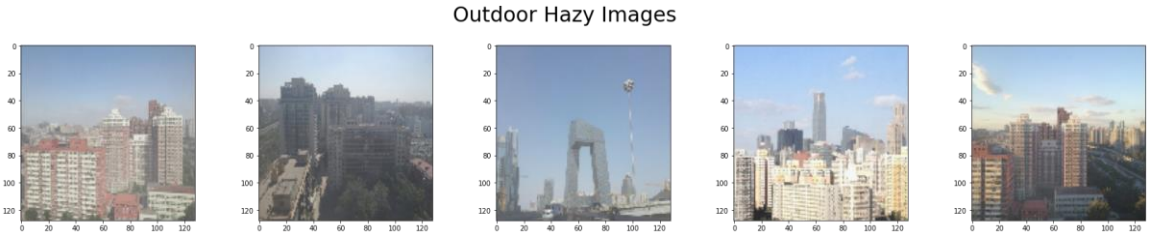
Implementing the above dehazing technique on the resized images of 128 x 128 in the dataset:

We have considered the indoor and outdoor images for the implementation.

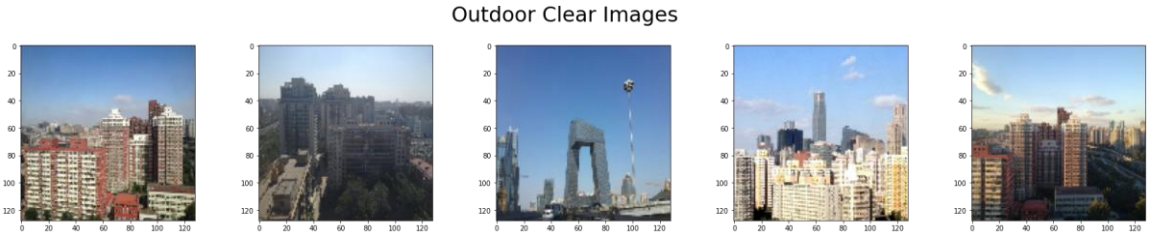
Indoor Hazy images:

Indoor clear images:

Outdoor haze images:

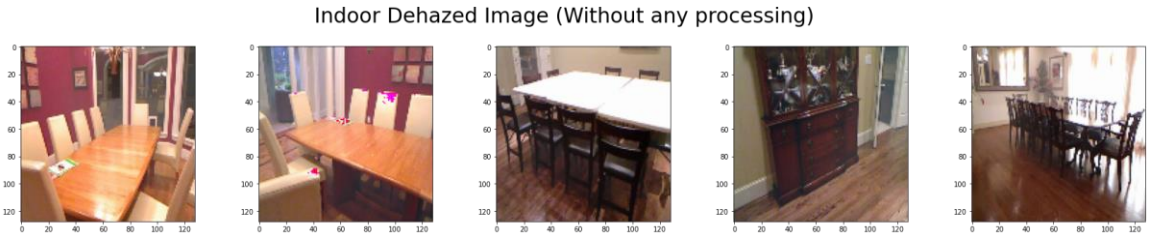


Outdoor clear images:

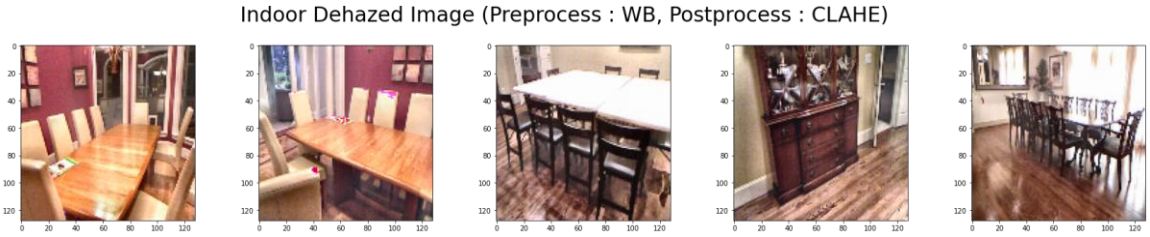


Dehazed images of size 128 x 128:

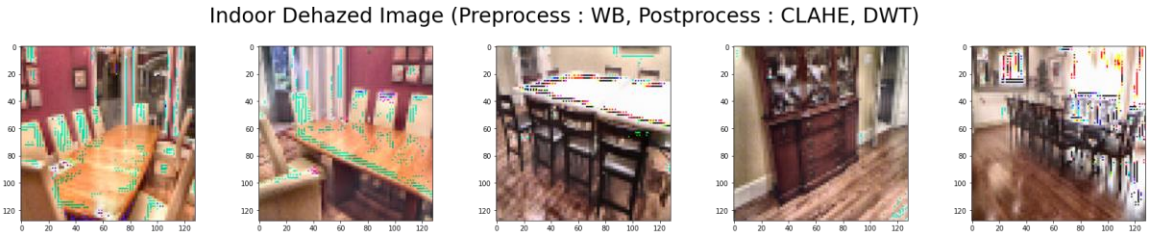
Output of DCP on the indoor images:



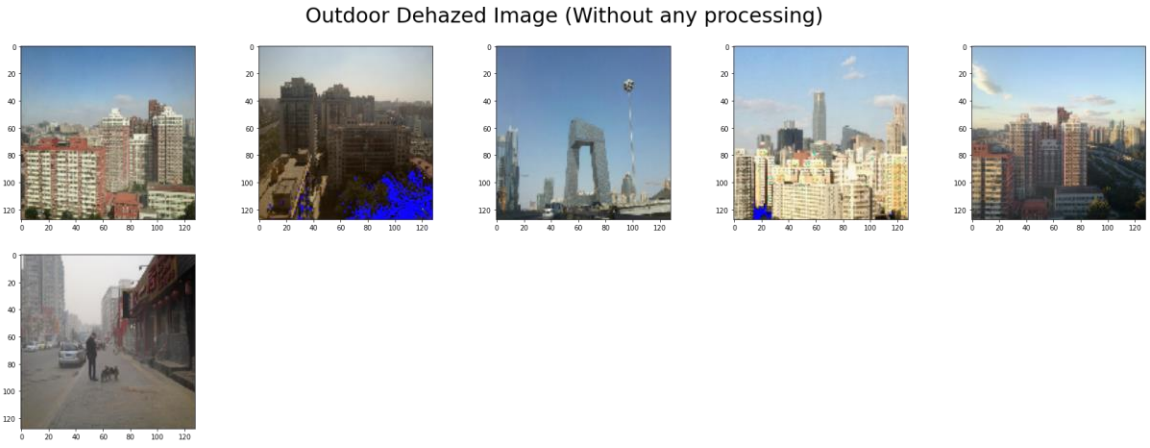
Output of DCP on the indoor images with preprocessing as WB and postprocessing as CLAHE:



Output of DCP on the indoor images with preprocessing as WB and postprocessing as CLAHE and DWT:



Output of DCP on the outdoor images:



Output of DCP on the outdoor images with preprocessing as WB and postprocessing as CLAHE:

Output of DCP on the outdoor images with preprocessing as WB and postprocessing as CLAHE and DWT:

Evaluation Metrics – PSNR and SSIM

For indoor images:

Average PSNR for SOTS Indoor Images Dehazed with DCP	**14.770367734287765**
Average PSNR for SOTS Indoor Images Dehazed with DCP with Preprocessing(Pipeline1)	**11.830743977178484**

Average PSNR for SOTS Indoor Images Dehazed with DCP with both Preprocessing and Postprocessing(Pipeline2)	**11.153118203661 97**
Average SSIM for SOTS Indoor Images Dehazed with DCP	**0.7756665394078519**
Average SSIM for SOTS Indoor Images Dehazed with DCP with Preprocessing(Pipeline1)	**0.677964536153642**
Average SSIM for SOTS Indoor Images Dehazed with DCP with both Preprocessing and Postprocessing(Pipeline2)	**0.511616417911044**

For outdoor images:

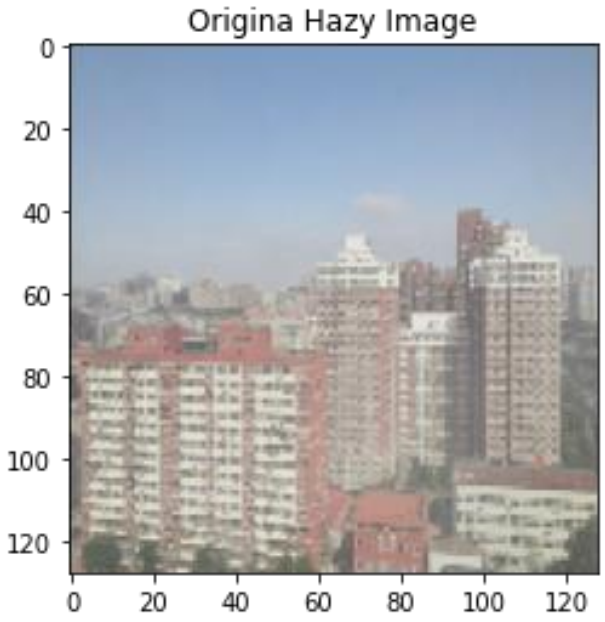
Average PSNR for SOTS Outdoor Images Dehazed with DCP	**22.64856789796369**
Average PSNR for SOTS Outdoor Images Dehazed with DCP with Preprocessing(Pipeline1)	**15.70474021815297**
Average PSNR for SOTS Outdoor Images Dehazed with DCP with both Preprocessing and Postprocessing(Pipeline2)	**14.925031079559234**
Average SSIM for SOTS Outdoor Images Dehazed with DCP	**0.9226589232228252**
Average SSIM for SOTS Outdoor Images Dehazed with DCP with Preprocessing(Pipeline1)	**0.771224421173174**
Average SSIM for SOTS Outdoor Images Dehazed with DCP with both Preprocessing and Postprocessing(Pipeline2)	**0.6395076251828142**

***Here the pipeline 1 means DCP with preprocessing as WB and postprocessing as CLAHE and pipeline 2 means DCP with preprocessing as WB and postprocessing as CLAHE and DWT.*

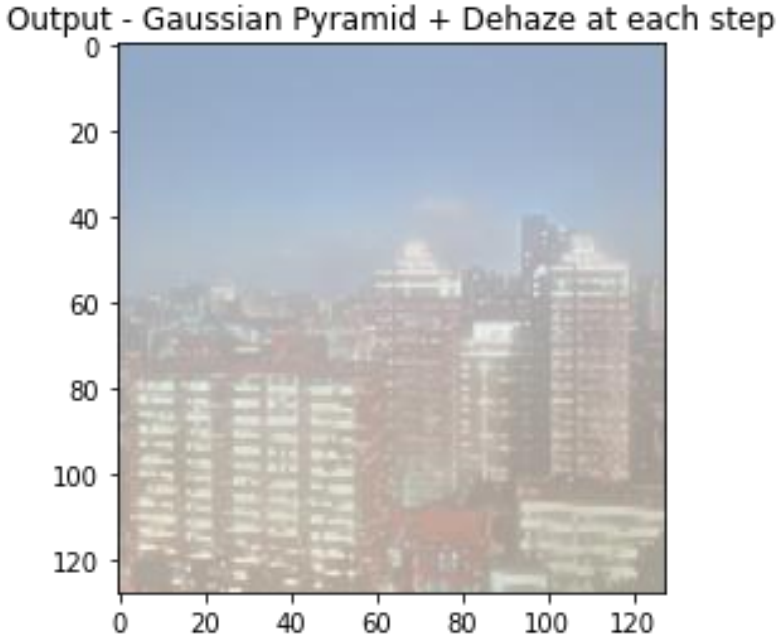
We have tried implementing the Gaussian pyramid but no luck.

<u>Results obtained on the resized images of 128 x 128:</u>

Original Hazy Image:



Output of the Gaussian pyramid and dehaze:



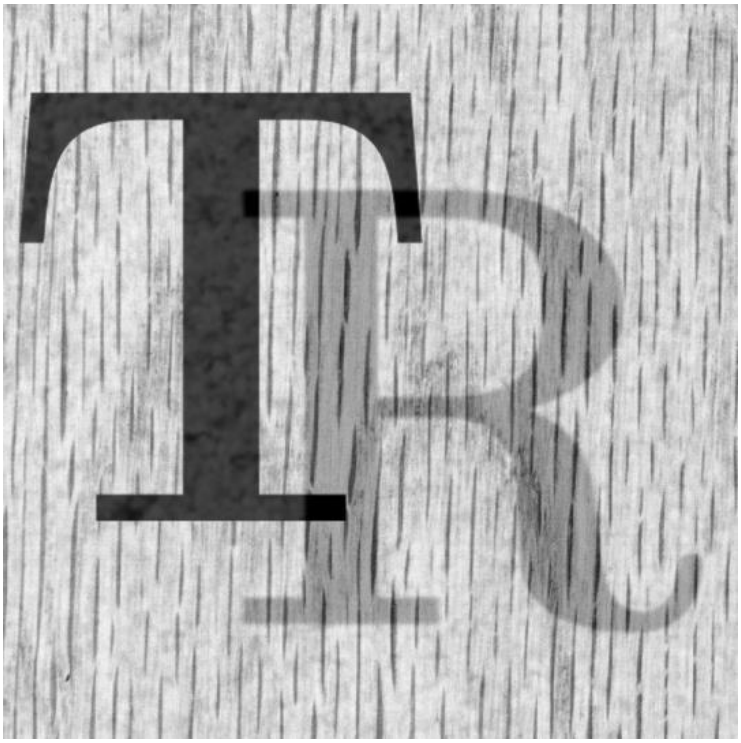
TASK 2: Implementing image reflection removal optimization techniques to perform image dehazing

Image reflection usually spoils the outlook of the image captured especially the images captured through glass or mirror selfies. So, image reflection suppression technique has a great practical significance. There were many approaches proposed to remove the reflection but the performance w.r.t the quality of the de-reflection wasn't satisfactory. Also, they were able to handle on the small sized images and had the problem of computational inefficiency. We have implemented the image reflection from one of the efficient approaches proposed in the recent papers, where, the model is convex and the optimal solution is obtained by solving the partial differential equation using Discrete Cosine Transform (DCT).

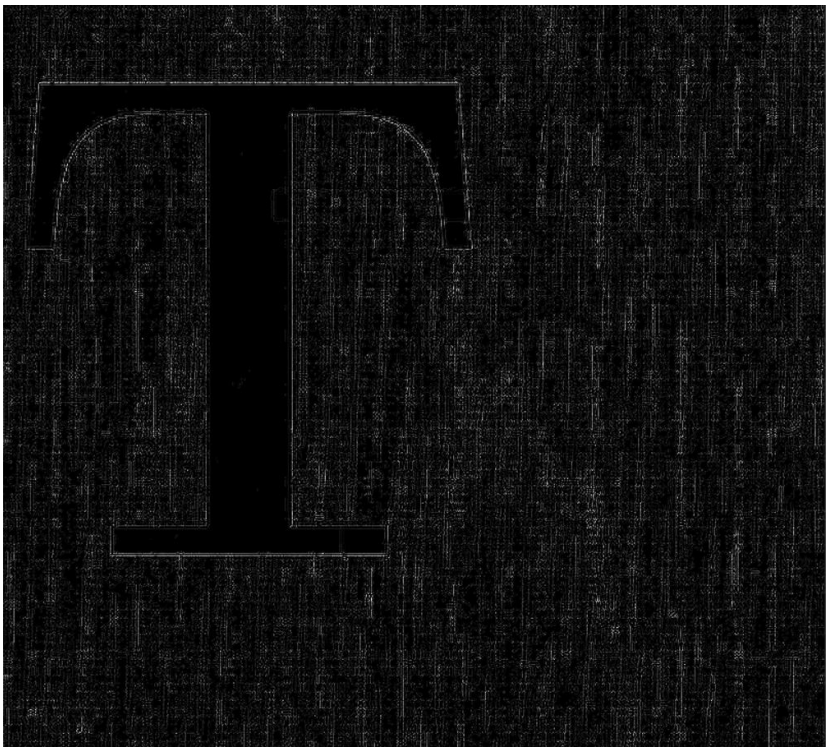
The base code of paper is on MATLAB and we have tried implementing the same in python but the obtained results are not satisfactory. Hence, we did not apply the reflection removal techniques for image dehazing.

Results Obtained:

Input image:



Output image:

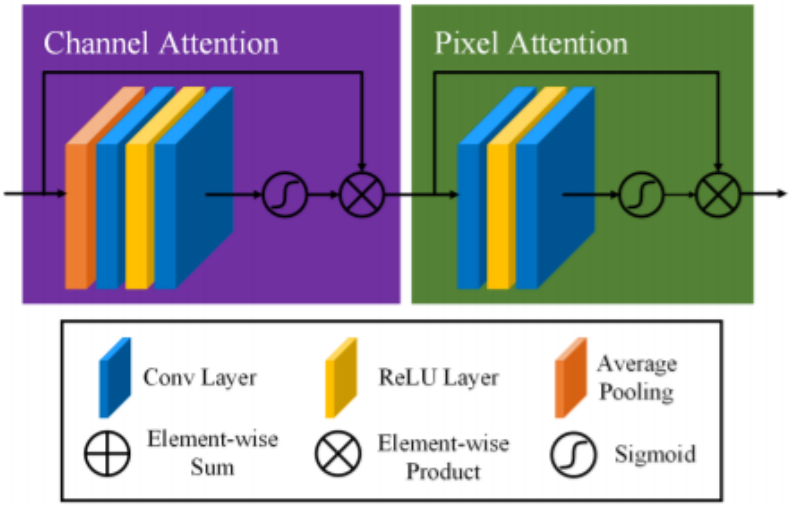


TASK 3: Implementing and reproducing existing results of the paper AAAI 2020 paper - FFA-Net: Feature Fusion Attention Network for Single Image Dehazing.

We have implemented the FFA-Net for single image dehazing in pytorch. The key contributions of the author are as below:

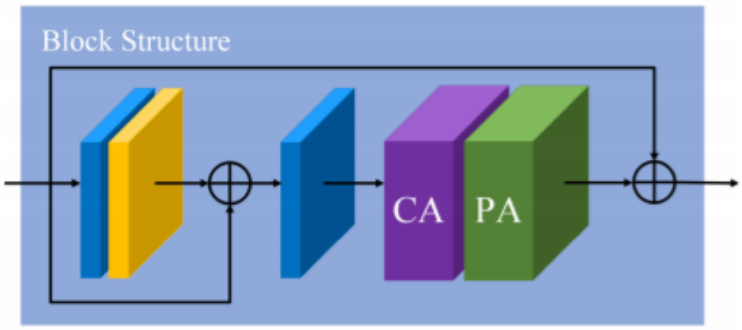
a) Authors have proposed a novel end-to-end feature fusion attention network FFA-Net for single image dehazing. The performance of this technique on the regions with thick haze and rich texture details is outstanding and also one of the main advantages was in the restoration of image details and color fidelity.
b) Feature attention model was proposed which can combine the channel attention and pixel attention mechanism providing additional flexibility with different types of information and by focusing on the thick haze pixels and important channel information.

FA module:



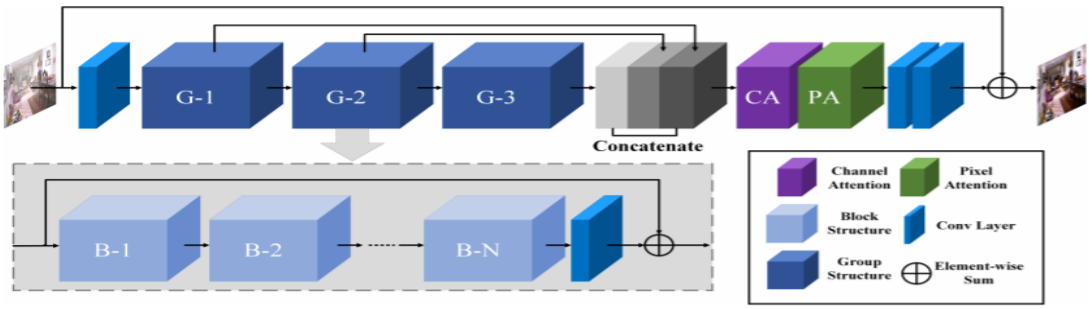
c) A basic block with local residual learning and FA was proposed in which the local residual learning can help in allowing the information having thin haze region and the low frequency information obtained will be bypassed by multiple skip connections. The FA will be helping to improve the capacity of the FFA-Net overall.

Basic Block Structure:



d) FFA architecture was proposed which will help in retaining the shallow layer information by passing it to the deep layers. It can fuse all the features and also adaptively learn the different weights at different level feature information, with this it helps in obtaining a better performance.

The FFA-Net architecture is as below:



- We have obtained the inference using pretrained model on resized test images of 128 x 128.
- We have trained the model from scratch with reduced resolution of 46 x 62.

Below are the results of the both:

<u>Results obtained on pretrained model for the resized images of 128 x 128:</u>

Indoor hazy images:

Indoor Hazy Images

Indoor clear images:

Indoor Clear Images

Outdoor Hazy images:

Outdoor Hazy Images

Outdoor clear images:

Outdoor Hazy Images

<u>Output of the dehazed FFA Net:</u>

Indoor dehazed images:

Indoor Dehazed Images

Outdoor dehazed images:

Outdoor Dehazed Images

Evaluation Metrics – PSNR and SSIM

	Test SSIM	Test PSNR
Indoor	**0.5559013981819153**	**14.856913864135743**
Outdoor	**0.6362881175870818**	**19.4084547864712 36**

Results obtained on model trained from scratch for the resized image of 46 x 62:

The images are resized to 46*62 for the model trained from scratch due to the compute limitations.

Epochs run : 15 – showing the last epoch output

epoch 15/15 | step 10/219 | running loss 0.114168428629636 76 | ssim 0.5856455862522125 | psnr 16.558590316772 46

epoch 15/15 | step 20/219 | running loss 0.117172446101903 92 | ssim 0.5889574348926544 | psnr 16.430514144897 46

epoch 15/15 | step 30/219 | running loss 0.117751784622669 22 | ssim 0.5789317846298218 | psnr 16.346840000152 59

epoch 15/15 | step 40/219 | running loss 0.116945819556713 1 | ssim 0.5836574375629425 | psnr 16.354582691192626

epoch 15/15 | step 50/219 | running loss 0.116734992712736 13 | ssim 0.5812386453151703 | psnr 16.413066768646 24

epoch 15/15 | step 60/219 | running loss 0.112908127158880 24 | ssim 0.5827911019325256 | psnr 16.678758430480958

epoch 15/15 | step 70/219 | running loss 0.118421257287263 87 | ssim 0.5801976203918457 | psnr 16.297183704376 22

epoch 15/15 | step 80/219 | running loss 0.115909116715192 8 | ssim 0.5810677289962769 | psnr 16.531455612182 62

epoch 15/15 | step 90/219 | running loss 0.114145397394895 56 | ssim 0.5789012074470 52 | psnr 16.611256790161132

epoch 15/15 | step 100/219 | running loss 0.114717289805412 29 | ssim 0.5796096503734589 | psnr 16.511036396026 61

epoch 15/15 | step 110/219 | running loss 0.11433384269475937 | ssim 0.5840663015842438 | psnr 16.57526397705078

epoch 15/15 | step 120/219 | running loss 0.12006083130836487 | ssim 0.5801469147205353 | psnr 16.1943264007568 36

epoch 15/15 | step 130/219 | running loss 0.11878406926989556 | ssim 0.5825272917747497 | psnr 16.356505870819092

epoch 15/15 | step 140/219 | running loss 0.1143928386271 | ssim 0.5896229565143585 | psnr 16.5391204833984 38

epoch 15/15 | step 150/219 | running loss 0.11767048984766007 | ssim 0.5794375777244568 | psnr 16.36728391647339

epoch 15/15 | step 160/219 | running loss 0.1176500953733921 | ssim 0.5790887951850892 | psnr 16.33465929031372

epoch 15/15 | step 170/219 | running loss 0.1144893430173397 | ssim 0.5869877934455872 | psnr 16.57353868484497

epoch 15/15 | step 180/219 | running loss 0.11747756078839303 | ssim 0.5708408236503602 | psnr 16.39449644088745

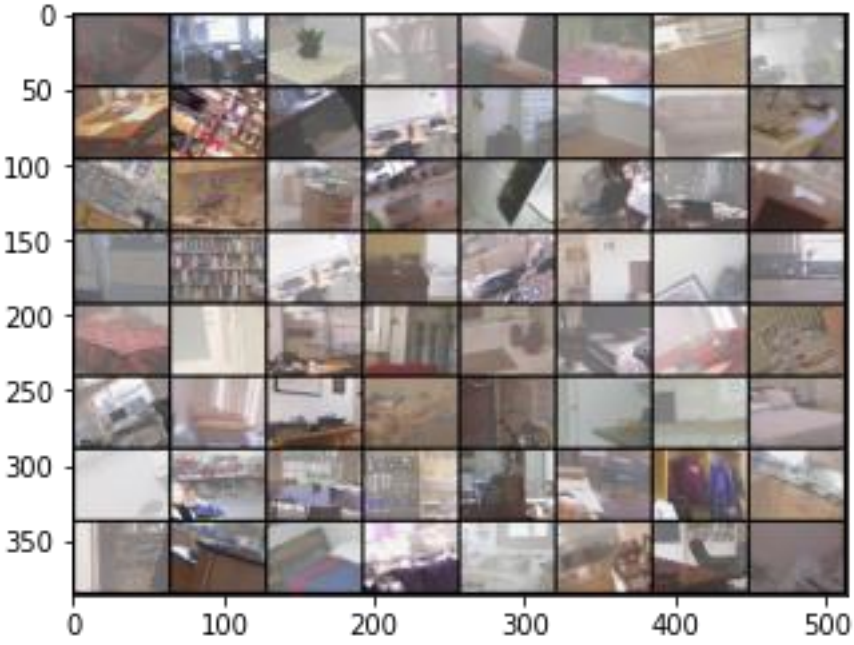
epoch 15/15 | step 190/219 | running loss 0.11486219689249992 | ssim 0.5787747144699097 | psnr 16.547403621673585

epoch 15/15 | step 200/219 | running loss 0.1187741443514824 | ssim 0.5857049643993377 | psnr 16.25970516204834

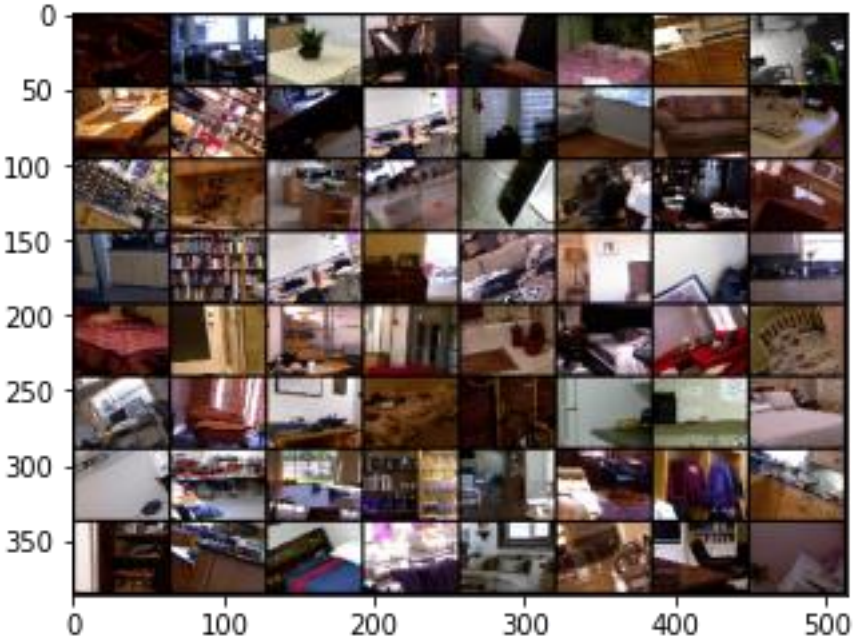
epoch 15/15 | step 210/219 | running loss 0.11838586628437042 | ssim 0.5773221254348755 | psnr 16.31971321105957

epoch 15/15 | step 219/219 | running loss 0.11875671231084 4 | ssim 0.5801493724187216 | psnr 16.314223289489746

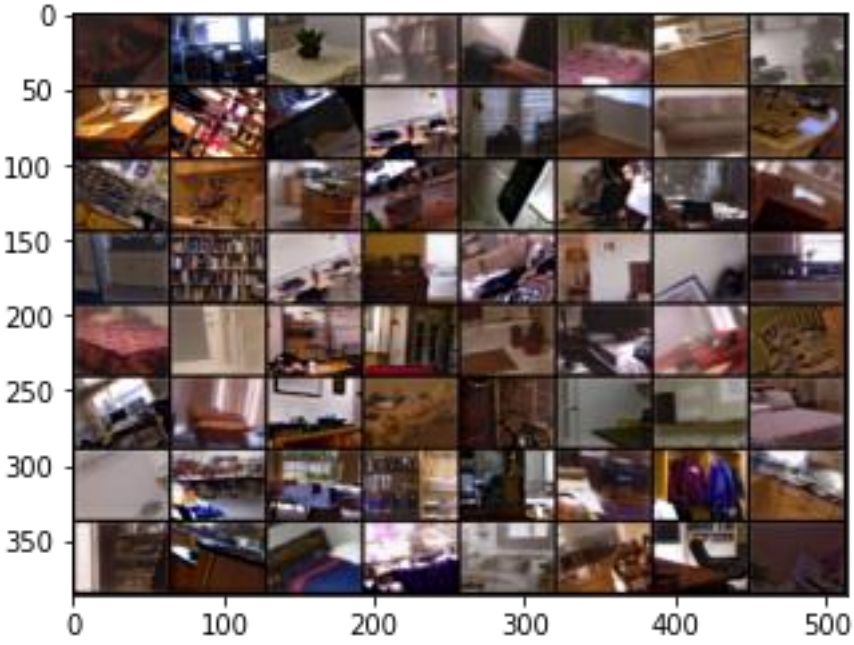
Indoor Hazy image :



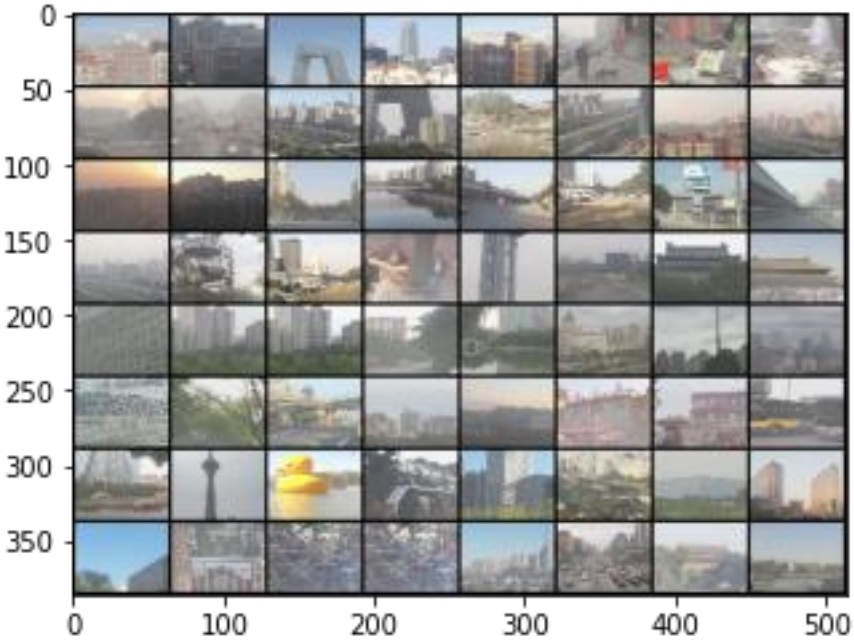
Indoor Clear image:



Indoor Dehazed image:

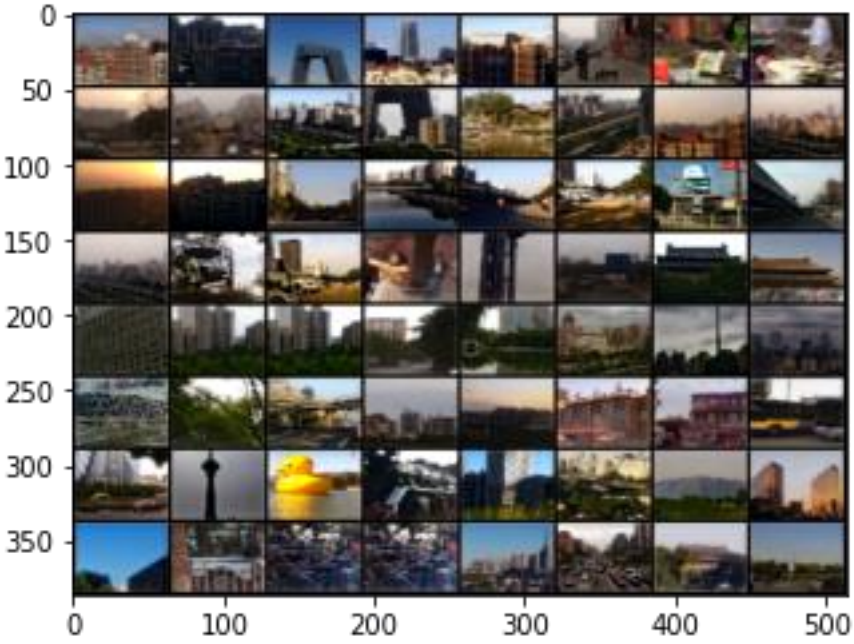


Outdoor Hazy image :



Outdoor Clear image:

Outdoor Dehazed image:

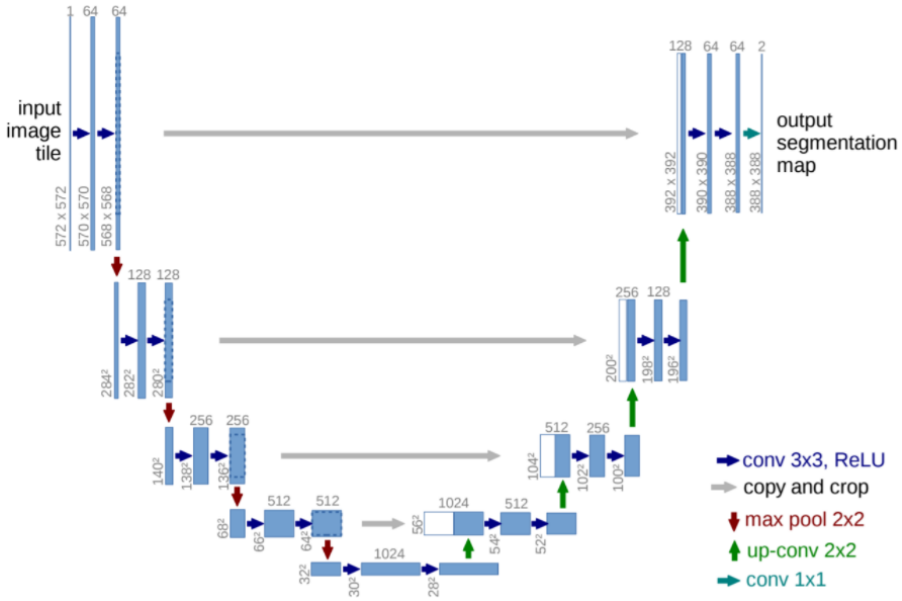


Evaluation Metrics – PSNR and SSIM

	Test SSIM	Test PSNR
Indoor	**0.6332830214500427**	**16.72899995422363**
Outdoor	**0.5486948914527893**	**15.9930496975975977**

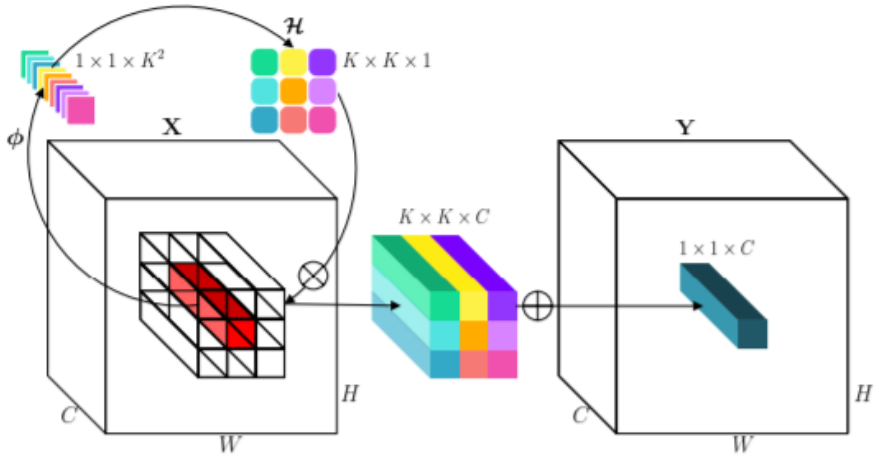
TASK 4: Implementing own architecture to obtain possible improvements of the current/near to SOTA.

U-Net architecture:



This architecture appears in U-shape hence the name U-Net which uses a fully convolution network model for the task. Here it consists of two parts – one is the encoder part (left side in the above architecture diagram) and other is the decoder part(right side in the above architecture diagram). The encoder consists of a stack of convolutions and max pooling layers, in other words the network will learn what information is in the image and decoder is used to enable the precise localization using transposed convolutions, in other words it helps to recover the where information by up sampling.

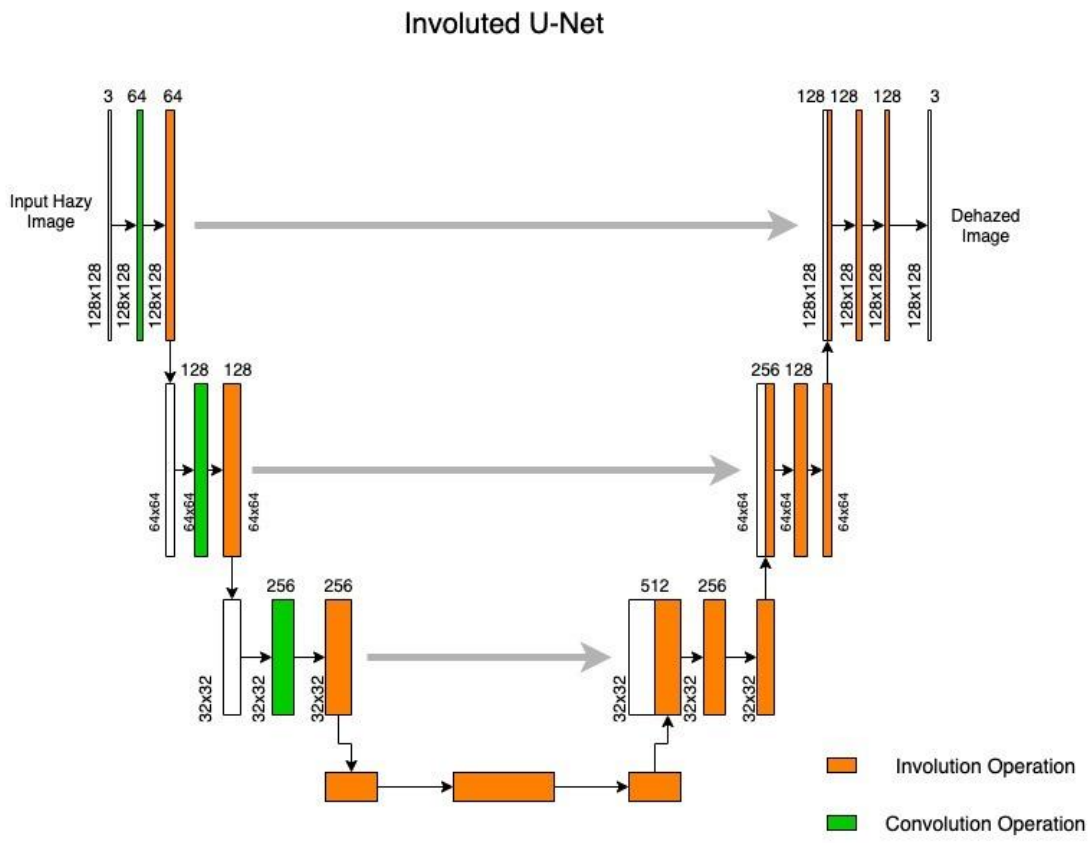
Involution:



In the rapid advances of neural network architectures, convolution remains the building mainstay of deep neural networks. The classical image filtering methodology has two main properties spatial-agnostic and channel-specific. Spatial-agnostic and spatial compact helps in enhancing the efficiency and interpreting the translation equivalence but it deprives the convolution kernel to adapt to the diverse visual patterns with respect to different spatial positions. To conquer this limitation, a novel atomic operation for deep neural networks by inverting the aforementioned design principles of convolution, coined as involution. Here involution bridges between the convolution and self-attention in the design and it is more effective and efficient than convolution and simpler than self-attention.

Using the above two concepts we have implemented a modified architecture called as **Involuted U-Net**.

We have implemented our own architecture to obtain a better performance motivated by U-Net architecture. The main difference is we have used involutions in place of convolutions and applied composite loss - perpetual (Alexnet) + SSIM loss + PSNR loss along with weightages. The architecture consists of two parts, one is the encoder part (left side in the below architecture diagram) and other is the decoder part(right side in the below architecture diagram). The number of channels considered for encoder are 64,128,256 and for the decoder are 256, 128, 64. To retain the dimensions of the images, all the involutions and convolutions are padded. In the output, instead of generating a single channel mask, were are generating an RGB dehazed image.



Composite Loss = 0.6 * Perpetual Loss (AlexNet) + 0.1 * SSIM Loss + 0.3* PSNR Loss

Model with U-Net and involution:

Layer (type)	Output Shape	Param #
Conv2d-1	[-1, 64, 128, 128]	192
Unfold-2	[-1, 576, 16384]	0
AvgPool2d-3	[-1, 3, 128, 128]	0
Conv2d-4	[-1, 64, 128, 128]	192
BatchNorm2d-5	[-1, 64, 128, 128]	128
ReLU-6	[-1, 64, 128, 128]	0
Conv2d-7	[-1, 9, 128, 128]	576
Involution2d-8	[-1, 64, 128, 128]	0
ReLU-9	[-1, 64, 128, 128]	0
Conv2d-10	[-1, 64, 128, 128]	36,928
ReLU-11	[-1, 64, 128, 128]	0
Block_en-12	[-1, 64, 128, 128]	0

```
        MaxPool2d-13           [-1, 64, 64, 64]               0
           Conv2d-14          [-1, 128, 64, 64]           8,192
           Unfold-15           [-1, 1152, 4096]               0
        AvgPool2d-16           [-1, 64, 64, 64]               0
           Conv2d-17          [-1, 128, 64, 64]           8,192
      BatchNorm2d-18          [-1, 128, 64, 64]             256
             ReLU-19          [-1, 128, 64, 64]               0
           Conv2d-20            [-1, 9, 64, 64]           1,152
     Involution2d-21          [-1, 128, 64, 64]               0
             ReLU-22          [-1, 128, 64, 64]               0
           Conv2d-23          [-1, 128, 64, 64]         147,584
             ReLU-24          [-1, 128, 64, 64]               0
         Block_en-25          [-1, 128, 64, 64]               0
        MaxPool2d-26          [-1, 128, 32, 32]               0
           Conv2d-27          [-1, 256, 32, 32]          32,768
           Unfold-28           [-1, 2304, 1024]               0
        AvgPool2d-29          [-1, 128, 32, 32]               0
           Conv2d-30          [-1, 256, 32, 32]          32,768
      BatchNorm2d-31          [-1, 256, 32, 32]             512
             ReLU-32          [-1, 256, 32, 32]               0
           Conv2d-33            [-1, 9, 32, 32]           2,304
     Involution2d-34          [-1, 256, 32, 32]               0
             ReLU-35          [-1, 256, 32, 32]               0
           Conv2d-36          [-1, 256, 32, 32]         590,080
             ReLU-37          [-1, 256, 32, 32]               0
         Block_en-38          [-1, 256, 32, 32]               0
        MaxPool2d-39          [-1, 256, 16, 16]               0
          Encoder-40  [[-1, 64, 128, 128], [-1, 128, 64, 64], [-1, 2
56, 32, 32]]               0
  ConvTranspose2d-41          [-1, 128, 64, 64]         131,200
           Conv2d-42          [-1, 128, 64, 64]         295,040
             ReLU-43          [-1, 128, 64, 64]               0
           Conv2d-44          [-1, 128, 64, 64]         147,584
             ReLU-45          [-1, 128, 64, 64]               0
         Block_de-46          [-1, 128, 64, 64]               0
  ConvTranspose2d-47         [-1, 64, 128, 128]          32,832
           Conv2d-48         [-1, 64, 128, 128]          73,792
             ReLU-49         [-1, 64, 128, 128]               0
           Conv2d-50         [-1, 64, 128, 128]          36,928
             ReLU-51         [-1, 64, 128, 128]               0
         Block_de-52         [-1, 64, 128, 128]               0
          Decoder-53         [-1, 64, 128, 128]               0
           Conv2d-54          [-1, 3, 128, 128]             195
================================================================
Total params: 1,579,395
Trainable params: 1,579,395
Non-trainable params: 0
----------------------------------------------------------------
```

```
Input size (MB): 0.19
Forward/backward pass size (MB): 1099511628116.73
Params size (MB): 6.02
Estimated Total Size (MB): 1099511628122.94
----------------------------------------------------------------
```

Training is carried out for 15 epochs with SGD optimizer, using learning rate 1e-2, weight decay of 0.01 and momentum 0.9 – displaying last few epochs:

========== Epoch: 11

Train loss: -4.477931434161042, Train SSIM: 0.5625017621927896, Train PSNR: 15.402120248823186

Test loss SOTS Indoor: -4.58300256729126, Test SSIM SOTS Indoor: 0.564318115234375, Test PSNR SOTS Indoor: 15.671393402099609

Test loss SOTS Outdoor: -4.48456805944442 75, Test SSIM SOTS Outdoor: 0.556801249341267, Test PSNR SOTS Outdoor: 15.481773314437246

========== Epoch: 12

Train loss: -4.398374682692088, Train SSIM: 0.5505708798670275, Train PSNR: 15.168653956884311

Test loss SOTS Indoor: -4.740819454193115, Test SSIM SOTS Indoor: 0.5974895257949829, Test PSNR SOTS Indoor: 16.205234985351563

Test loss SOTS Outdoor: -4.667151629924774, Test SSIM SOTS Outdoor: 0.5678127781162418, Test PSNR SOTS Outdoor: 16.130863461068007

========== Epoch: 13

Train loss: -4.3862909356208695, Train SSIM: 0.5472255715222935, Train PSNR: 15.132052006697638

Test loss SOTS Indoor: -4.5327398180961 61, Test SSIM SOTS Indoor: 0.5852424521446228, Test PSNR SOTS Indoor: 15.483044326782226

Test loss SOTS Outdoor: -5.1877481341362, Test SSIM SOTS Outdoor: 0.5790352128385529, Test PSNR SOTS Outdoor: 17.787031685433735

========== Epoch: 14

Train loss: -4.407835767693715, Train SSIM: 0.5563238682791196, Train PSNR: 15.192759141792477

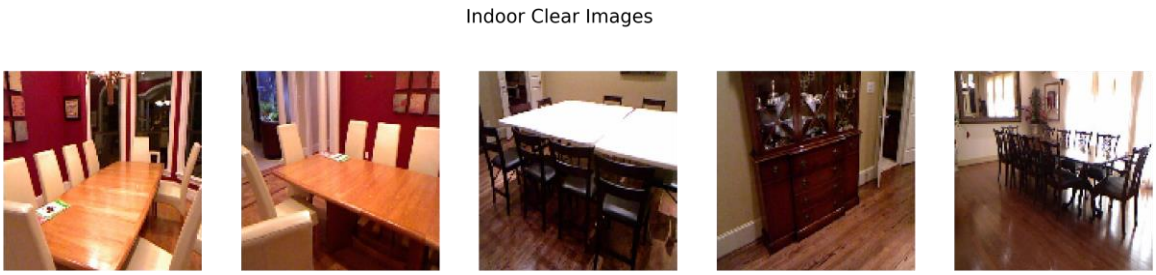
Test loss SOTS Indoor: -4.943126082420349, Test SSIM SOTS Indoor: 0.6073142008781434, Test PSNR SOTS Indoor: 16.847265869140625

Test loss SOTS Outdoor: -4.481691896915436, Test SSIM SOTS Outdoor: 0.5506985415288104, Test PSNR SOTS Outdoor: 15.506303709696947

Inference on the Best Model

Indoor hazy images:

Indoor clear images:



Indoor dehazed images:

Outdoor hazy images:

Outdoor clear images:

Outdoor dehazed images:

Evaluation Metrics – PSNR and SSIM

	Test Loss SOTS	Test SSIM	Test PSNR
Indoor	**-4.9431260824203 49**	**0.6073142008781434**	**16.847265869140625**
Outdoor	**-5.1877481341362**	**0.5790352128385529**	**17.7870316854337 35**

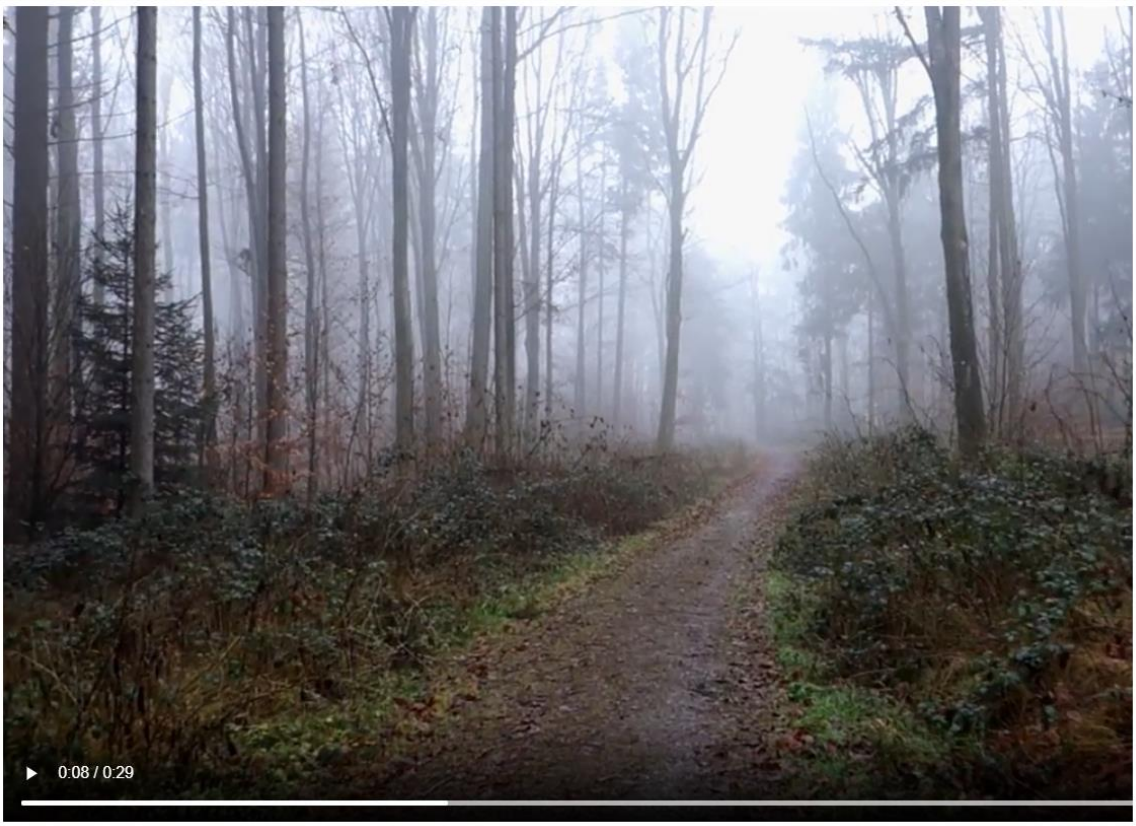
TASK 5: Apply improved pipeline to video data for dehazing

We built a video dehazing pipeline using our trained Involuted U-Net and applied it to hazy/foggy video

Results obtained:

We have shown the snippet of the video here in the report but the complete video is available in the results folder mentioned at the start of the report:

Hazy video snippet:



Dehazed video snippet:

Conclusion

- We have shown the results on the pipeline developed for image enhancing via dehazing in task1. We have obtained better results on the indoor images and comparable results on the outdoor images considering the PNSR and SSIM metrics.
- We have implemented and reproduced the results of the FFA-Net paper for both – model trained from scratch and by using pretrained model. For the pretrained model we have resized the image to 128 x 128 and for the model trained from scratch, the images are resized to 46 x 62 due to the GPU limitations (discussion in the next section).
- We have implemented a modified architecture – we have used involutions in place of convolutions and applied composite loss - perpetual (Alexnet) + SSIM loss + PSNR loss along with weightages. We have obtained a better result than others.
- The visual quality of the images obtained in Task4 doesn't appear to have the realistic colors. This can be possibly be improved by considering the perpetual loss of VGG.
- Also, if this model is trained on the higher image size or on the original size of the images then we can obtain much better results and visual quality compared to the results obtained FFA-Net paper (possibly by using better GPU's).
- We have implemented all the proposed tasks and we have clearly shown that our own architecture implemented in task4 is giving better results compared to others.

Methods	Indoor		Outdoor	
	PSNR	SSIM	PSNR	SSIM
DCP	14.77	0.7757	22.65	0.9226
DCP with Preprocessing and Postprocessing(Pipeline1)	11.83	0.678	15.7	0.7712

DCP with Preprocessing and Postprocessing(Pipeline2)	11.15	0.5116	14.93	0.6395
FFA-Net on pretrained model	14.86	0.5559	19.41	0.6363
Ours	**16.85**	**0.6073**	**17.78**	**0.5790**

Below are the methods used where the image size are not 128 x 128 so it is not considered for the comparison.

Methods	Indoor		Outdoor	
	PSNR	**SSIM**	**PSNR**	**SSIM**
DCP	14.91	0.7808	21.99	0.9086
DCP with Preprocessing and Postprocessing(Pipeline1)	15.08	0.7487	18.55	0.8236
DCP with Preprocessing and Postprocessing(Pipeline2)	14.89	0.7091	18.07	0.7580
FFA-Net on the model trained from scratch(resized image)	**16.73**	**0.6333**	**15.99**	**0.5487**

Limitations

The main limitation was with respect to the GPU usage, as we had to run all the codes on colab / colab pro, we had to compromise on things listed below.

- We have resized the images to 128 x 128 due to GPU limitations (in task1).
- We weren't able to run the inference on the test dataset with the actual image size so we had to resize it for both pre-trained and the model trained from scratch (in task3).
- We weren't able to use perpetual loss using VGG due to GPU limitations. So, we have implemented the AlexNet perpetual loss (in task4 and 5).
- We weren't able to replace all the convolutions in the model with involutions. So, we have just replaced for few in the encoder (in task4 and 5).

- We weren't able to add residual connections in both encoder and decoder (in task4 and 5).
- We weren't able to train the model with higher batch sizes.

References

[1] FFA-Net: Feature Fusion Attention Network for Single Image Dehazing, Xu Qin1 et al. AAAI'20

[2] NH-HAZE: An Image Dehazing Benchmark with Non-Homogeneous Hazy and Haze-Free Images, Codruta O. Ancuti et al. CVPR'20

[3] Multi-Scale Boosted Dehazing Network with Dense Feature Fusion, Hang Dong et al. CVPR'20

[4] Single Image Reflection Removal Beyond Linearity, Qiang Wen et al. CVPR'20

[5] Fast Single Image Reflection Suppression via Convex Optimization, Yang Yang et al. CVPR'19

[6] Distilling Image Dehazing With Heterogeneous Taks Imitation, Ming Hong et al, CVPR'19

[7] https://github.com/zhilin007/FFA-Net/blob/master/net/metrics.py

[8] https://scikit-image.org/docs/dev/api/skimage.metrics.html#skimage.metrics.structural_similarity

[9] https://amaarora.github.io/2020/09/13/unet.html

[10] https://kornia.readthedocs.io/en/latest/losses.html

[11] https://medium.com/pytorch/multi-target-in-albumentations-16a777e9006e

[12] https://github.com/richzhang/PerceptualSimilarity

[13] https://richzhang.github.io/PerceptualSimilarity/

[14] https://discuss.pytorch.org/t/torchvision-transfors-how-to-perform-identical-transform-on-both-image-and-target/10606/7

[15] https://github.com/yyhz76/reflectSuppress

[16] https://sanjivgautamofficial.medium.com/perceptual-loss-well-it-sounds-interesting-after-neural-style-transfer-d09a48b6fb7d

[17] https://github.com/ChristophReich1996/Involution

[18] https://arxiv.org/abs/2103.06255]