Name: Batuhan

Surname: Yalçın

ID: 64274

COMP527/ELEC519, Spring 2023- Programming Assignment 4: Deep Low-light Image Enhancement

IMPORTANT

In the calculation of the PSNR and SSIM metrics, I used the skimage.metric library. These metrics require the true image and the test image as inputs. However, in our case, we didn't have a true image as a reference. Instead, we used PIL autocontrast to enhance the images. Interestingly, after 50 epochs, the test image was better than the reference image. Consequently, even though the images were improved compared to the reference, the PSNR and SSIM metrics showed smaller values.

Lower PSNR and SSIM values are not considered good in image enhancement tasks. PSNR (Peak Signal-to-Noise Ratio) measures the quality of an enhanced image by comparing it to the reference (original) image. Higher PSNR values indicate a closer match between the enhanced image and the reference, suggesting better image quality.

Similarly, SSIM (Structural Similarity Index) assesses the structural similarity between the enhanced image and the reference image. Higher SSIM values indicate a higher level of similarity, implying better image quality.

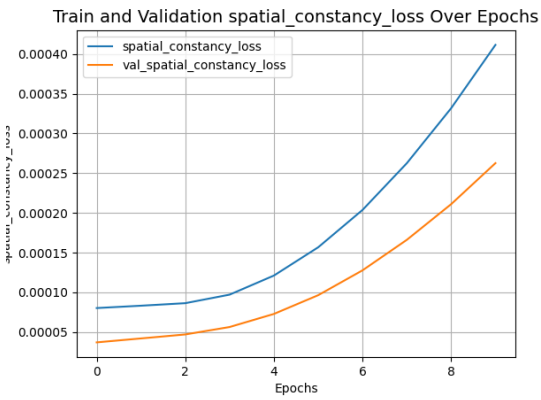
However, since the referenced image is autocorrected and not the true value, it can sometimes introduce wrong interpolations. In our task, the best image for me was the one that displayed more details, preserved features, and appeared to be captured in daylight. These qualitative factors can provide a better assessment of image enhancement performance.

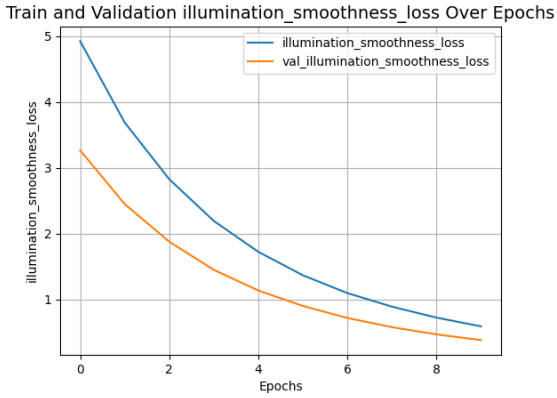
Table of Contents

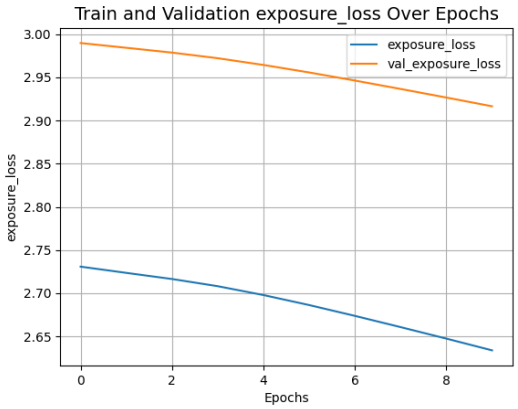
Part 1 Enhancement via DCE-Net*)

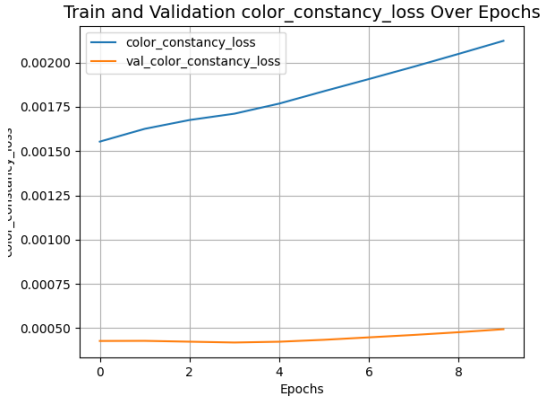
Epoch 10:

Loss functions:









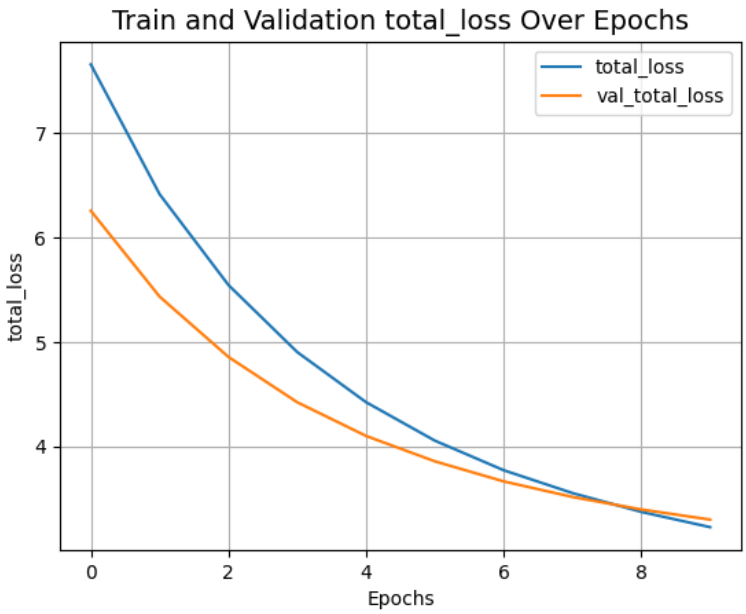
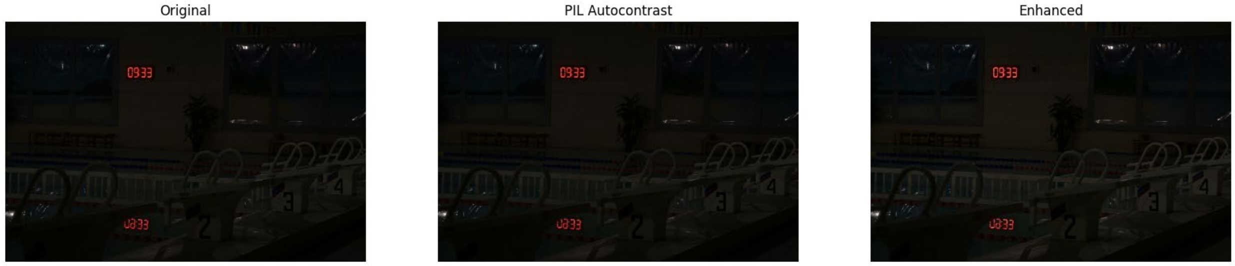


Image Results:

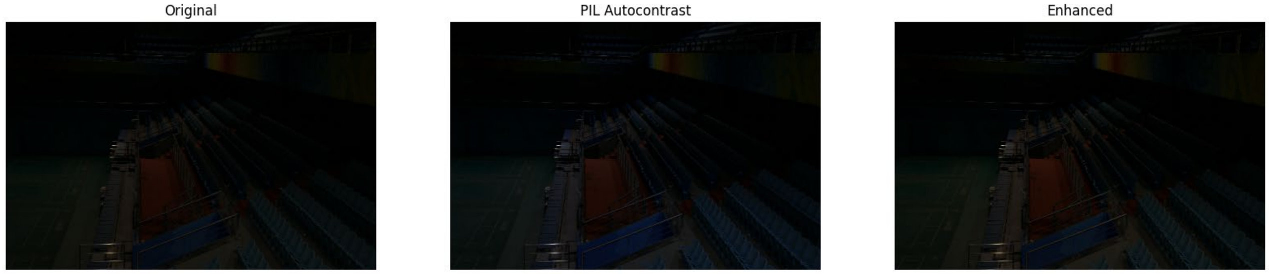
Original | PIL Autocontrast | Enhanced



Original | PIL Autocontrast | Enhanced

Original | PIL Autocontrast | Enhanced

Original | PIL Autocontrast | Enhanced



Original | PIL Autocontrast | Enhanced

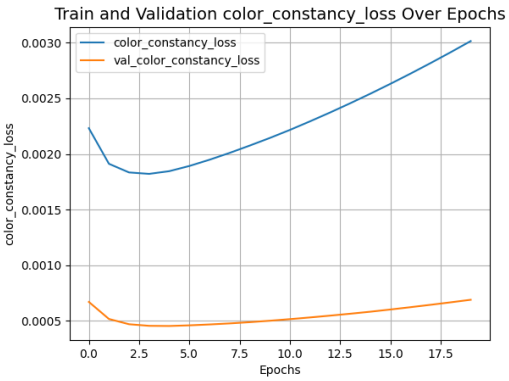
Metric Results:

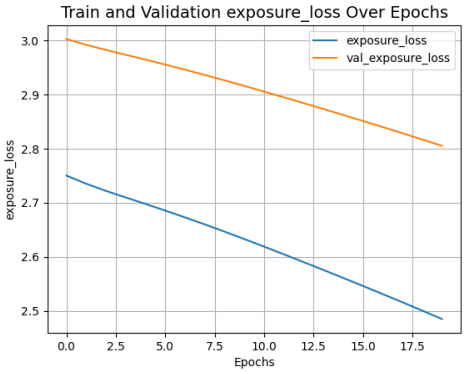
Average PSNR_referanced: 21.58981911537362

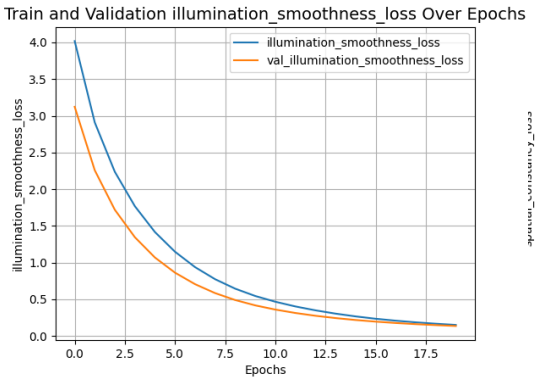
Average SSIM_ referanced: 0.6543375356002495

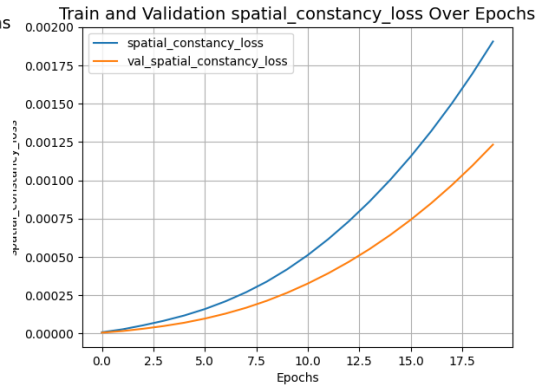
Epoch 20:

Loss functions:









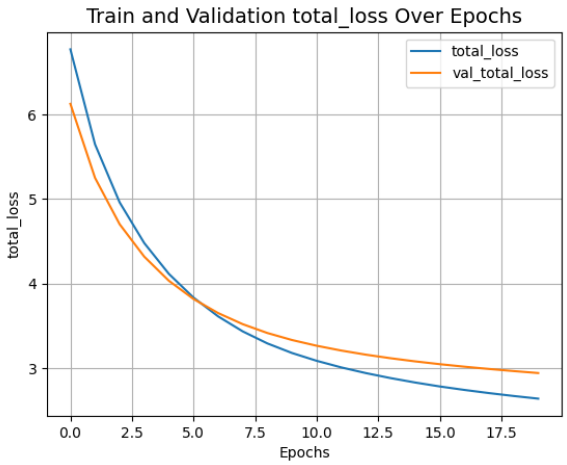
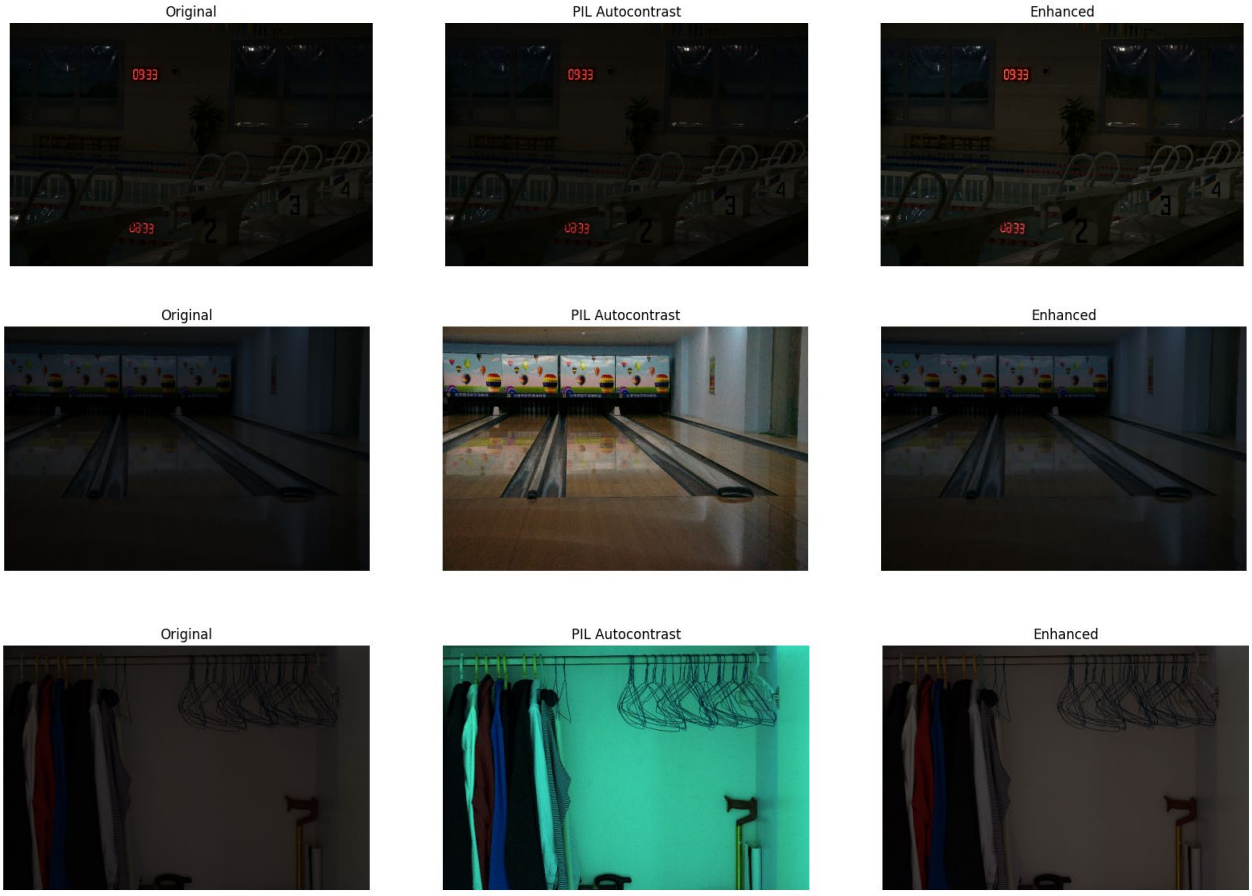
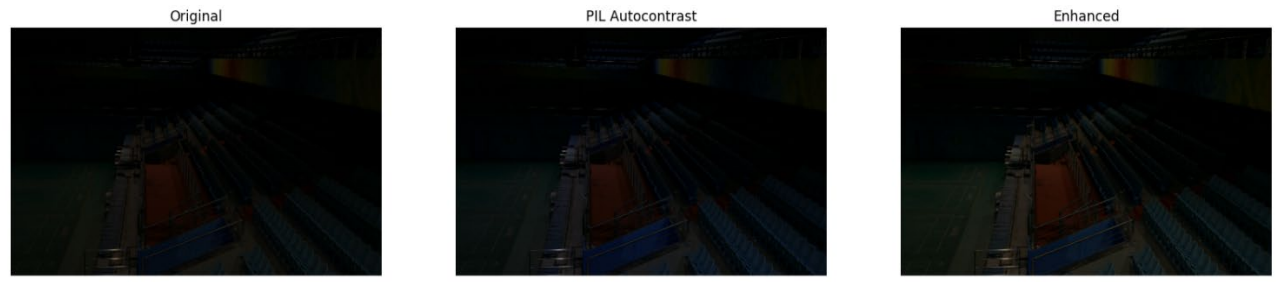


Image Results:

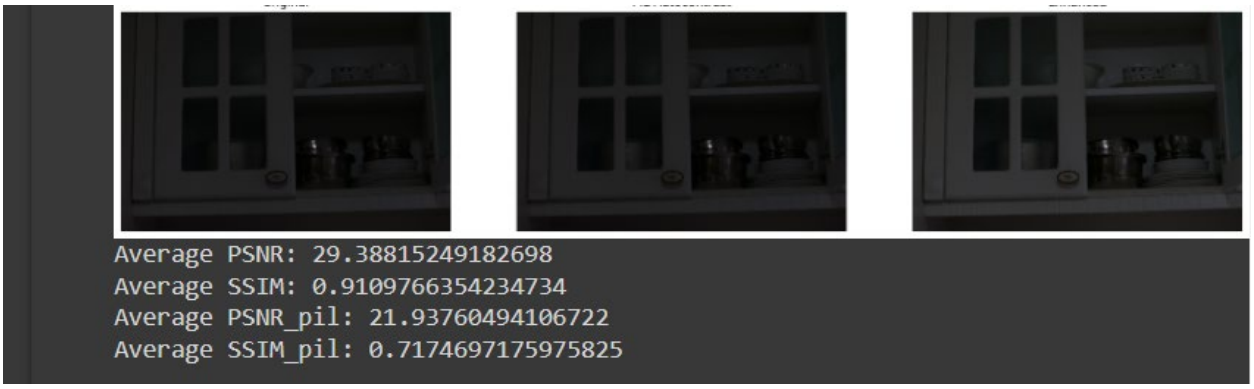




Metric Results:

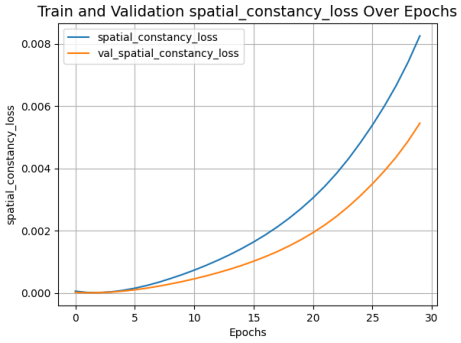
Average PSNR_pil: 21.93760494106722

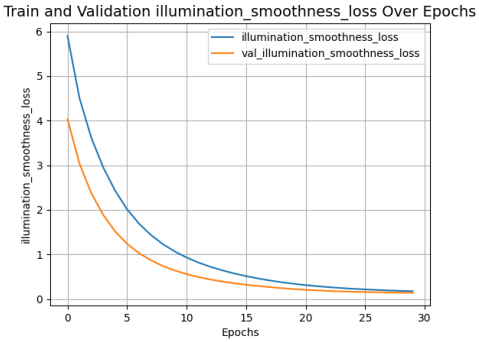
Average SSIM_pil: 0.7174697175975825

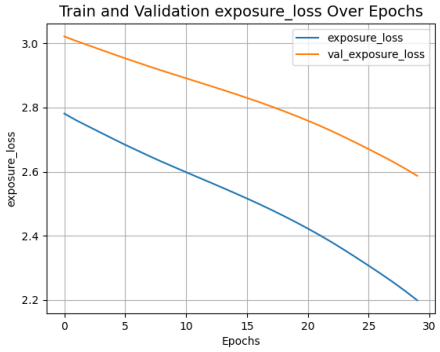


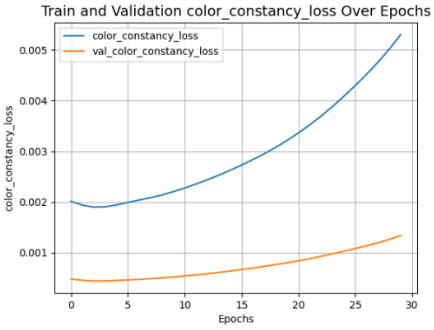
Epoch 30:

Loss functions:









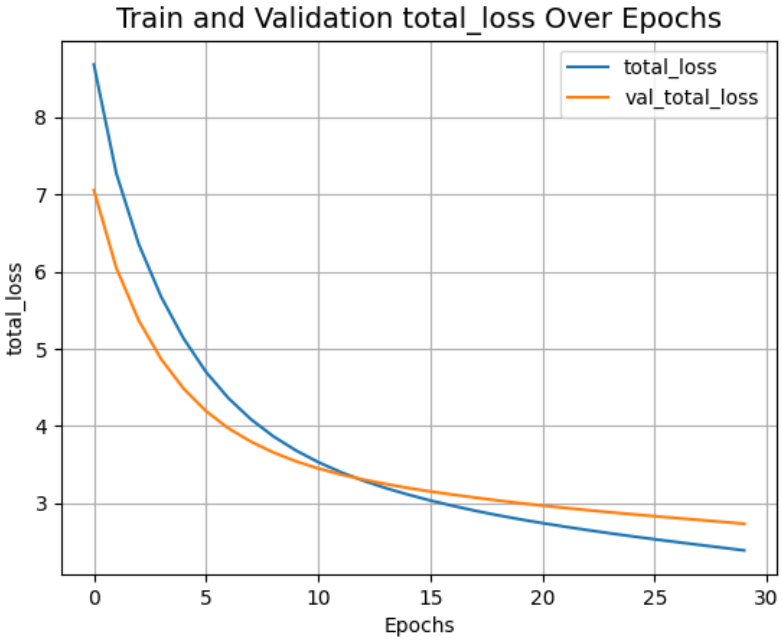
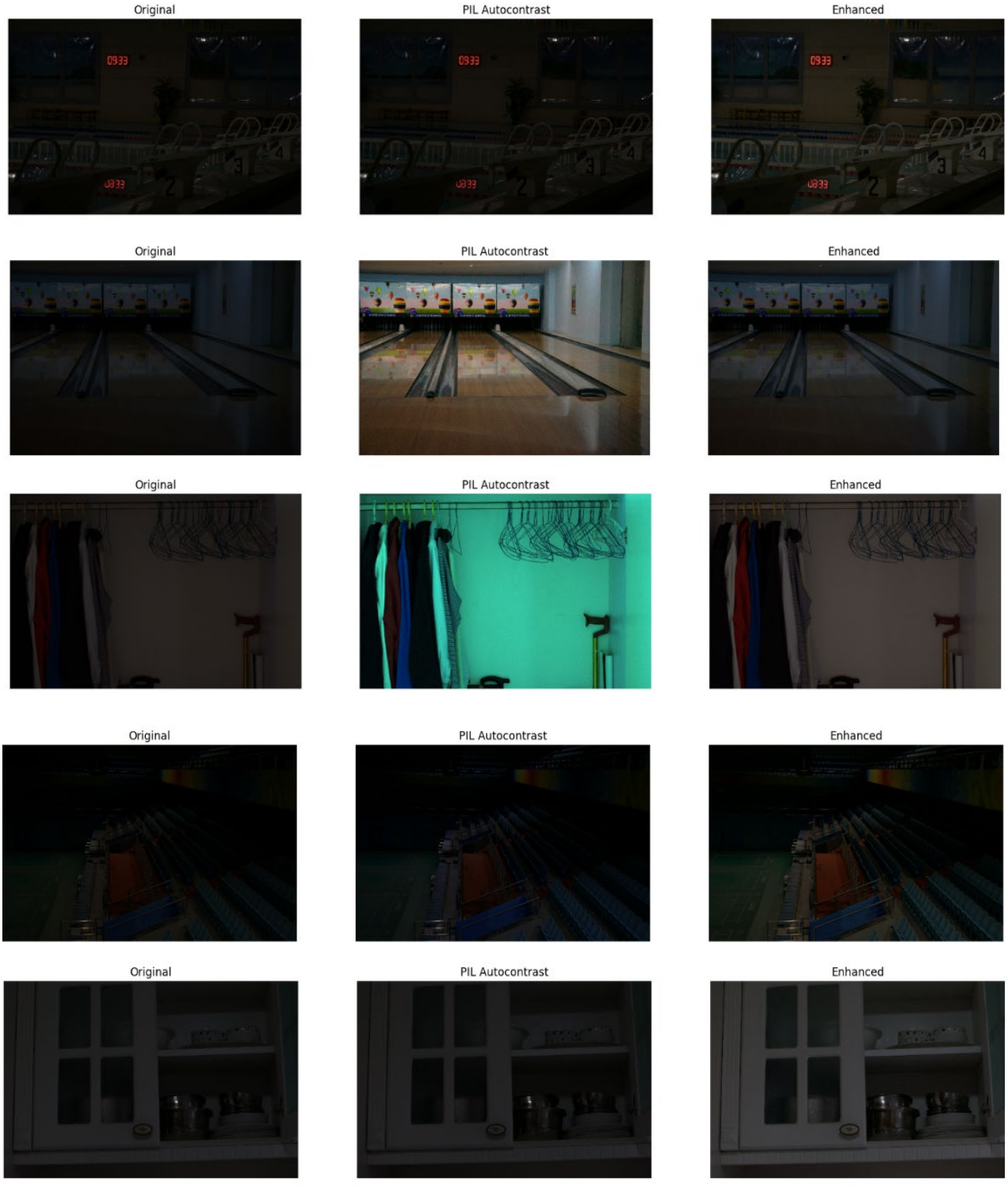


Image Results:



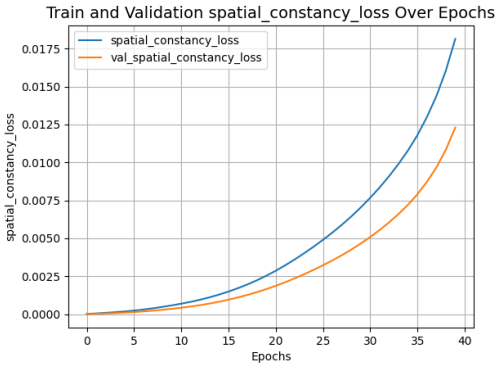
Metric Results:

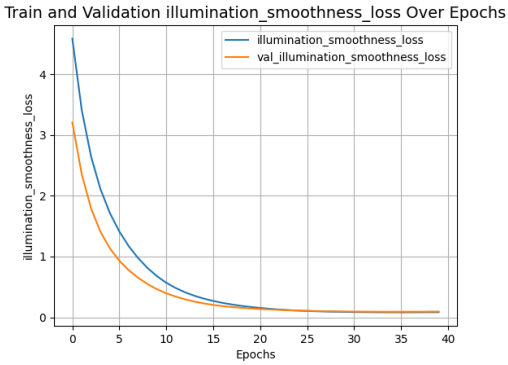
Average PSNR_pil: 23.162757053925002

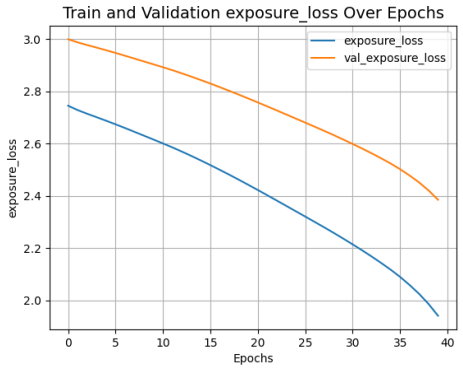
Average SSIM_pil: 0.7540703607440307

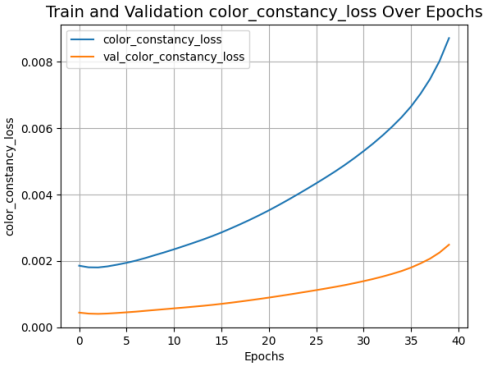
Epoch 40:

Loss functions:









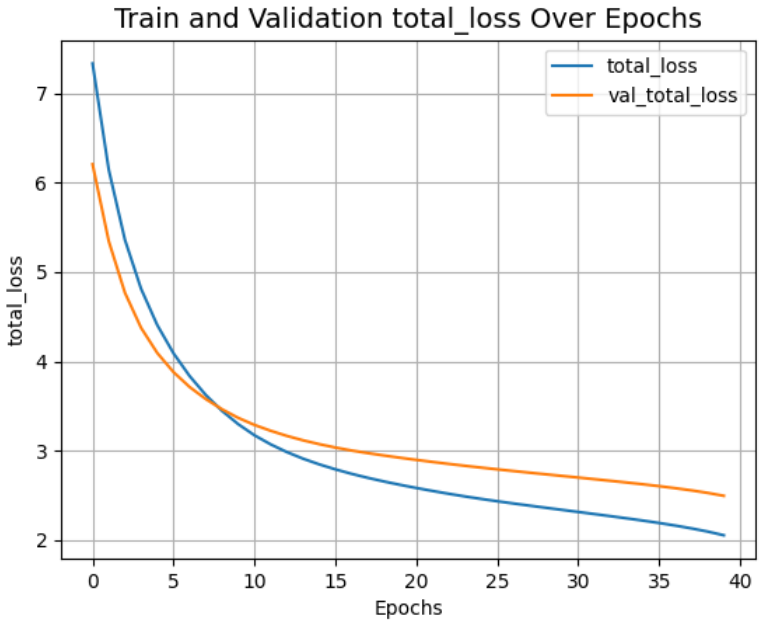
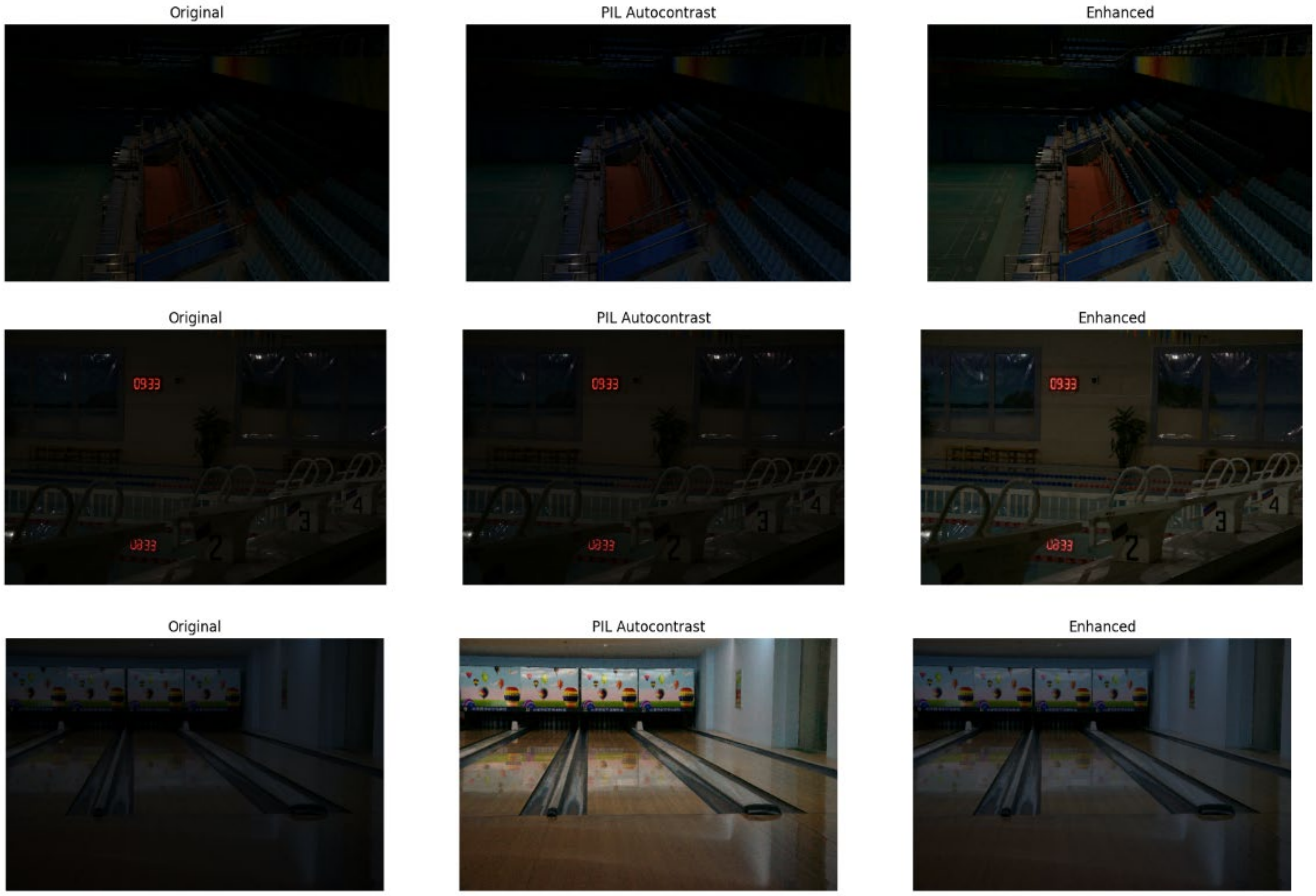
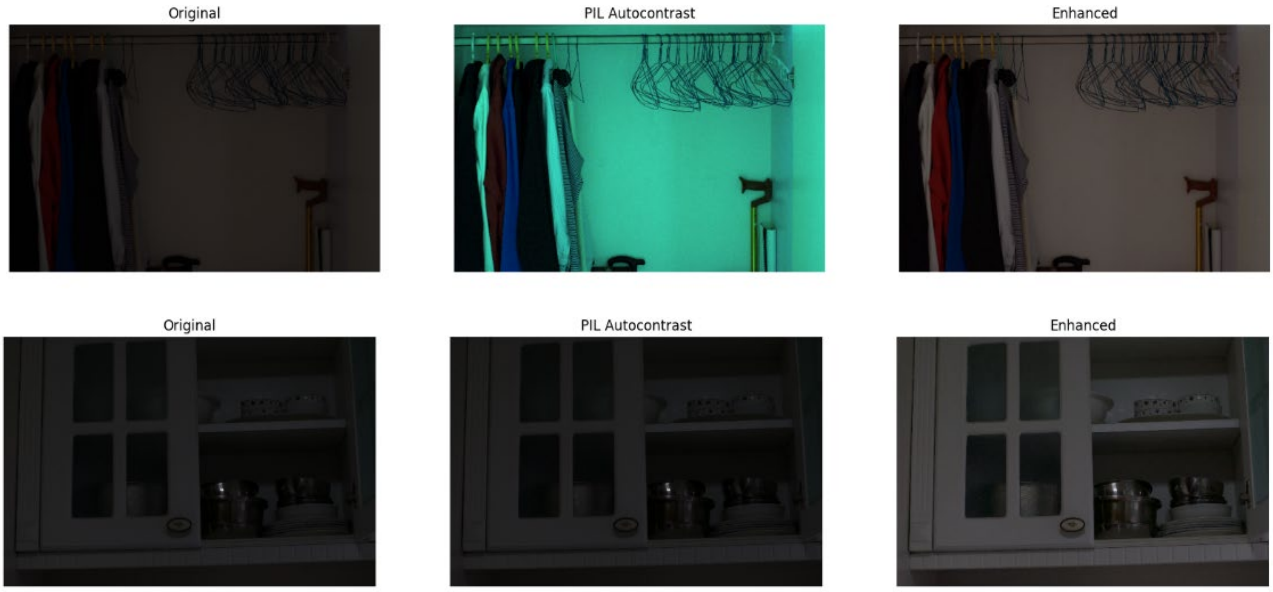


Image Results:

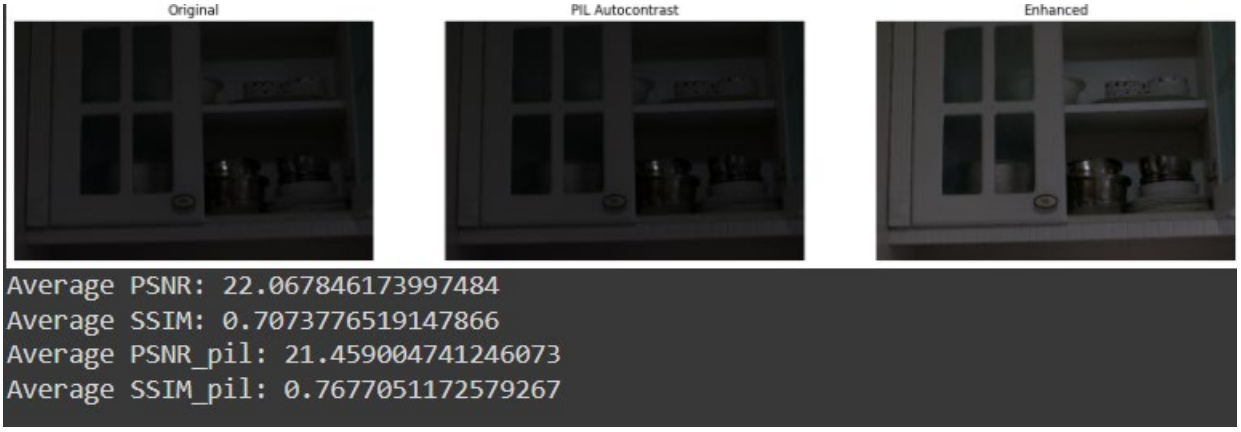




Metric Results:

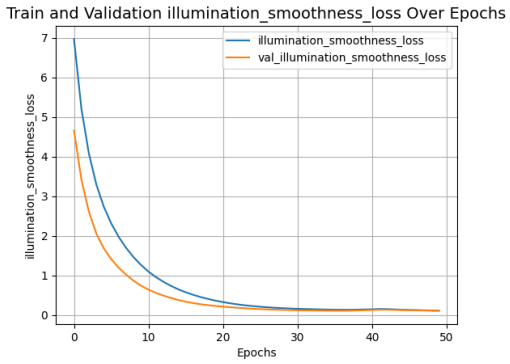
Average PSNR_pil: 21.459004741246073

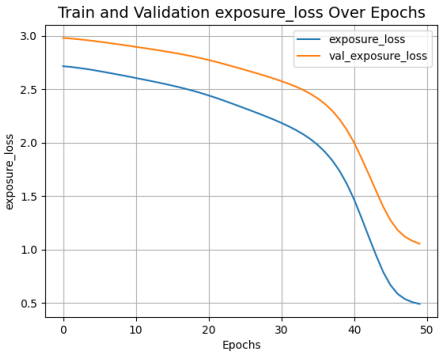
Average SSIM_pil: 0.7677051172579267

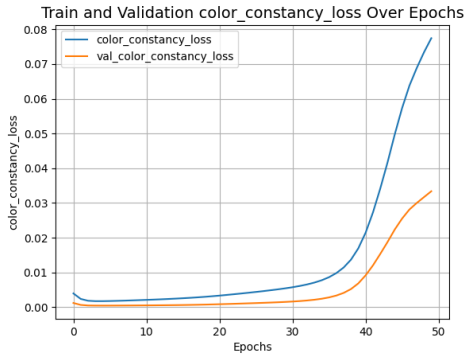


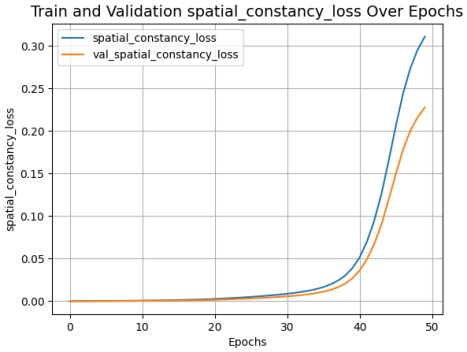
Epoch 50:

Loss functions:









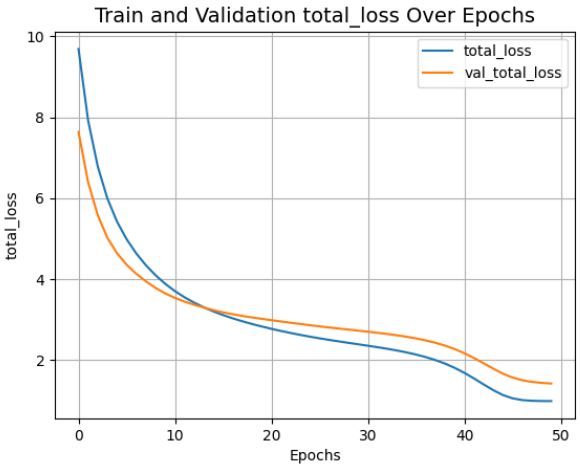
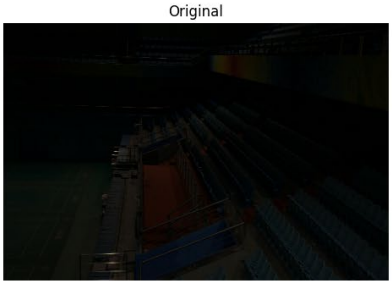
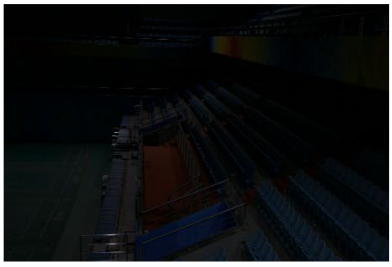


Image Results:

Original



PIL Autocontrast



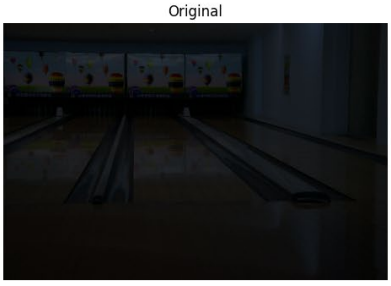
Enhanced

Original

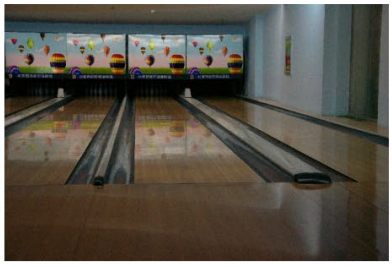
PIL Autocontrast

Enhanced

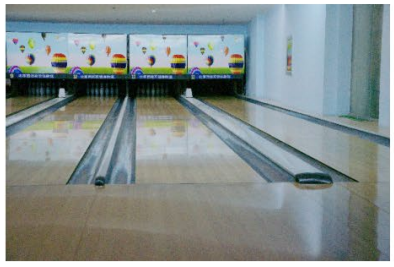
Original



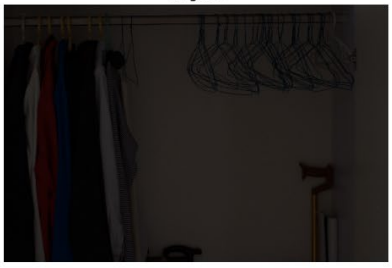
PIL Autocontrast



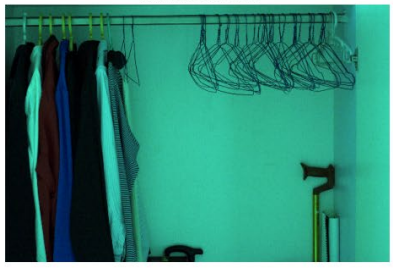
Enhanced



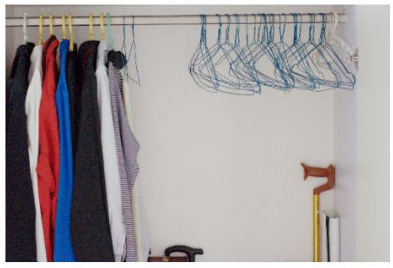
Original



PIL Autocontrast



Enhanced



Original

PIL Autocontrast

Enhanced

Metric Results:

Average PSNR_pil: 11.419754132146842

Average SSIM_pil: 0.4710940770635877

Part 1 Results Summary:

Qualitative Results:

Increasing the number of epochs significantly improves the image quality. The image quality improves as the number of epochs increases. However, after around 10 to 20 epochs, the resulting image quality becomes very close to the reference image, except for some lightly illuminated scenes, such as the bowling studio photo. When the number of epochs reaches 50, we observe better results even perfect for me (it depends to people).

Quantitative Results:

The differences in loss functions between the training and validation sets become smaller as the number of epochs increases, especially in the Total loss function. Around epoch 13, the training loss surpasses the validation loss and becomes larger. In a normal application, this could indicate overfitting. However, in our case, the image quality continues to improve. It is true that after this epoch, the results become very close to the reference images.

When considering the PSNR and SSIM metrics, value increases as the number of epochs increases to the 30. Then it decreases. Actually, it should not happen because when the number of epochs increases the image quality should increase also the PNSR should increase. Because bigger PNSR means better image. I compared both PIL images and original images separately as reference images. Since the autocorrection process can introduce noise, especially at epoch 50, the PSNR and SSIM values are lower when using the original image as a reference. However, using PIL images as reference images is a more accurate approach, and I interpreted my results based on the comparisons with PIL images.

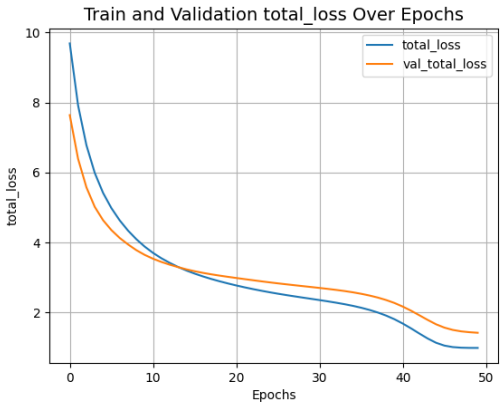
In epoch 50, there is a significant decrease in the PNSR and SSIM, even though resulting in very good image quality. The resulting image is very close to the image shown in the beginning of the PDF description, demonstrating the effectiveness of the model. However, since the referenced image is very different than this image, the PNSR and SSIM values get small. It also related our image is better than the referenced value.

In addition, while increasing the epoch number total losses, illumination and exposure losses decreasing, the spatial constancy and color constancy errors are increased. However, since their effect not too much in total losses. The total losses decreased when the epoch number increase.

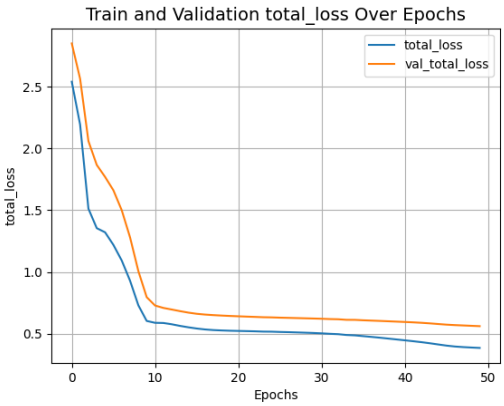
Part 2 Loss Functions)

Illumination loss:

Loss functions:



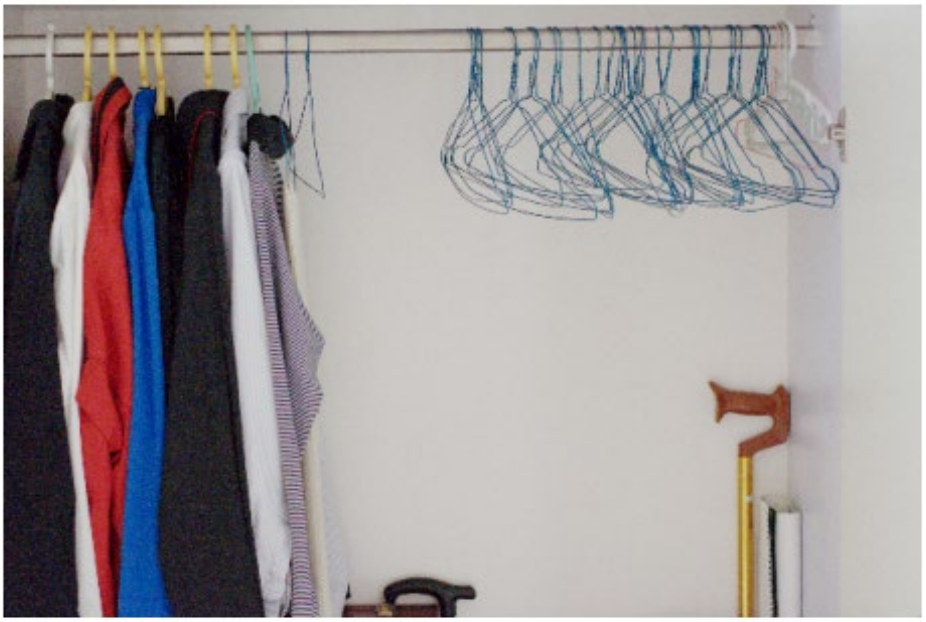
Original



New implementation

Image Results:

Enhanced



Original

Enhanced

New Implemented

Enhanced

Original

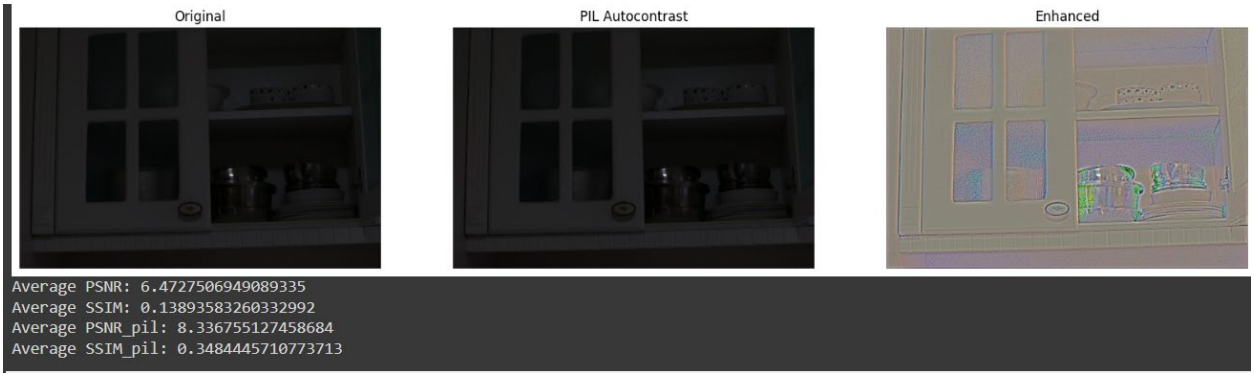
Enhanced

New Implemented

Metric Results:

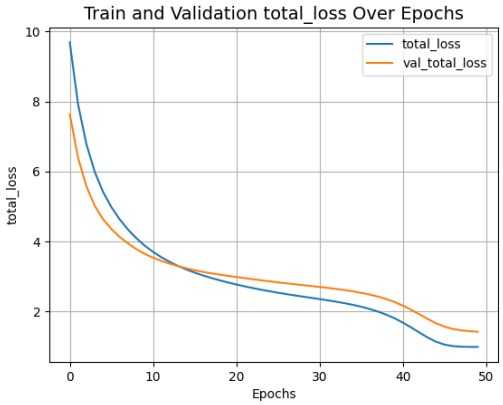
Average PSNR_pil: 8.336755127458684

Average SSIM_pil: 0.3484445710773713

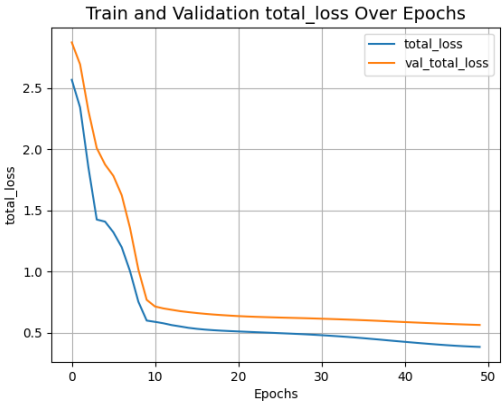


Spatial Constancy loss:

Loss functions:

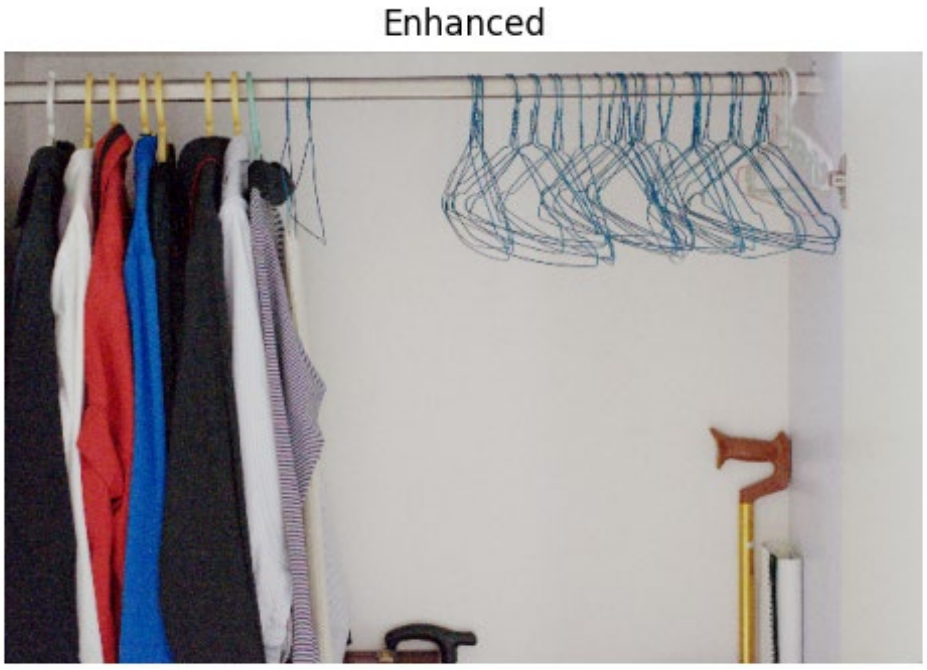


Original

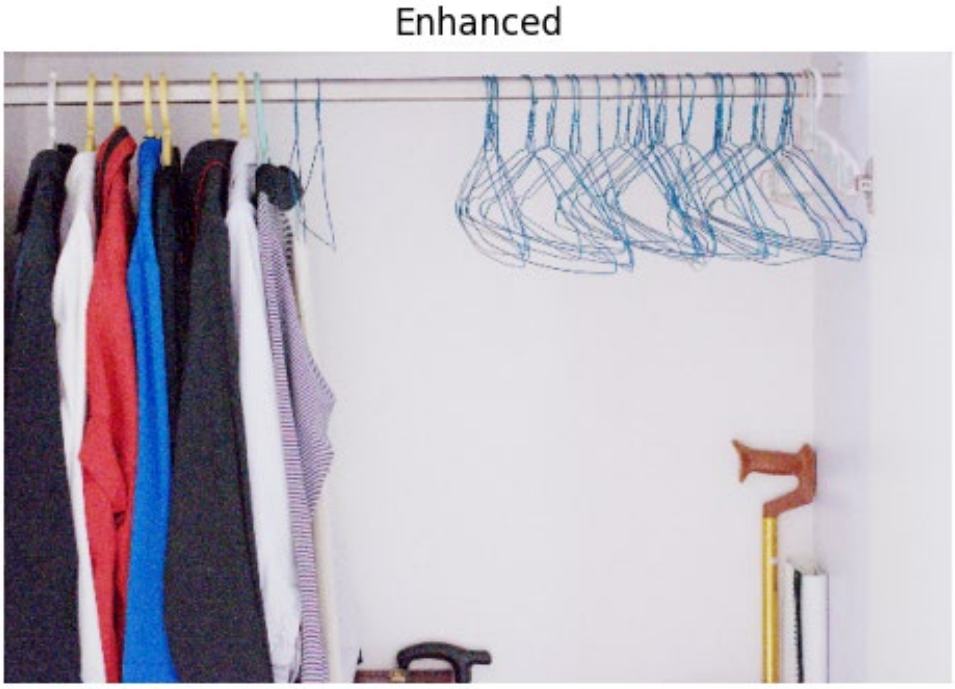


New Implemented

Image Results:



Original



New Implemented

Enhanced

Original

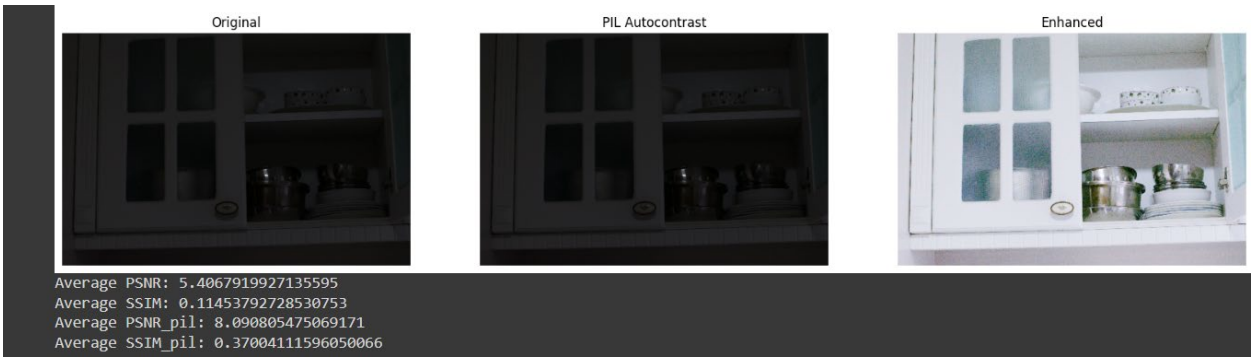
Enhanced

New Implemented

Metric Results:

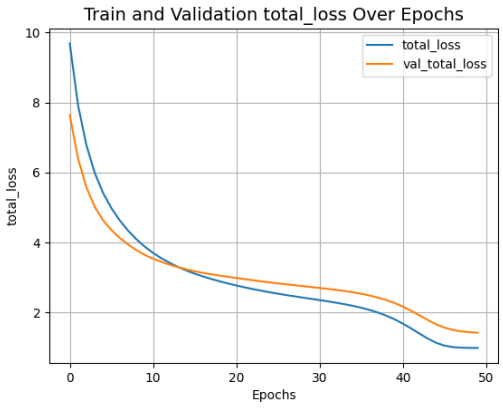
Average PSNR_pil: 8.090805475069171

Average SSIM_pil: 0.37004111596050066

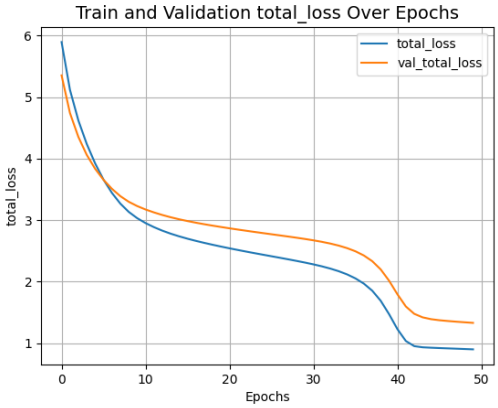


Color Constancy loss:

Loss functions:

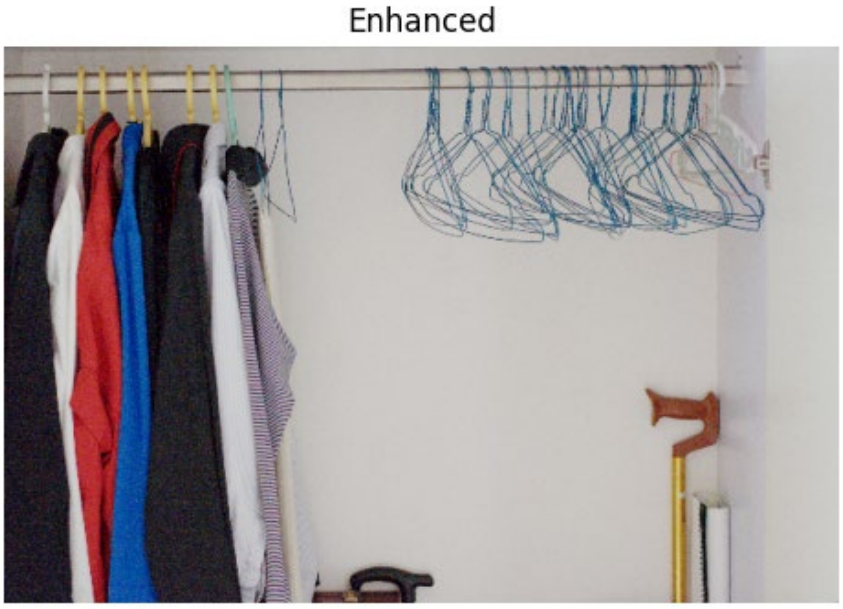


Original

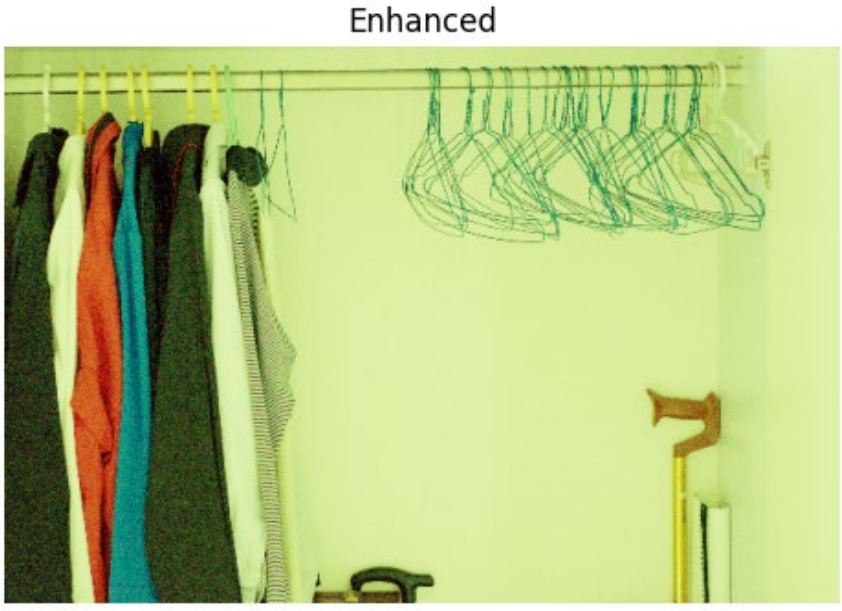


New Implemented

Image Results:



Original



New Implemented

Enhanced

Original

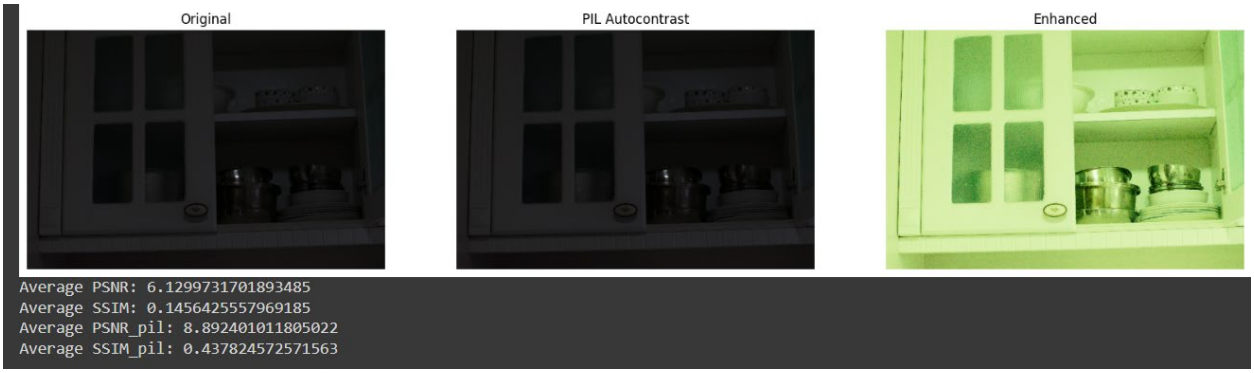
Enhanced

New Implemented

Metric Results:

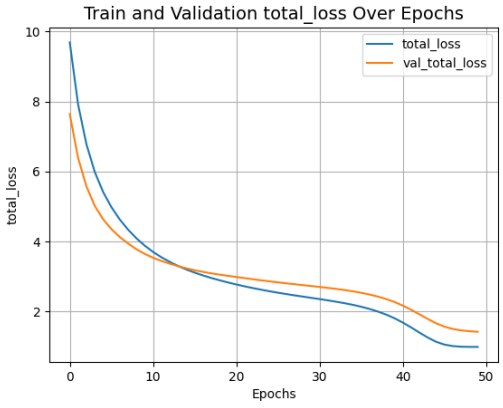
Average PSNR_pil: 8.892401011805022

Average SSIM_pil: 0.437824572571563

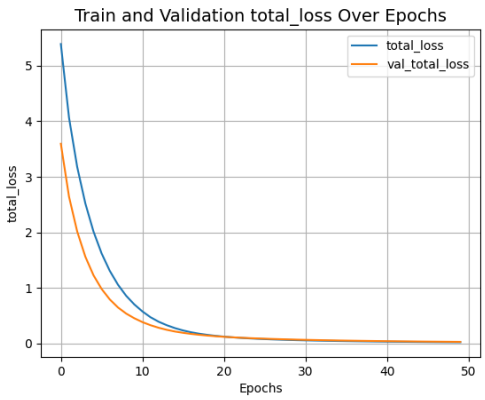


Exposure loss:

Loss functions:

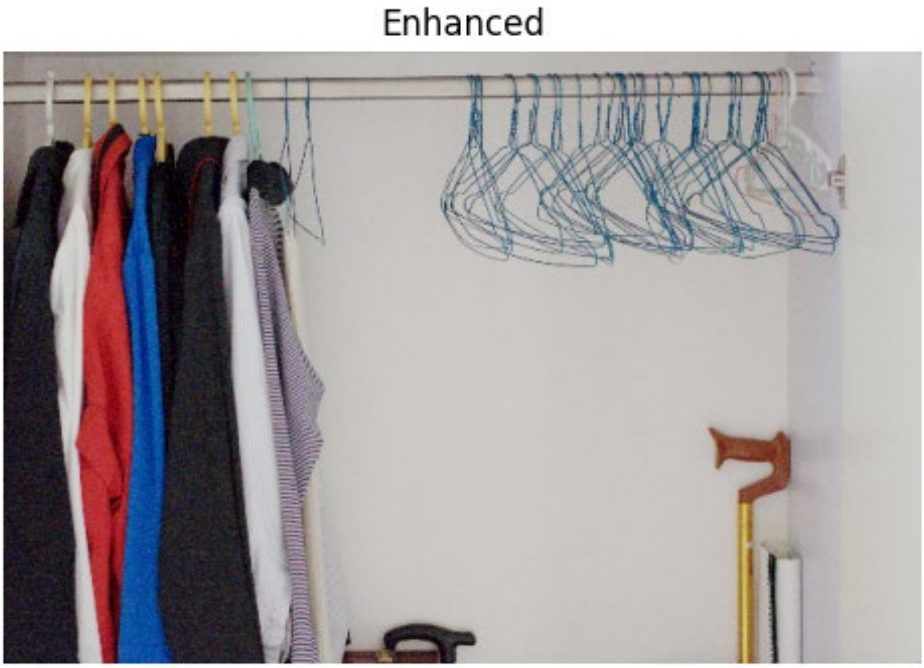


Original

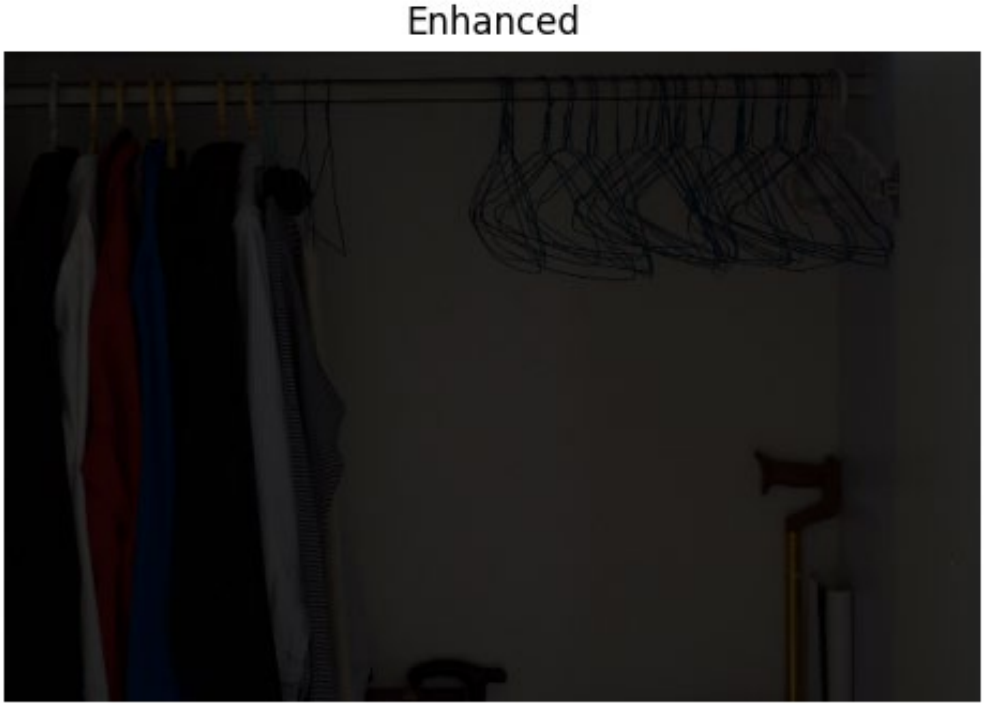


New Implemented

Image Results:



Original

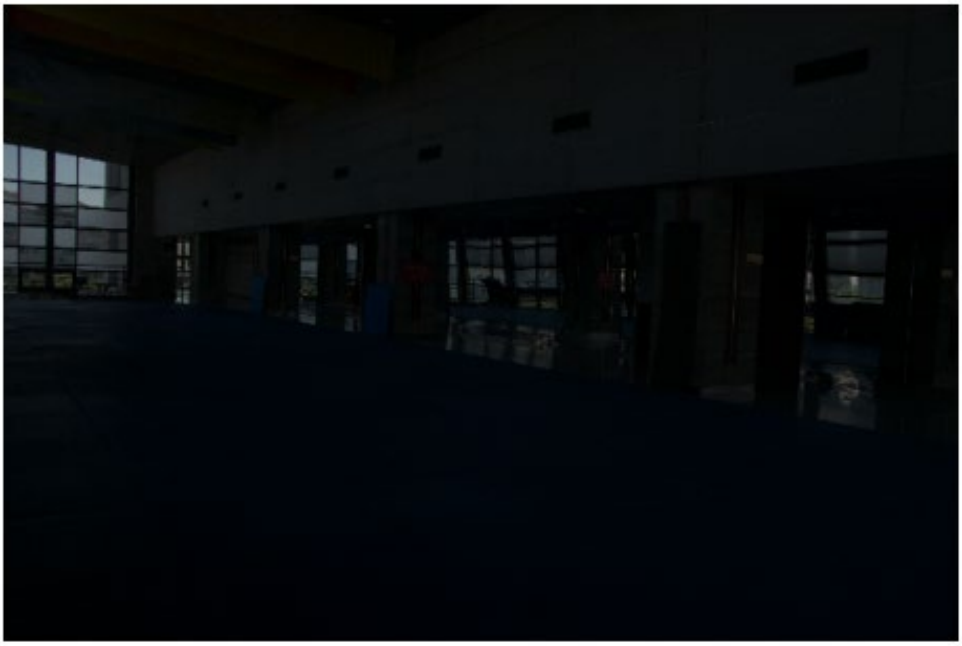


New Implemented

Enhanced

Original

Enhanced

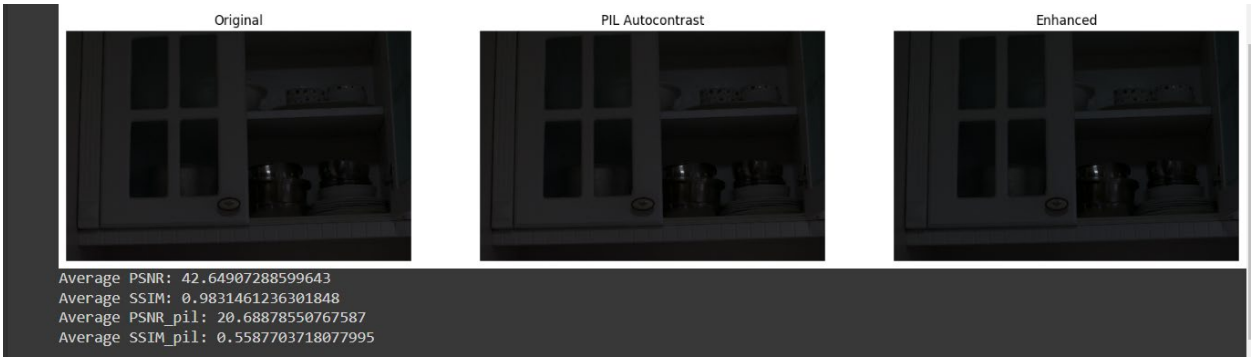


New Implemented

Metric Results:

Average PSNR_pil: 20.68878550767587

Average SSIM_pil: 0.5587703718077995



Part 2 Results Summary:

Qualitative Results:

Without Illumination Smoothness Loss:

This model's improved photos might have reduced illumination component smoothness. Images may be improved as a result, perhaps with sharper brightness shifts or more obvious lighting differences. As a result, my images become like they are in gray world.

Without Spatial Constancy Loss:

The augmented images from this model might not be spatially consistent, which would lead to irregularities in the overall structure or edge preservation. Overall, I couldn't see too much loss other than the edge conservation.

Without Spatial Constancy Loss:

The improved photos produced by this model might have less consistent color, which could lead to color shifts or other irregularities in the enhanced images. It's possible for colors to appear less realistic or to differ in different areas of the image. In most of the image the images was like yellowish with loosing white balance.

Without Exposure Loss:

Since Exposure related with the brightness in other word amount of the light captured, this loss directly effects the brightness. It gave the darker image overall. In opposite situation it may give the brighter results.

Quantitative Results:

Without Illumination Smoothness Loss:

In comparison to the initial total loss model, the enhanced images' average PSNR and SSIM scores were a little lower. This suggests that there may be a drop in image integrity and similarity to the enhanced reference photos.

The total loss graph also changed. The total losses are decreased significantly it means that most of the loss error comes from Illumination loss. Also, the train and validation loss never cross each other.

Without Spatial Constancy Loss:

In similar, in comparison to the initial total loss model, the enhanced images' average PSNR and SSIM scores might be a little lower. This raises the possibility of a drop in image quality and likeness to the enhanced reference photos.

Again, the total loss graph also changed and react very similar to the removing illumination loss.

Without Spatial Constancy Loss:

Again, the average PSNR and SSIM scores for the enhanced images may be slightly lower compared to the original total loss model. This indicates a potential decrease in color fidelity and similarity to the reference enhanced images.

The changes in total loss graph this time was not significant, but this after the epoch number increase the loss differences between train and validation increases, also they converge same loss in small epoch numbers.

Without Exposure Loss:

This time, the average PSNR and SSIM scores for the enhanced images was slightly bigger compared to the original total loss model. Again, I except it may be slightly lower compared to the original total loss model, but it didn’t. This indicates a potential increase in exposure correction (compared to reference) and similarity to the reference enhanced images. However, I was waiting the potential decrease in exposure correction.

The total loss graph effect was less then illumination, but bigger than the Spatial Constancy. However, this time after around epoch 18 the validation and train losses are converging the same value, and increasing the epoch number is not changed this converge, while decreasing the total error.

Part 3 Neural Architecture)

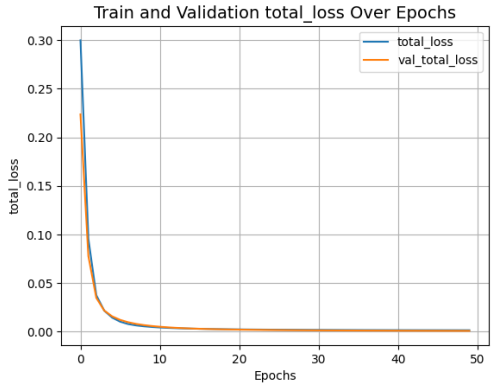
Removing the symmetrical concatenation:

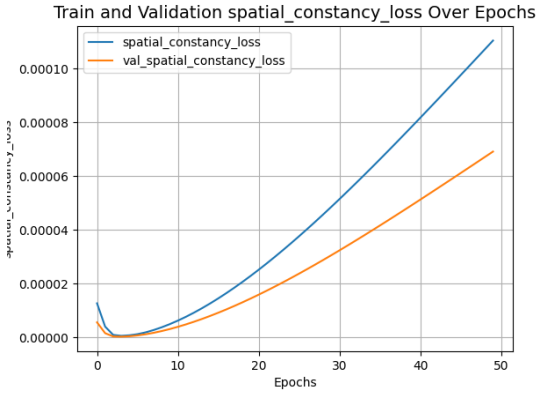
To apply changes:

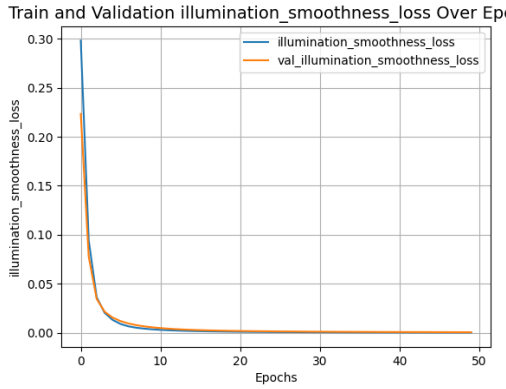
```
def build_dce_net():
    input_img = keras.Input(shape=[None, None, 3])
    conv1 = layers.Conv2D(
        16, (3, 3), strides=(1, 1), activation="relu", padding="same"
    )(input_img)
    conv2 = layers.Conv2D(
        16, (3, 3), strides=(1, 1), activation="relu", padding="same"
    )(conv1)
    conv3 = layers.Conv2D(
        16, (3, 3), strides=(1, 1), activation="relu", padding="same"
    )(conv2)
    conv4 = layers.Conv2D(
        16, (3, 3), strides=(1, 1), activation="relu", padding="same"
    )(conv3)
    x_r = layers.Conv2D(24, (3, 3), strides=(1, 1), activation="tanh", padding="same")(
        conv4
    )
    return keras.Model(inputs=input_img, outputs=x_r)
```

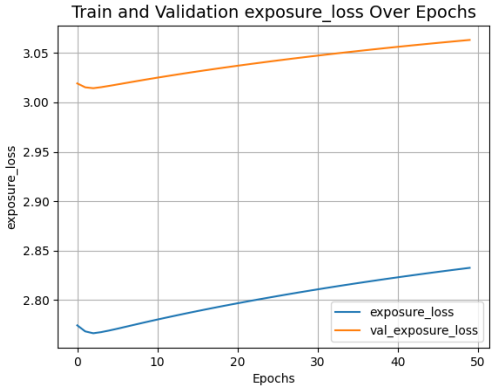
This modified code removes the concatenation operations int_con1 and int_con2 from the original code, resulting in a network without the symmetrical concatenations in the third and fourth convolutional layers.

Loss functions:









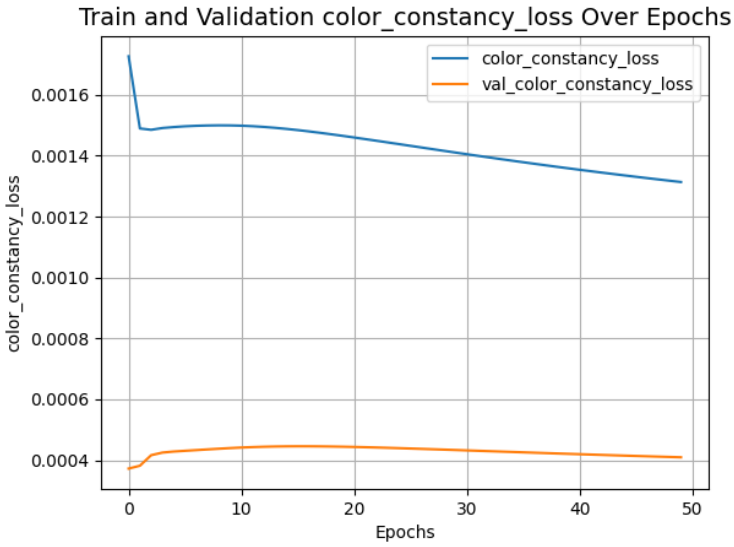
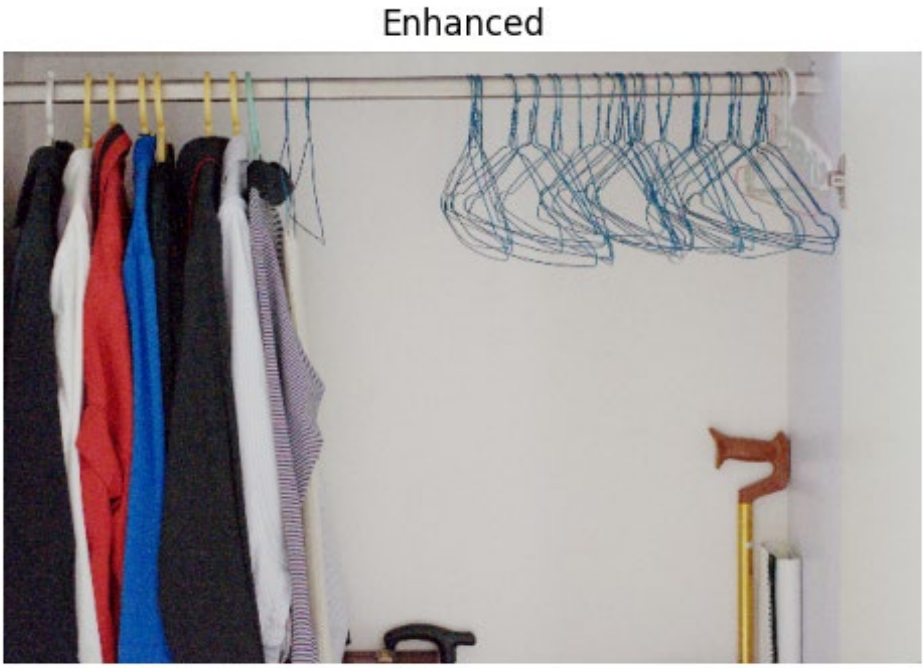
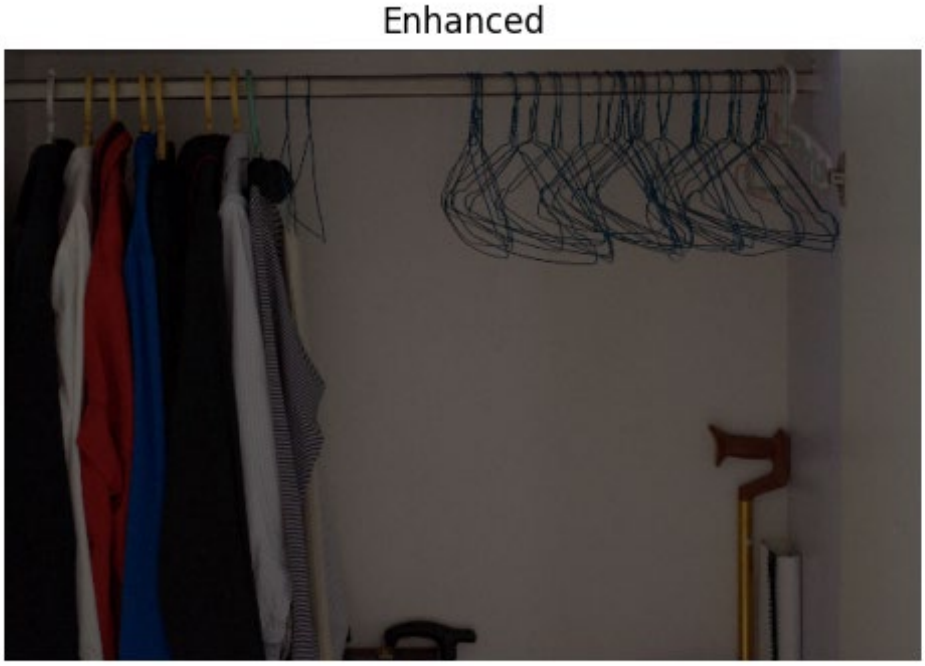


Image Results:



Original



New Implemented

Enhanced

Original

Enhanced

New Implemented

Metric Results:

Average PSNR_pil: 21.00531708707135

Average SSIM_pil: 0.7698392295014024

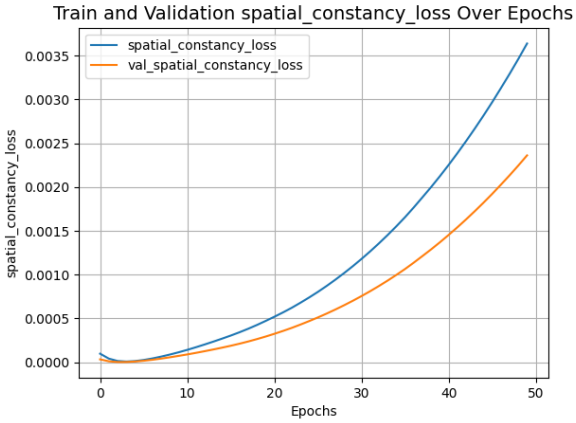
Decreasing the number of iterations:

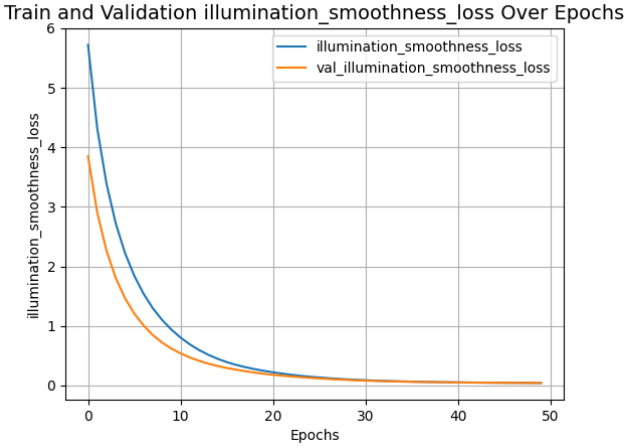
To apply changes:

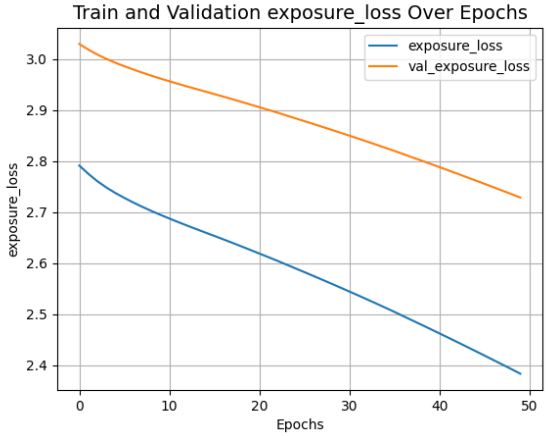
```
def get_enhanced_image(self, data, output):
    r1 = output[:, :, :, :3]
    r2 = output[:, :, :, 3:6]
    r3 = output[:, :, :, 6:9]
    r4 = output[:, :, :, 9:12]
    #r5 = output[:, :, :, 12:15]
    #r6 = output[:, :, :, 15:18]
    #r7 = output[:, :, :, 18:21]
    #r8 = output[:, :, :, 21:24]
    x = data + r1 * (tf.square(data) - data)
    x = x + r2 * (tf.square(x) - x)
    x = x + r3 * (tf.square(x) - x)
    enhanced_image = x + r4 * (tf.square(x) - x)
    #x = enhanced_image + r5 * (tf.square(enhanced_image) - enhanced_image)
    #x = x + r6 * (tf.square(x) - x)
    #x = x + r7 * (tf.square(x) - x)
    #enhanced_image = x + r8 * (tf.square(x) - x)
    return enhanced_image
```

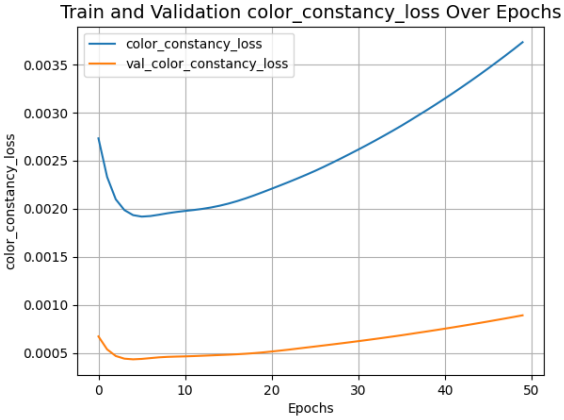
5, 6, 7, and 8 iterations are commented only 4 iterations used.

Loss functions:









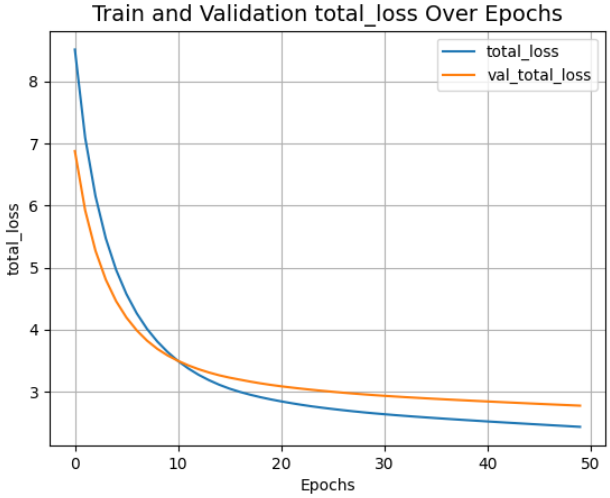
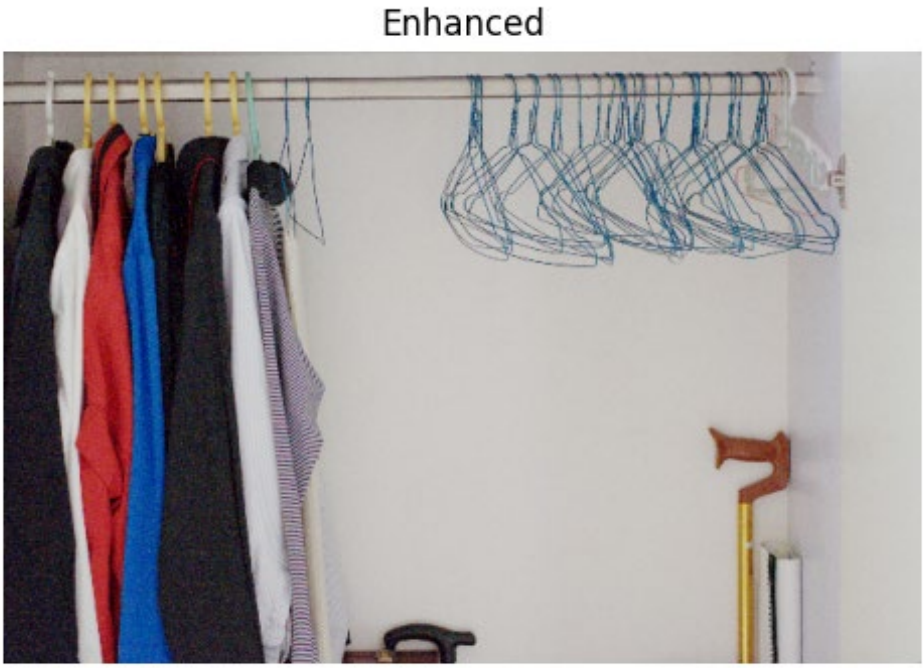
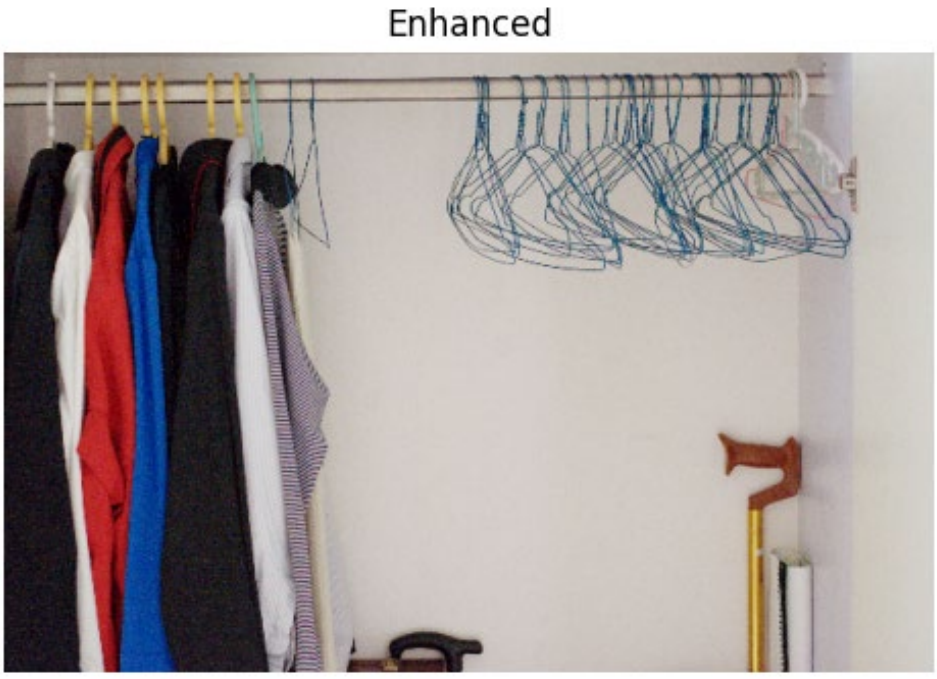


Image Results:



Original



New Implemented

Enhanced

Original

Enhanced

New Implemented

Metric Results:

Average PSNR_pil: 12.312752827974704

Average SSIM_pil: 0.5035847459255381


```
Average PSNR: 8.01575882919296
Average SSIM: 0.15092815296756176
Average PSNR_pil: 12.312752827974704
Average SSIM_pil: 0.5035847459255381
```

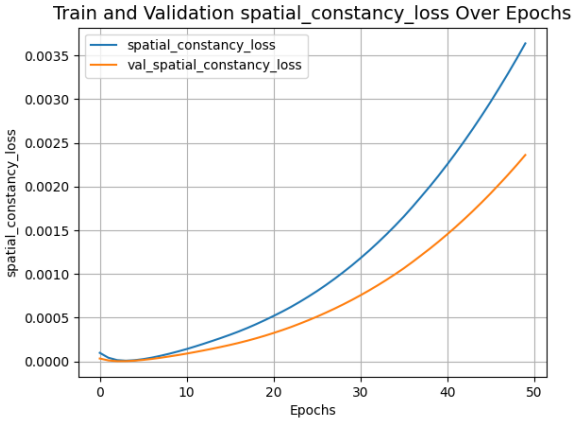
Increasing the number of convolutional kernels:

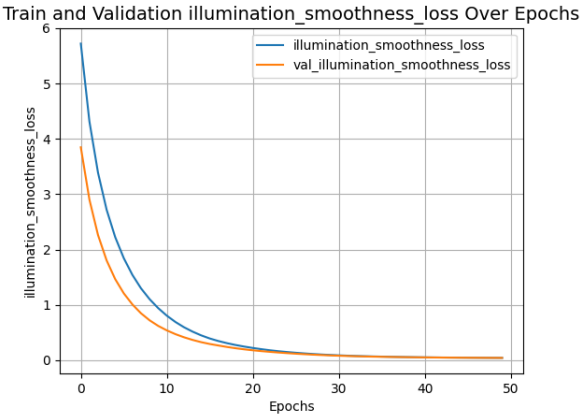
To apply changes:

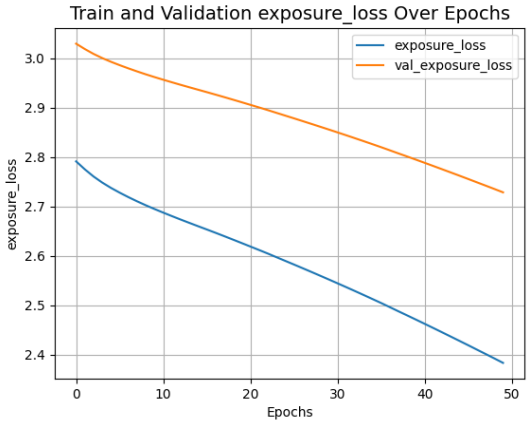
```python
def build_dce_net():
    input_img = keras.Input(shape=[None, None, 3])
    conv1 = layers.Conv2D(
        32, (3, 3), strides=(1, 1), activation="relu", padding="same"
    )(input_img)
    conv2 = layers.Conv2D(
        32, (3, 3), strides=(1, 1), activation="relu", padding="same"
    )(conv1)
    conv3 = layers.Conv2D(
        32, (3, 3), strides=(1, 1), activation="relu", padding="same"
    )(conv2)
    int_con1 = layers.Concatenate(axis=-1)([conv3, conv2])
    conv4 = layers.Conv2D(
        32, (3, 3), strides=(1, 1), activation="relu", padding="same"
    )(int_con1)
    int_con2 = layers.Concatenate(axis=-1)([conv4, conv1])
    x_r = layers.Conv2D(24, (3, 3), strides=(1, 1), activation="tanh", padding="same")(
        int_con2
    )
    return keras.Model(inputs=input_img, outputs=x_r)
```

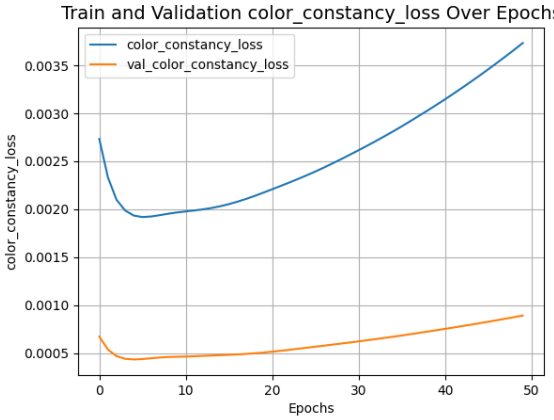
I just increase kernel 32 from the 16 for all layers.

Loss functions:









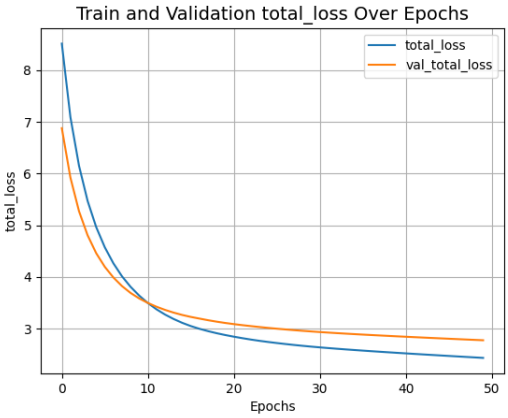
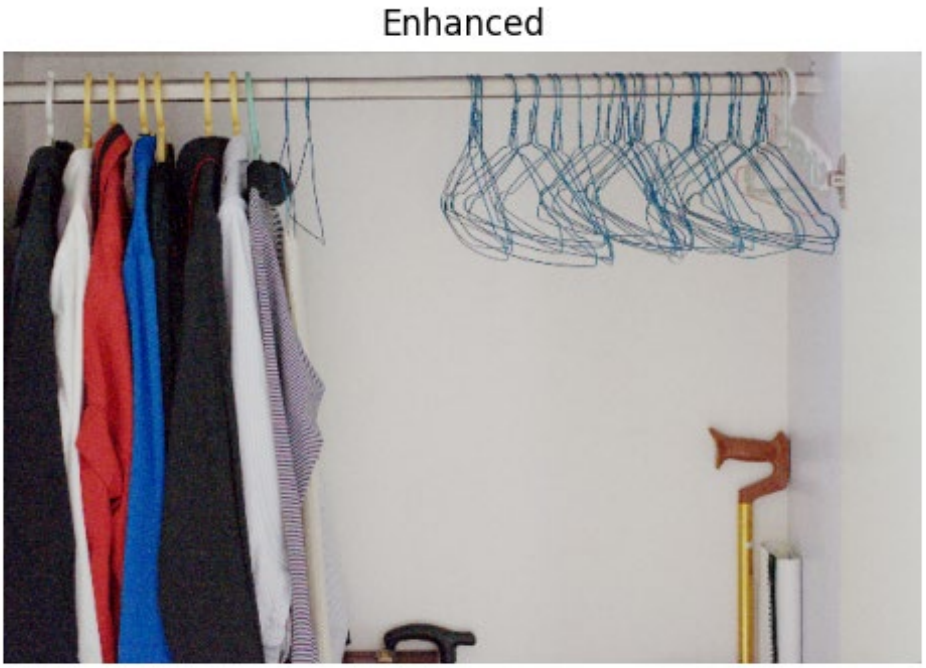
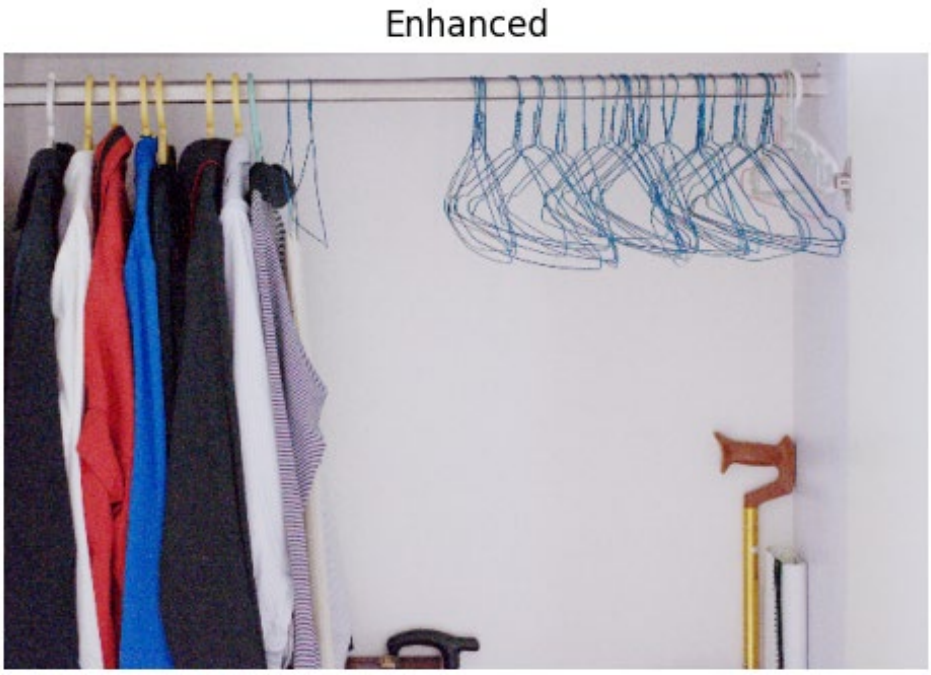


Image Results:



Original



New Implemented

Enhanced

Original

Enhanced

New Implemented

Metric Results:

Average PSNR_pil: 9.887353535400885

Average SSIM_pil: 0.4259114301765504

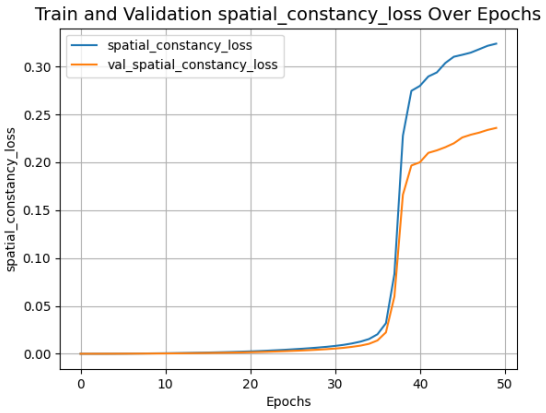
Increasing the number of convolutional layers:

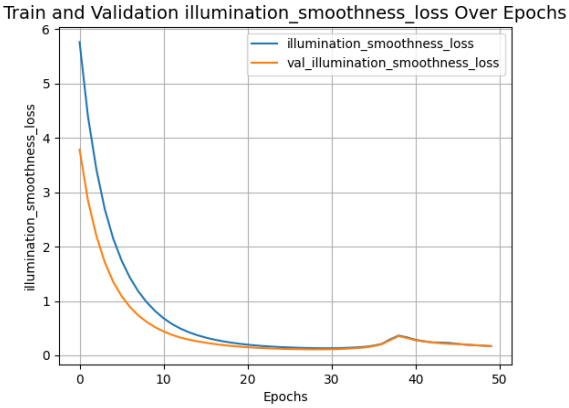
To apply changes:

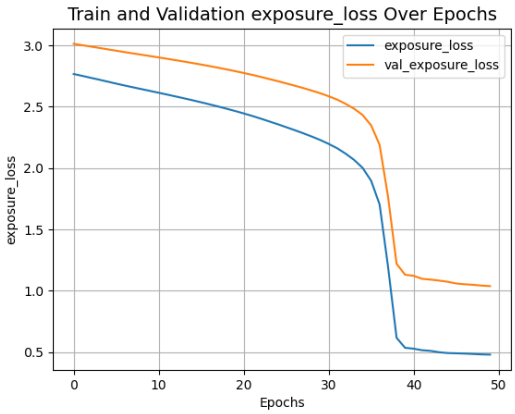
```
def build_dce_net():
    input_img = keras.Input(shape=[None, None, 3])
    conv1 = layers.Conv2D(
        16, (3, 3), strides=(1, 1), activation="relu", padding="same"
    )(input_img)
    conv2 = layers.Conv2D(
        16, (3, 3), strides=(1, 1), activation="relu", padding="same"
    )(conv1)
    conv3 = layers.Conv2D(
        16, (3, 3), strides=(1, 1), activation="relu", padding="same"
    )(conv2)
    conv4 = layers.Conv2D(
        16, (3, 3), strides=(1, 1), activation="relu", padding="same"
    )(conv3)
    int_con1 = layers.Concatenate(axis=-1)([conv4, conv3])
    conv5 = layers.Conv2D(
        16, (3, 3), strides=(1, 1), activation="relu", padding="same"
    )(int_con1)
    int_con2 = layers.Concatenate(axis=-1)([conv5, conv2])
    conv6 = layers.Conv2D(
        16, (3, 3), strides=(1, 1), activation="relu", padding="same"
    )(int_con2)
    int_con3 = layers.Concatenate(axis=-1)([conv6, conv1])
    x_r = layers.Conv2D(24, (3, 3), strides=(1, 1), activation="tanh", padding="same")(
        int_con3
    )
    return keras.Model(inputs=input_img, outputs=x_r)
```

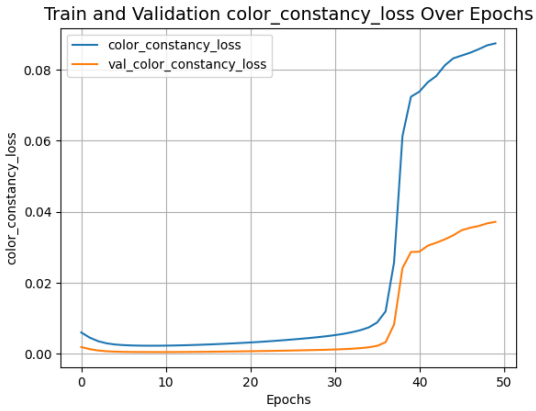
In PDF description the drawn schematics actually were 8 layers. So as similar to the this schematics I re-wrote the build function as shown in above and described in PDF. I followed the black arrows for skip connections.

Loss functions:









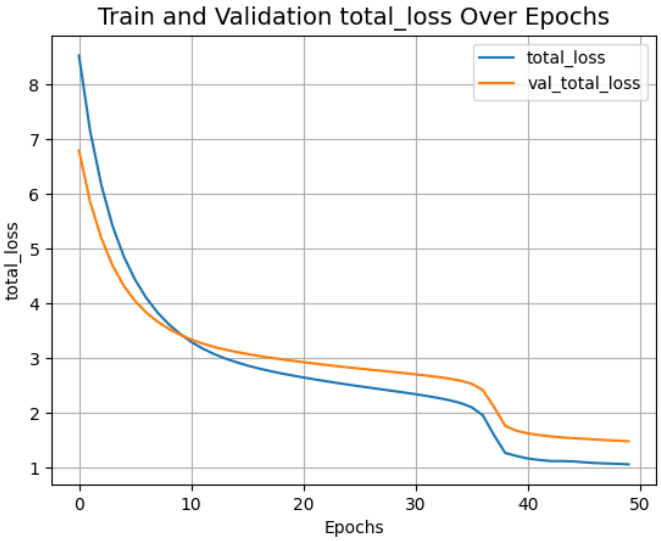
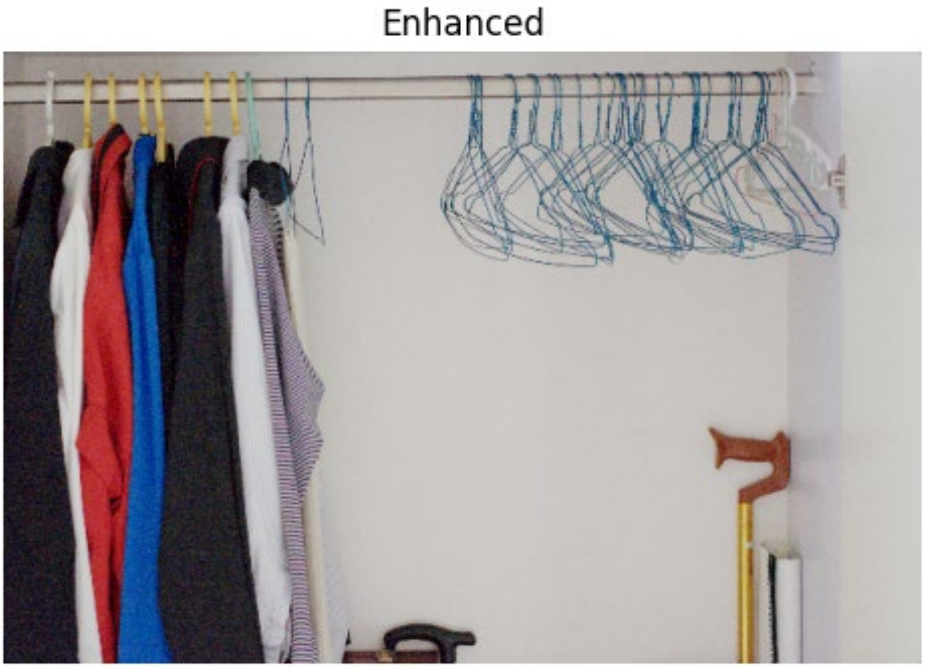


Image Results:



Original

New Implemented

Enhanced

Original

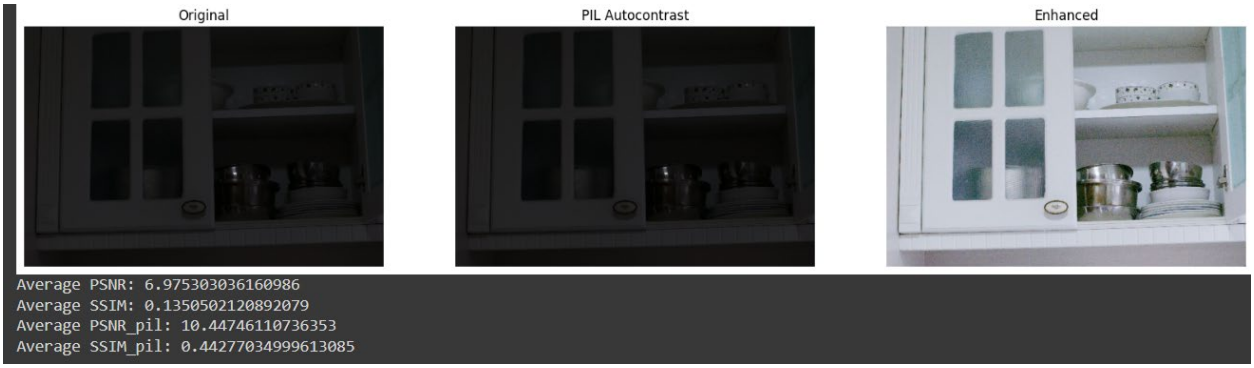
Enhanced

New Implemented

Metric Results:

Average PSNR_pil: 10.44746110736353

Average SSIM_pil: 0.44277034999613085



Part 3 Results Summary:

Qualitative Results:

Removal of Symmetrical Concatenations:

The elimination of symmetrical concatenations may prevent feature maps at various levels of the network from exchanging information with one another. Reduced spatial consistency and other quality issues could emerge from this adjustment. It was better image before the removing symmetrical concatenations.

Decrease in the Number of Iterations:

Using fewer iterations, can cause the insufficient time to converge and capture fine details in the enhanced images. The reduction in the number of iterations result in a less refined enhancement output. There were some noises in the images.

Increase in the Number of Convolutional Kernels:

The model may be better able to learn complicated picture features if there are more convolutional kernels. Sharper details and better overall image enhancement appear to result from this. However, for me the images were not better. It was overfit the detail. The key features were much more focused which was not naturel.

Increase in the Number of Convolutional Layers with Skip Connections:

Better information flow and gradient propagation in the network are made possible by the introduction of skip connections. By assisting the model in capturing both low-level and high-level characteristics, skip connections can increase the results of augmentation. As a result, the images become better.

Quantitative Results:

Removal of Symmetrical Concatenations:

The average PSNR and SSIM scores were bigger than the symmetrical concatenations. Actually, it should be less than due to loss of key features and increasing noise in image. However, may be over fitting or

import function related issues may lead this dilemma. Main reason can be referenced image since is not true image and it just the autocorrected image.

The total loss graph also changed. The total losses are decreased significantly in amount, and it give the very close values to the validation errors since the symmetrical concatenations are removed.

Decrease in the Number of Iterations:

Again, the average PSNR and SSIM scores were bigger than the 8 iterations. Actually, it should be less than due to loss of key features and increasing noise in image. However, may be over fitting or import function related issues may lead this dilemma. Main reason can be referenced image since is not true image and it just the autocorrected image.

The total loss graph also changed. The total losses are decreased but not too much, and the validation and train error crossed themselves in epoch 10, further that the train error is decreased much faster than the validation error.

The Decreasing the number of iterations gave worse result.

Increase in the Number of Convolutional Kernels:

Again, the average PSNR and SSIM should increase however it oppositely decreased as like above. May be over fitting or import function related issues may lead this dilemma. Also referenced image not as true image. Main reason can be referenced image since is not true image and it just the autocorrected image.

The changes in total loss graph this time was not significant, but this time even though the plots was very similar to the above model the total loss was better. After the epoch number increase the loss differences between train and validation increases, also they converge same loss in small epoch numbers.

It improves the image quality.

Increase in the Number of Convolutional Layers with Skip Connections:

Again, the average PSNR and SSIM have same dilemma I do not want to explain again. The main reason of this dilemma since the true images in calculations of PNSR and SSIM, are not actually perfect (true image) result. They are just referenced autocorrected images.

The total loss graphs this time was perfect graph. The train losses and validation losses were close to each other. In around 35 epoch we saw second sudden decreasing in the total loss, which is better.

As a result, it gave the better image.

Overall:

These qualitative and quantitative studies allow us to assess how each architectural modification affects the functionality of the DCE-Net* model. The comparisons to the original model will aid in determining the alterations that cause an increase or decrease in the quality of image enhancement. Adding skip connections and increasing the number of convolutional kernels give the better result. However, removing symmetrical connections and decreasing the number of iterations give not good results.