BBM 444 – Programming Assignment 4: Deep Low-light Image Enhancement

Due date: Tuesday, 26-4-2022, 11:59 PM.

Overview

The purpose of this assignment is to explore the use of deep learning approaches for enhancing low-light images[1]. In general, capturing images in low-light conditions is a challenging task. As we discussed in class, the level of the signal measured by the camera sensors is generally much lower than the noise in the measurements. Noise present in a low-light image also affects various image characteristics such as fine-scale structures and color balance, further degrading the image quality.

Direct approaches for capturing bright photos in low light conditions include widening the aperture of the camera lens, lengthening the exposure time, or using camera flash. These methods, however, do not solve the problem completely as each of these hacks has its own drawbacks. Opening the aperture is limited by the hardware constraints. When the camera flash is used, the objects closer to the camera are brightened more than the objects or the scene elements that are far away. Images captured with long exposure times might have unwanted image blur due to camera shake or object movements in the scene.

Instead, we can attempt to produce a high-quality image from a given low-light image by using a deep model which is specifically designed and trained for this task. In this assignment, you are going to use and analyze a simplied version of the Zero-Reference Deep Curve Estimation (Zero-DCE) model [1]. In Figure 1, a sample low-light input image and its enhanced version obtained by Zero-DCE model are given.

Figure 1: A low-light input image and its enhanced version obtained by Zero-DCE model.

In particular, Zero-DCE formulates low-light image enhancement as the task of estimating an image-specific tonal curve with a deep neural network. The corresponding deep model estimates pixel-wise and high-order tonal curves for dynamic range adjustment of a given image. In short, it takes a low-light image as input and produces high-order tonal curves as its output. These curves are then used for pixel-wise adjustment on the dynamic range of the input to obtain an enhanced image. The curve estimation process is done in such a way that it maintains the range of the enhanced image and preserves the contrast of neighboring pixels. This curve estimation is inspired by curves adjustment used in photo editing software such as Adobe Photoshop where users can adjust points throughout an image's tonal range.

Zero-DCE is appealing because of its relaxed assumptions with regard to reference images: it does not require any input/output image pairs during training. This is achieved through a set of carefully formulated non-reference loss functions, which implicitly measure the enhancement quality and guide the training of the network.

[1] Adapted from the Hugging Face demo on Zero-DCE for low-light image enhancement prepared by Soumik Rakshit.

Setting Up

We recommend that you use Colab (`https://colab.research.google.com/`) for the assignment, as all the assignment notebooks have been tested on Colab. Otherwise, if you are working on your own environment, you will need to install Python 2, PyTorch (`https://pytorch.org`), iPython Notebooks, SciPy, NumPy and scikit-learn. Check out the websites of the course and relevant packages for more details.

From the assignment zip file, you will find the python notebook file named `zero_dce.ipynb`. To setup the Colab environment, you will need to upload the notebook file using the upload tab at `https://colab.research.google.com/`.

1. Enhancement via DCE-Net* (20 points)

Experimental analysis is carried out on the LoL Dataset which has been created for low-light image enhancement. It provides 485 images for training and 15 for testing. Each image pair in the dataset consists of a low-light input image and its corresponding well-exposed reference image. You will use 300 low-light images from the LoL Dataset training set for training, and use the remaining 185 low-light images for validation. The images are resized to size 128×128 to be used for both training and validation. Note that in order to train the DCE-Net, you will not require the corresponding enhanced images. They are provided only for reference.

The goal of DCE-Net is 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 to obtain the final enhanced image.

A light-enhancement curve is a kind of curve 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. When designing such a curve, three objectives should be taken into account:

- Each pixel value of the enhanced image should be in the normalized range [0,1], in order to avoid information loss induced by overflow truncation.
- It should be monotonous, to preserve the contrast between neighboring pixels.
- The shape of this curve should be as simple as possible, and the curve should be differentiable to allow backpropagation.

The light-enhancement curve is separately applied to three RGB channels instead of solely on the illumination channel. The three-channel adjustment can better preserve the inherent color and reduce the risk of over-saturation.

In this assignment, you will use DCE-Net*, a simplified version of the original DCE-Net model, to learn a mapping between an input image and its best-fitting curve parameter maps. The input to the DCE-Net* is a low-light image while the outputs are a set of pixel-wise curve parameter maps for corresponding higher-order curves. It is a plain CNN of five convolutional layers with symmetrical concatenation. Each layer consists of 16 convolutional kernels of size 3×3 and stride 1 followed by the ReLU activation function. The last convolutional layer is followed by the tanh activation function, which produces 24 parameter maps for 8 iterations, where each iteration requires three curve parameter maps for the three channels. Figure 2 demonstrates this neural architecture of the DCE-Net* model.

To enable zero-reference learning in DCE-Net and evaluate the quality of enhanced images, the following set of differentiable zero-reference losses are utilized:

- *Color constancy loss* is used to correct the potential color deviations in the enhanced image.

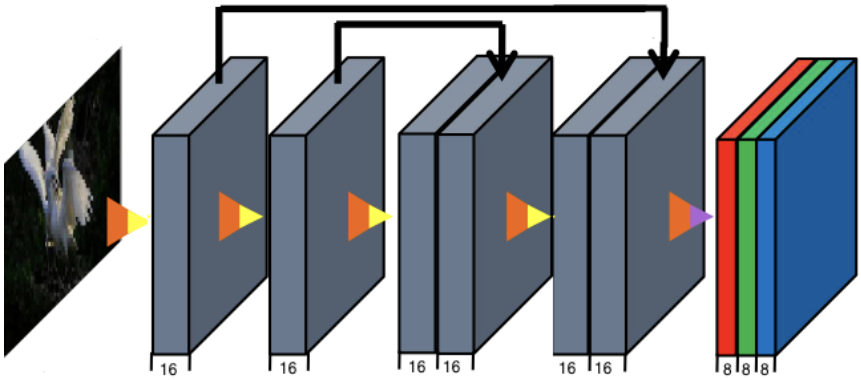

Figure 2: Neural architecture of the DCE-Net* model.

- *Exposure control loss* is employed to restrain under-/over-exposed regions. It measures the distance between the average intensity value of a local region and a preset well-exposedness level (set to 0.6).
- *Illumination smoothness loss* is added to each curve parameter map to preserve the monotonicity relations between neighboring pixels,
- *Spatial consistency loss* is used to encourage spatial coherence of the enhanced image by preserving the contrast between neighboring regions across the input image and its enhanced version.

In the first part of this assignment, just run the notebook `zero_dce.ipynb` provided in the assignment ZIP file without any change. As you will see, the notebook includes training and testing routines. Observe how the loss values are changed throughout the epochs and how the DCE-Net* model enhances some sample images from the test set. Does increasing the number of epochs from 10 to 50 improve the results? You should report the loss plots and the images enhanced by DCE-Net*, and discuss the results you obtained. Moreover, for quantitative analysis, you should estimate and report average PSNR and SSIM scores considering the reference enhanced images for the provided test images.

2. Loss Functions (40 points)

In the second part of the assignment, you will investigate the contribution each loss functions to the enhancement performance by performing an ablation study. As mentioned earlier, the total loss function used during training includes four individual loss functions defined in *Deep curve estimation model* section of the notebook:

```
total_loss = (
            loss_illumination
            + loss_spatial_constancy
            + loss_color_constancy
            + loss_exposure
            )
```

In your analysis, you will exclude each one of these loss functions from the total loss function, and accordingly train four separate models for at least 50 epochs. Observe how excluding each individual loss function affects the enhancement results. You should report the images enhanced

by the trained models, and compare these results with each other and that of the model trained with the original total loss function you considered in Part 1 in both qualitative and quantitative manner.

3. Neural Architecture (40 points)

In the last part of the assignment, you will play with the neural architecture of DCE-Net* model and analyze the effects of each change you made to the model. In particular,

- Remove the symmetrical concatenations considered in the third and fourth convolutional layers.
- Decrease the number of iterations from 8 to 4.
- Increase the number of convolutional kernels from 16 to 32.
- Increase the number of convolutional layers from 5 to 7 by introducing skip connections between first and sixth, second and fifth, and third and fourth layers, respectively.

In your analysis, you will train four separate models including the aforementioned changes for at least 50 epochs. Observe how each model performs on the test images. You should report the images enhanced by the trained models, and compare these results against each other and that of the model considered in Part 1 again trained for 50 epochs in both qualitative and quantitative manner.

What to Hand In

Your submitted solution should include the following:

- A PDF report explaining what you did for each problem, including answers to all questions asked throughout Problems 1, 2, and 3. The report should include any figures and intermediate results that you think may help. Make sure to include explanations of any issues that you may have run into that prevented you from fully solving the assignment, as this will help us determine partial credit.
- The codes you will submit should be well commented and in the form of a Jupyter notebook.
- Image files for Problems 1-3, showing the various image enhancement results you are asked to generate.

You should prepare a ZIP file named `name-surname(s)-assgn4.zip` containing your solution / implementation `assgn4.ipynb` and your report `assgn4-report.pdf`, and submit it via email to `erkut@cs.hacettepe.edu.tr`.

Late policy

You may use up to five *extension* days (in total) over the course of the semester for the programming assignments. Late submission will not be allowed.

Academic Integrity

All work on assignments must be done individually unless stated otherwise. You are encouraged to discuss with your other classmates about the given assignments, but these discussions should be carried out in an abstract way. That is, discussions related to a particular solution to a specific problem (either in actual code or in the pseudocode) will not be tolerated. In short, turning in someone else's work, in whole or in part, as your own will be considered as a violation of academic integrity. Please note that the former condition also holds for the material found on the web as everything on the web has been written by someone else.

References

[1] Chunle Guo, Chongyi Li, Jichang Guo, Chen Change Loy, Junhui Hou, Sam Kwong, and Runmin Cong. Zero-Reference Deep Curve Estimation for Low-Light Image Enhancement. In *Proceedings of IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2020.