Restoration and enhancement of low light images

26.04.2024

—

Aritra Bhaduri
Indian Institute of Technology, Patna

Overview

The project aims to address the challenges associated with images captured under low light conditions. Low light images often suffer from noise, blurriness, and lack of detail, which can significantly degrade their quality and usability. The primary objective of this project is to develop and implement algorithms and techniques to restore and enhance low light images, thereby improving their visual quality and increasing their utility for various applications such as surveillance, photography, and medical imaging.

Acknowledgement

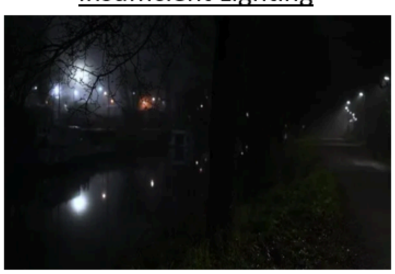
This project used the Retinex Net code from the github: https://github.com/weichen582/RetinexNet

This project also uses the LoL dataset available at kaggle: https://www.kaggle.com/datasets/soumikrakshit/lol-dataset

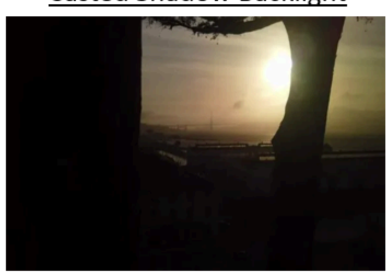
Problem Description

Low light images are mainly caused due to under exposure, wrong lighting conditions or camera defects that make the image difficult to see.

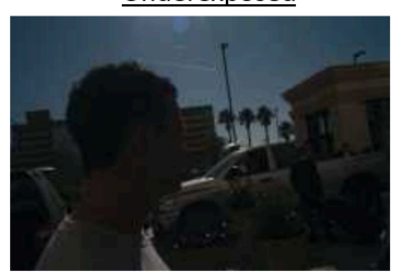
Insufficient Lighting



Casted Shadow Backlight



Underexposed



Solution approach

1. There are various classical and deep learning based methods for enhancing low light images.
2. Classical approaches include Histogram Equalization, gamma transformation, etc.
3. Possible deep learning approaches include GANs, Autoencoders, stable diffusion, Retinex models etc.

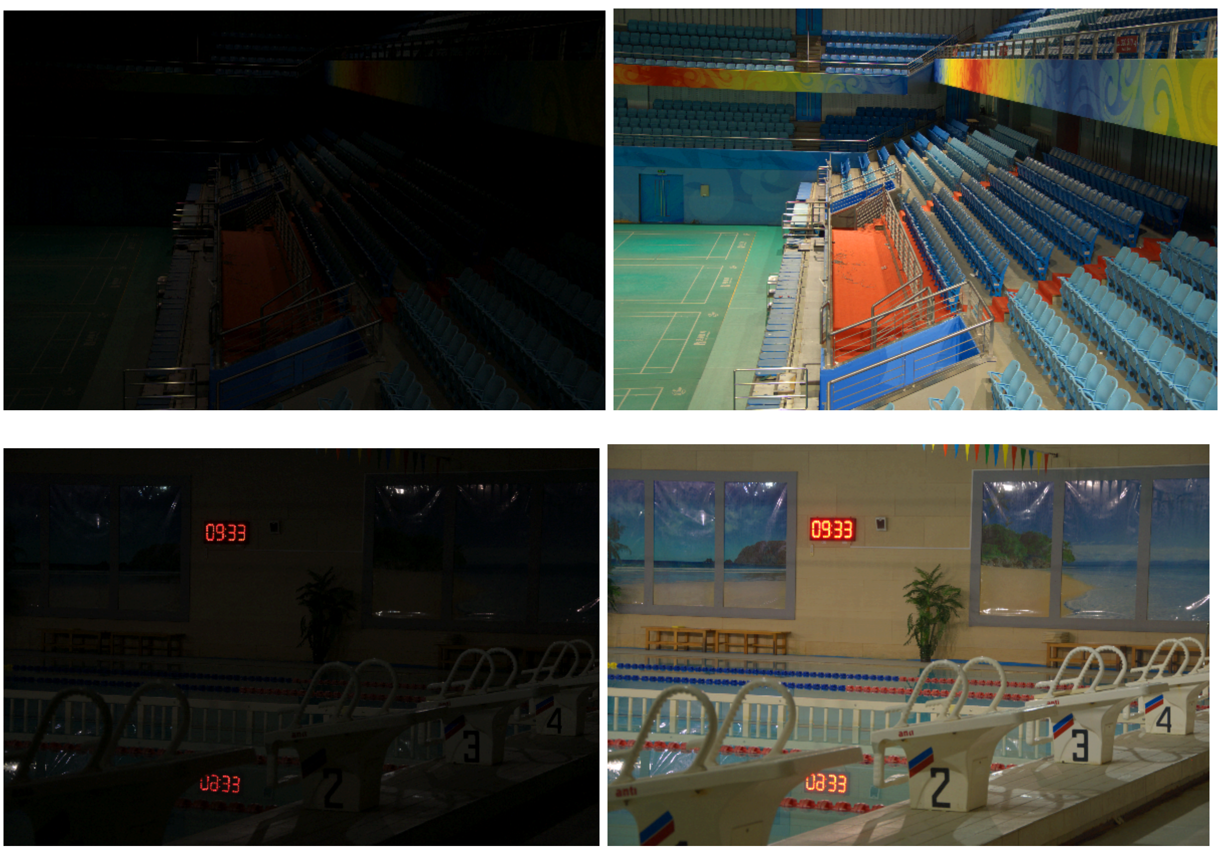
4. These are GPU intensive solutions, so we need to reduce image dimensionality or model parameters to make it run on platforms like Google Colab which has limited GPU capacity.

Dataset

After referring to various technical and academic literature, I have selected the LoL dataset (LOw-Light Dataset) for this task. It contains pairs of light and dark images of the same scene, and has a size of 485 images in the training set and 15 images for testing.

Dataset paper: Wei, Chen, Wenjing Wang, Wenhan Yang, and Jiaying Liu. "Deep retinex decomposition for low-light enhancement." arXiv preprint arXiv:1808.04560 (2018).

For this task, we have scaled down all the images to 256x256 dimensions for training and testing due to GPU constraints and CNN complexity in deep learning based models.



These are two samples from the dataset.

Classical Models

For classical models, I have tried Histogram equalization that gives remarkably good performance in spite of the fact that it is such a simple algorithm. It is very good at preserving image features, colors and in some cases even makes the image sharper than the original ground truth.

Here are a few samples of the histogram equalization output:

Deep Learning Based Models

For deep learning models, I have explored three model architectures, Baseline feed forward network, Autoencoder and RetinexNet.

Baseline

We use a very simple encoder-decoder architecture for our baseline model with following hyperparameters:

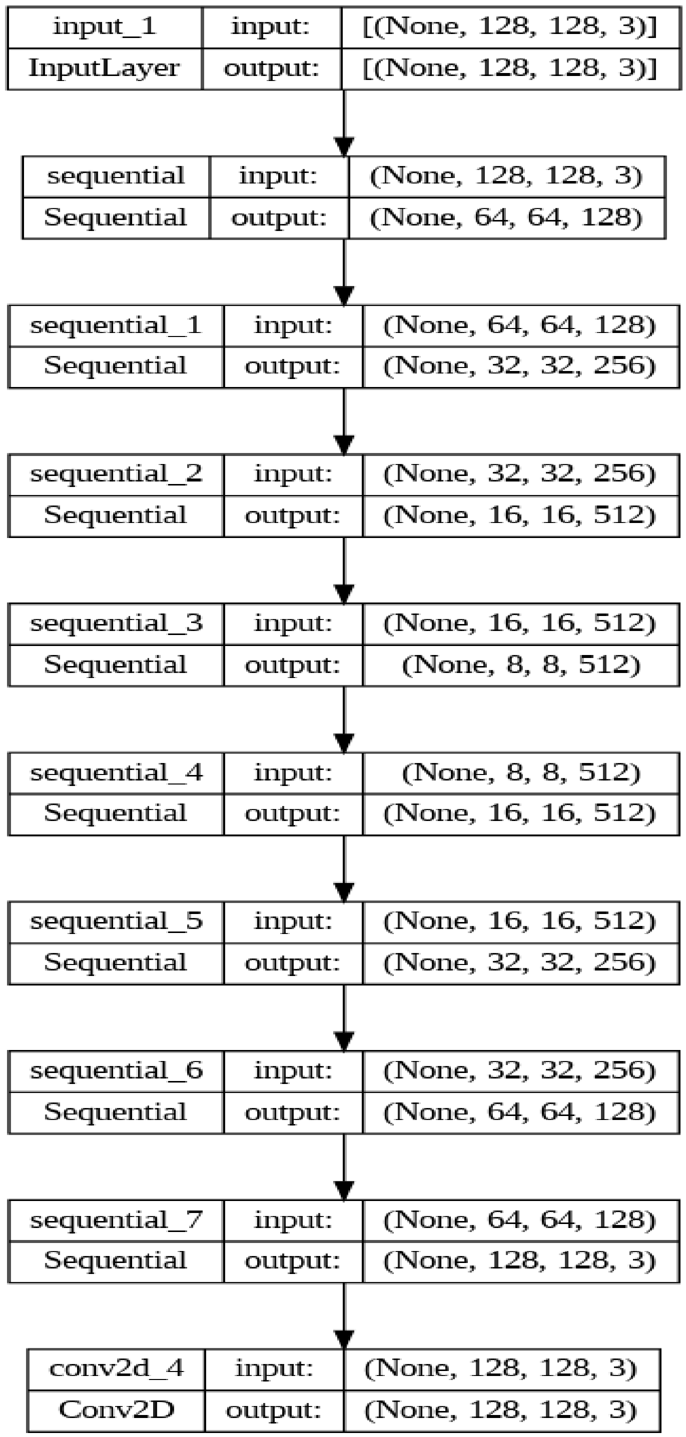
- Input: 128x128x3
- Output: 128x128x3
- Batch size: 32
- Dropout rate: 0.1
- Activation: LeakyReLU
- Learning Rate: 0.001
- Epochs: 80

Network Architecture::

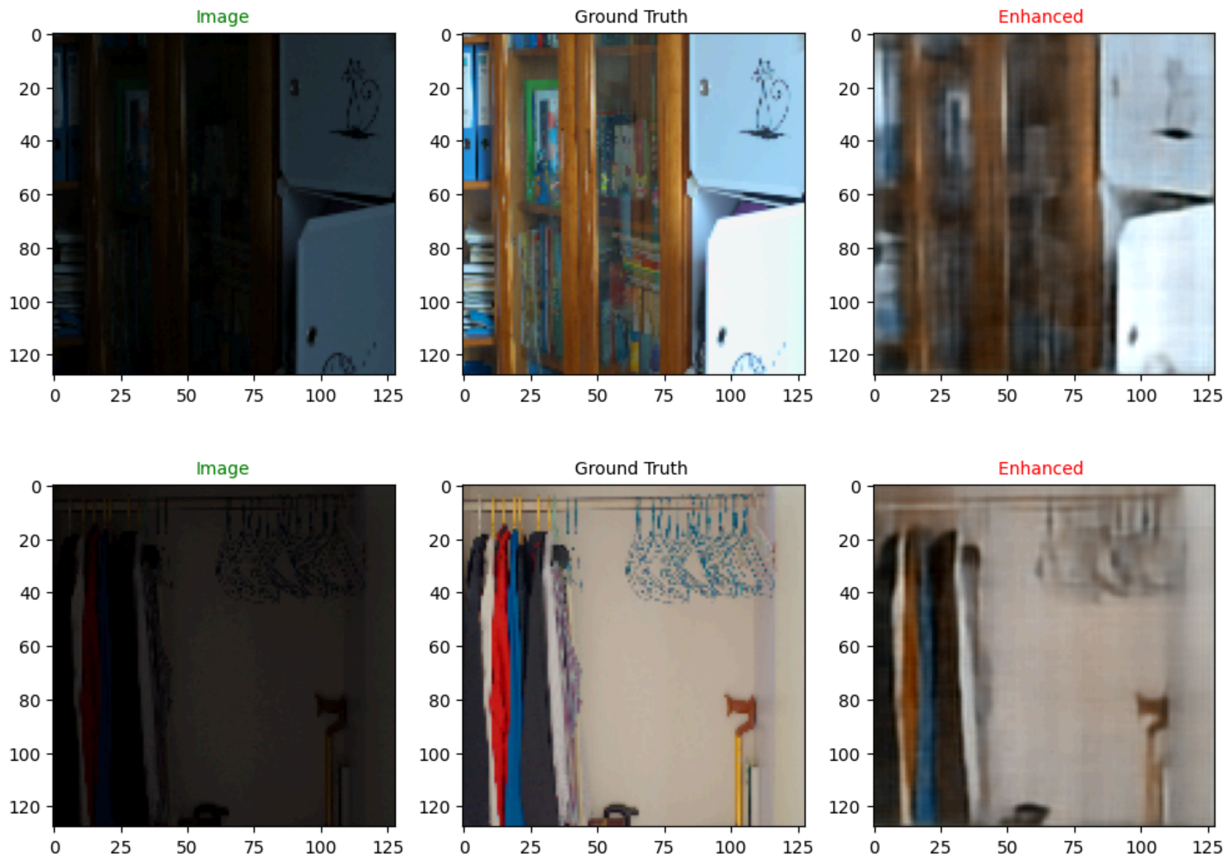
- 4 downsampling CNN layers

- 4 upsampling CNN layers

A diagrammatic view of the model is given:



Some preliminary results with the model:



As we can notice, it introduces a large amount of artifacts and noises in the image.

As this was a very simple baseline approach, we present a more refined version of the work in the next section.

Autoencoder Based Framework

We directly work upon the model to improve its performance by making some useful changes:

1. Increase the depth of the network
2. Add residual connections that preserve some features of the original image.

Residual deep CNN has been proposed to tackle the issue of accuracy degradation in deeper learning networks. Residual connection provides another path for data to reach latter parts of the neural network by skipping some layers.

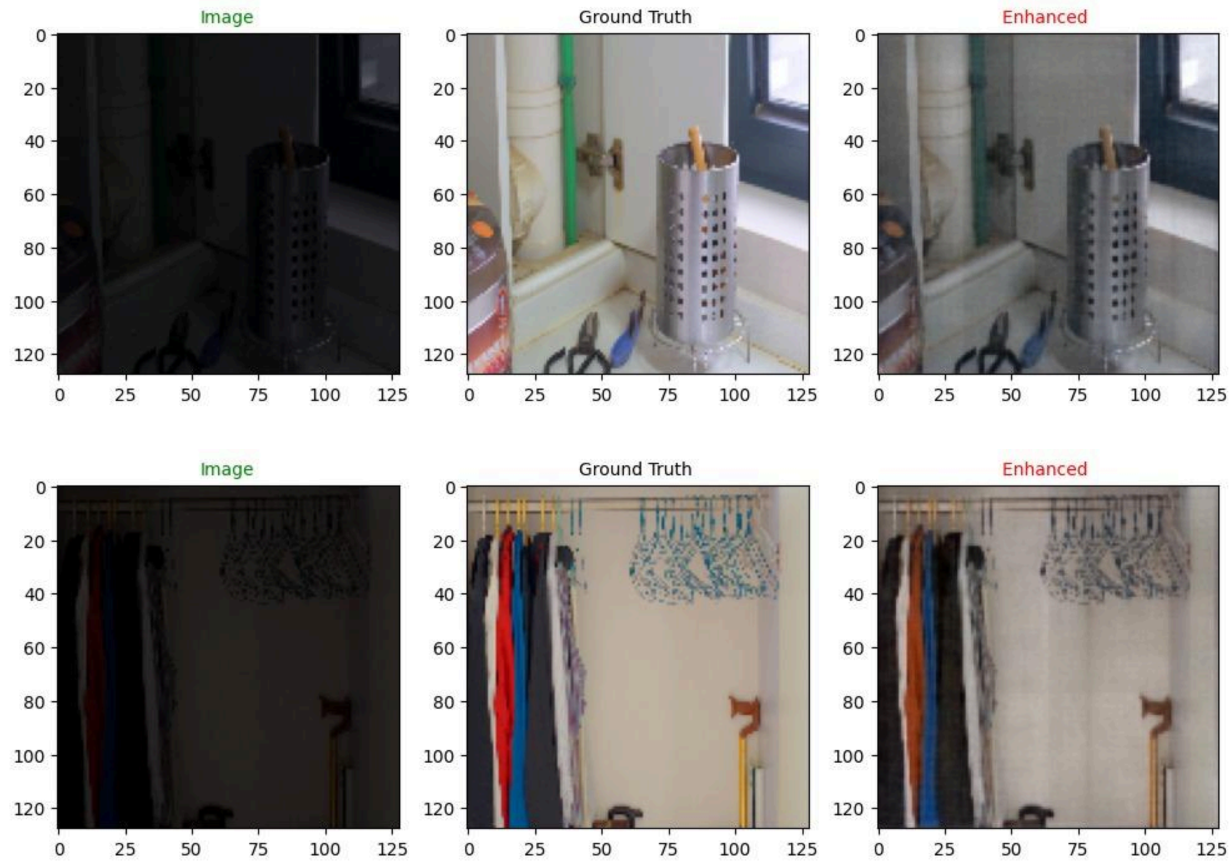
It helps to solve issues like vanishing and exploding gradients, also it acts as a mechanism to retain some of the information that the network lost during forward pass.

The model summary of the network is given:

```
Model: "model"
__________________________________________________________________________________
________
 Layer (type)                   Output Shape                Param #   Connected
to
==================================================================================
========
 input_1 (InputLayer)           [(None, 128, 128, 3)]       0         []


 sequential (Sequential)        (None, 64, 64, 128)         9728
['input_1[0][0]']


 sequential_1 (Sequential)      (None, 32, 32, 256)         819456
['sequential[0][0]']


 sequential_2 (Sequential)      (None, 16, 16, 512)         3279360
['sequential_1[0][0]']


 sequential_3 (Sequential)      (None, 8, 8, 512)           6556160
['sequential_2[0][0]']


 sequential_4 (Sequential)      (None, 16, 16, 512)         6554112
['sequential_3[0][0]']


 concatenate (Concatenate)      (None, 16, 16, 1024)        0
['sequential_4[0][0]',

'sequential_2[0][0]']


 sequential_5 (Sequential)      (None, 32, 32, 256)         6553856
['concatenate[0][0]']
```

```
 concatenate_1 (Concatenate  (None, 32, 32, 512)          0
['sequential_5[0][0]',
 )
'sequential_1[0][0]']


 sequential_6 (Sequential)   (None, 64, 64, 128)          1638528
['concatenate_1[0][0]']


 concatenate_2 (Concatenate  (None, 64, 64, 256)          0
['sequential_6[0][0]',
 )
'sequential[0][0]']


 sequential_7 (Sequential)   (None, 128, 128, 3)          19203
['concatenate_2[0][0]']


 concatenate_3 (Concatenate  (None, 128, 128, 6)          0
['sequential_7[0][0]',
 )
'input_1[0][0]']


 conv2d_4 (Conv2D)           (None, 128, 128, 3)          75
['concatenate_3[0][0]']


==============================================================================
========
Total params: 25430478 (97.01 MB)
Trainable params: 25428430 (97.00 MB)
Non-trainable params: 2048 (8.00 KB)
______________________________________________________________________________
________
```

Results



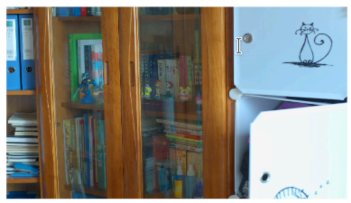
Deep Retinex Model

Retinex enhancement process is divided into three steps: decomposition, adjustment, and reconstruction.In the decomposition step, a subnetwork **Decom-Net** decomposes the input image into reflectance and illumination. In the following adjustment step, an encoder-decoder based **Enhance-Net** brightens up the illumination.Multi-scale concatenation is introduced to adjust the illumination from multi-scale perspectives. Noise on the reflectance is also removed at this step. Finally, we reconstruct the adjusted illumination and reflectance to get the enhanced result.

(a) Input (b) Reflectance (c) Illumination

Results

Original low light



Original bright image

Enhanced image

Original low light

Original bright image

Enhanced image

Overall Results

Autoencoder Model:

Image	**PSNR**	**SSIM**
669.png	22.30305367	0.802650425
547.png	19.71941722	0.766579178
1.png	18.07628825	0.75271031
748.png	21.43982215	0.785107417
55.png	12.93418623	0.708342468

778.png	19.6732167	0.646577932
23.png	12.62898359	0.689070625
111.png	19.41770793	0.770111472
146.png	21.57790401	0.776712799
493.png	14.91872852	0.505492156
22.png	18.84251086	0.805640638
780.png	16.8139392	0.648441552
79.png	21.31506967	0.83638852
179.png	13.36377956	0.582897155
665.png	14.00455015	0.647457023
Average	**17.80194385**	**0.714945311**

Histogram Equalization Model:

Image	**PSNR**	**SSIM**
669.png	15.56601536	0.373327344
547.png	14.19534514	0.361257589
1.png	19.00887611	0.575391343
748.png	11.88751554	0.345597165
55.png	14.26875753	0.379159644
778.png	15.01152037	0.473657978
23.png	15.05004999	0.323870576
111.png	15.73420381	0.398760918
146.png	15.40305669	0.383143653
493.png	12.31039939	0.415728634
22.png	19.49184363	0.517070929

780.png	10.79557584	0.245310347
79.png	15.16424205	0.444704639
179.png	9.967894155	0.314259437
665.png	14.26594976	0.247492057
Average	**14.54141636**	**0.38658215**

Retinex-Net Model:

Image	**PSNR**	**SSIM**
669.png	19.80725183	0.395585723
547.png	17.21920985	0.365576227
1.png	13.61057836	0.541331322
748.png	17.61572062	0.41994558
55.png	11.92696249	0.316121579
778.png	18.32997843	0.450181776
23.png	12.33726398	0.303322035
111.png	17.82139656	0.447517898
146.png	21.78441748	0.588773344
493.png	18.64941842	0.412756054
22.png	18.34396108	0.489884184
780.png	15.34684464	0.306410302
79.png	17.77992183	0.491856451
179.png	15.59363399	0.546740108
665.png	15.44380462	0.298948319
Average	**16.77402428**	**0.424996727**

Conclusion

We have done a comparative analysis across the deep learning and classical methods of low light image enhancement. Although we can see that deep learning models perform slightly below expectations when compared to the Histogram Equalization method, this can be attributed to less amount of training and using resized images due to hardware constraints. In cases where we have sufficient hardware available and we train the model to its full potential, the deep learning based models can perform on par and maybe even outperform the classical methods.