REPORT ON
IMAGE ENHANCEMENT USING ZERO-REFERENCE DEEP CURVE ESTIMATION

VAISANT K , 22117147, MECHANICAL

Introduction

Image enhancement is a fundamental task in image processing aimed at improving the visual quality of images for various applications such as medical imaging, surveillance, photography, and more. Traditional methods often rely on manual tuning or statistical techniques which may not always yield optimal results across different types of images and conditions. In recent years, deep learning approaches have shown promise in automating and improving the quality of image enhancement tasks. Zero Reference Deep Curve Estimation (DRCE) is a specific technique within this domain that focuses on enhancing images without requiring a reference image.

Zero Reference Deep Curve Estimation (DCE)

Zero Reference DRCE is an advanced method that utilizes deep learning to estimate a transformation curve for enhancing image contrast and brightness. Unlike traditional methods that rely on reference images for comparison, DRCE techniques learn to enhance images based solely on the input image itself. This approach is advantageous in scenarios where a high-quality reference image is not available or where real-time enhancement is required.

Framework

A Deep Curve Estimation Network (DCE-Net) is devised to estimate a set of best-fitting Light-Enhancement curves (LE-curves) given an input image. The framework then maps all pixels of the input's RGB channels by applying the curves iteratively for obtaining the final enhanced image.

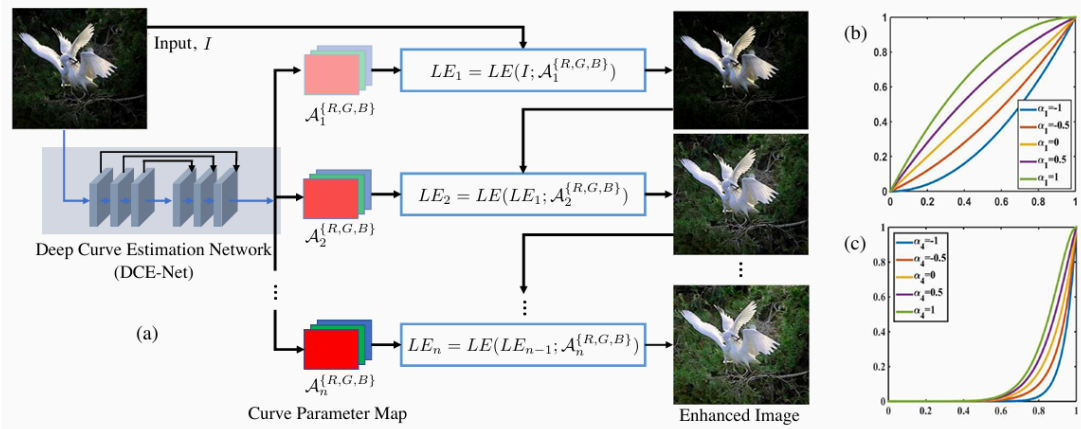


Figure 2: (a) The framework of Zero-DCE. A DCE-Net is devised to estimate a set of best-fitting Light-Enhancement curves (LE-curves) that iteratively enhance a given input image. (b, c) LE-curves with different adjustment parameters α and numbers of iteration n. In (c), α_1, α_2, and α_3 are equal to -1 while n is equal to 4. In each subfigure, the horizontal axis represents the input pixel values while the vertical axis represents the output pixel values.

Key Components of Zero Reference DRCE

The key components include LE-curve, DCE-Net, and non-reference loss functions which work together to make this method a success.

1.Light-Enhancement Curve (LE-curve)

These are the curves that can map a low-light image to its enhanced version automatically, where the self-adaptive curve parameters are solely dependent on the input image. That's why

we don’t require both high and low pairs for training, rather only low light images are enough for training. This can be expressed as a quadratic curve,

$$LE(I(\mathbf{x});\alpha) = I(\mathbf{x}) + \alpha I(\mathbf{x})(1 - I(\mathbf{x})),$$

Higher-Order Curve

The LE-curve defined above can be applied iteratively to enable more versatile adjustment to cope with challenging low-light conditions.

$$LE_n(\mathbf{x}) = LE_{n-1}(\mathbf{x}) + \alpha_n LE_{n-1}(\mathbf{x})(1 - LE_{n-1}(\mathbf{x})),$$

Pixel-Wise Curve

A higher-order curve can adjust an image within a wider dynamic range. Nonetheless, it is still a global adjustment since alpha is used for all pixels. A global mapping tends to over-/under- enhance local regions.

$$LE_n(\mathbf{x}) = LE_{n-1}(\mathbf{x}) + \mathcal{A}_n(\mathbf{x}) LE_{n-1}(\mathbf{x})(1 - LE_{n-1}(\mathbf{x})),$$

2.DCE-Net

Zero Reference DCE typically employs convolutional neural networks (CNNs) of seven convolutional layers with symmetrical concatenation. Each layer consists of 32 convolutional kernels of size 3x3 and stride 1 followed by the ReLU activation function.These networks are trained to estimate a non-linear transformation curve

that maps the input image to an enhanced version with improved contrast and brightness.

3.Non-Reference Loss Functions

During training, appropriate loss functions are employed to optimize the network parameters.

Spatial Consistency Loss

The spatial consistency loss encourages spatial coherence of the enhanced image through preserving the difference of neighboring regions between the input image and its enhanced version

$$L_{spa} = \frac{1}{K}\sum_{i=1}^{K}\sum_{j\in\Omega(i)}(|(Y_i - Y_j)| - |(I_i - I_j)|)^2,$$

Exposure Control Loss

To restrain under-/over-exposed regions, we design an exposure control loss Lexp to control the exposure level.

$$L_{exp} = \frac{1}{M}\sum\nolimits_{k=1}^{M}|Y_k - E|,$$

Color Constancy Loss

Following Gray-World color constancy hypothesis that color in each sensor channel averages to gray over the entire image, we design a color constancy loss to correct the potential color deviations in the enhanced image and also build the relations among the three adjusted channels.

$$L_{col} = \sum\nolimits_{\forall(p,q)\in\varepsilon}(J^p - J^q)^2, \varepsilon = \{(R,G),(R,B),(G,B)\},$$

Illumination Smoothness Loss

To preserve the monotonicity relations between neighboring pixels, we add an illumination smoothness loss to each curve parameter map A.

$$L_{tv_{\mathcal{A}}} = \frac{1}{N}\sum_{n=1}^{N}\sum_{c\in\xi}(|\nabla_x \mathcal{A}_n^c| + \nabla_y \mathcal{A}_n^c|)^2, \xi = \{R, G, B\},$$

Total Loss

The total loss can be expressed as

$$L_{total} = L_{spa} + L_{exp} + W_{col}L_{col} + W_{tv_{\mathcal{A}}}L_{tv_{\mathcal{A}}},$$

Implementation

The implementation of Zero reference DCE from theory to code involved several steps. The overall model consists of 6 files which are main.py, model.py, Model_Train.ipynb, predictor.py and image_utils.py and training dataset. LoL dataset has been used for the training where 400 images for training and 86 images for validation. And for testing 15 images were used.

Tools used

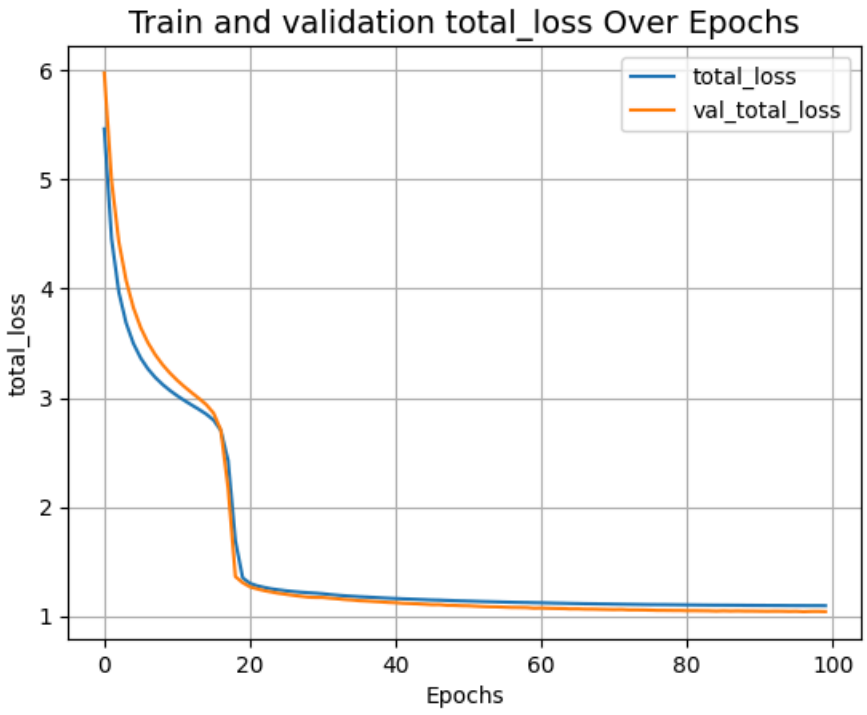
The libraries and modules used were tensorflow, keras, glob, numpy, matplotlib, PIL, math and cv2.

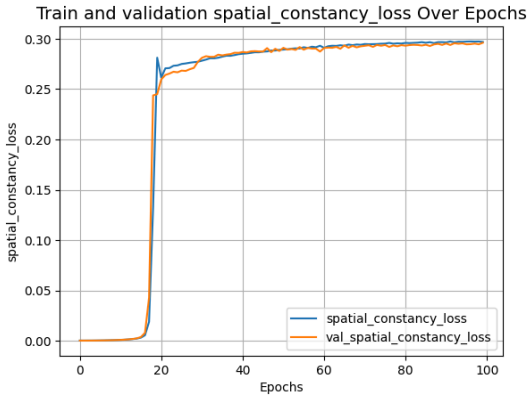
model.py

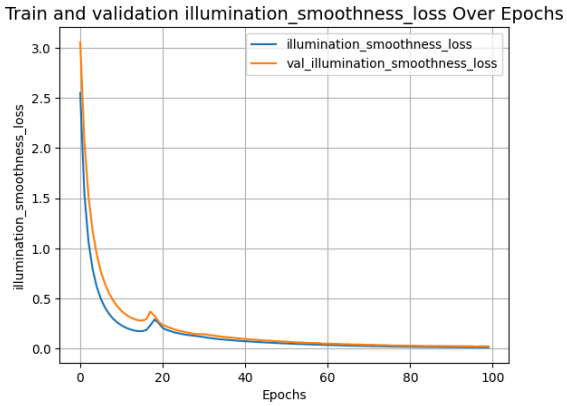
- A CNN of DCE Net was built from scratch.
- Then different losses were defined in code as different functions.
- Then a class ZeroDCE was defined which contained the compilation function, metrics for different losses, get_enhanced_image function to work on the pixels for enhancement, compute_loss function to compute the total loss, a train_step and test-step functions for forward and backward propagation in accordance with the losses and save_weights and load_weights functions.

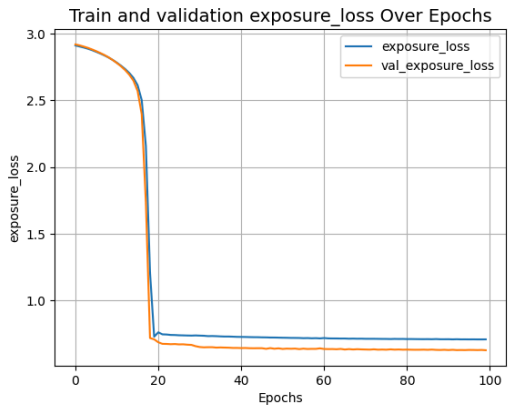
Mode_Train.ipynb

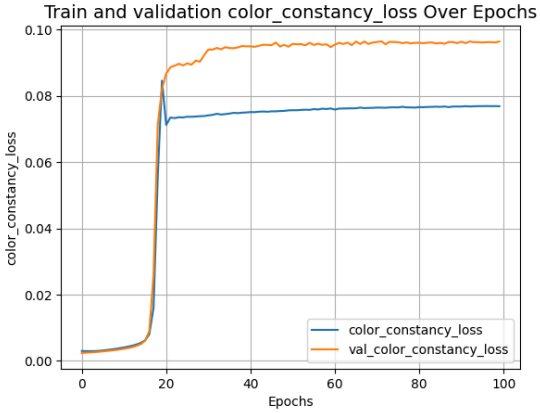
- As I was facing issues in importing the model, I defined the model code in this file and instantiated ZeroDCE class.
- Then compilation was done with a learning rate of 1e-4 and training started.
- After 15 hours the training was over and it generated the results with different losses.
- Results











- After training, the weights and model have been saved.

predictor.py

- A predictor class has been defined with 3 functions.
- The plot_results function can be used to view the enhanced image.
- The infer function is the function which does the job of giving the enhanced images from the input images with the help of the model with the saved weights.
- The from_path function does the job of loading the model with the saved weights “wt.h5”.

image_utils.py

- The image_utils class takes help of the predictor class.
- The image_utils class has enhance_images function which uses infer function from predictor to enhance the images and saves it in the predicted folder.
- The AVG_MSE_PSNR function calculates the average mse and psnr for the images enhanced with high light original images from the dataset.

main.py

- The test images have been loaded.
- From image_utils enhanced_images function has been used to enhance the images from test images and saved in the predicted folder.
- Then the high light original images and predicted images have been loaded.
- Then AVG_MSE_PSNR function has been used to calculate the MSE and PSNR for the model.
- Results

```
PS C:\Users\Vaisant K\OneDrive\Desktop\image denoising\imgd> python -u
2024-06-18 17:38:31.922688: I tensorflow/core/platform/cpu_feature_gua
-critical operations.
To enable the following instructions: SSE SSE2 SSE3 SSE4.1 SSE4.2 AVX
<keras.src.engine.functional.Functional object at 0x000002253D948E50>
{'zerodce': <model.ZeroDCE object at 0x0000022502965040>}
MSE: 101.81513564814817 PSNR 28.07976056684471
PS C:\Users\Vaisant K\OneDrive\Desktop\image denoising\imgd>
```

Conclusion

The Zero Reference Deep Curve Estimation model has been implemented and produced good results with **MSE 101.81 and PSNR of 28.07 dB**. By anchoring predictions to a zero reference, DCE models reduce errors and enhance the precision of curve estimations.The zero reference point provides context for predictions, making it easier to interpret and validate model outputs in practical applications.

References

https://arxiv.org/abs/2001.06826

https://keras.io/examples/vision/zero_dce/

https://youtu.be/9Rbi3f0Vg7k?si=l3vRHnGiHK_ZNnx0

https://youtu.be/yF4iJx9Ap2E?si=ktZpCmSNi9lP9kNo

https://youtu.be/PGOWjMCMB8I?si=IJdCVGTlVLRLA8VU

NOTE : Before running the main.py file make sure you open the directory in the explorer of VSC(which i used) and then run the main.py file. Otherwise i am getting this error

FileNotFoundError: [Errno 2] Unable to synchronously open file (unable to open file: name = 'wt.h5', errno = 2, error message = 'No such file or directory', flags = 0, o_flags = 0)