Low-Light Denoising

Ding Li
New York University
dl4222@nyu.edu

Mengqi Fan
New York University
mf3971@nyu.edu

Abstract

As the technology of photography developed, people are not only satisfied with taking photos under a bright condition, for example, they want to explore and record the world at night as well. However the quality of a photo is usually not enough if people try to take photos under a dark condition. Although we have some physical methods to increase the light coming into cameras, like increasing the exposure time, it sometimes comes with negative side effects such as blur. We introduce a little bit faster post-processing pipeline by using group convolution to process low-light images to enhance the quality. And we have some experiments to compare our approach with the state-of-the-art method.

Introduction

When a digital camera is taking a photo, the light with heat will fall on the CMOS, heat causes noise. In a low-light environment, some photographers Increase the ISO to make the image brighter, but higher ISO will also increase the sensitivity of the CMOS, thus amplifying the noise as well. Some people try to increase the exposure time, but images often get blurred if without using a tripod. Change to another aperture will also change the depth of field of the photo. Those physical approaches all have their own negative side effects.

Researchers create a variety of methods to process images, such as denosing[1], deblurring[2] and brightening up algorithms[3]. Nevertheless, they do not consider an extreme low-light condition. Among those researches, by using a fully convolutional network(FCN), Chen[4] introduced a post-processing pipeline to enhance the quality and denoise. It achieves a good performance not only on the quality of processed images, but also on the processing speed.

In this paper, according to Chen's[4] method, learning to see in the dark(SID), by modifying its network framework, we implement a new version pipeline with a faster processing speed by group convolution[9] and almost the same quality. The result has been proved by some experiments showing later in this paper.

Figure 1 shows the data information. To control variables, we use the same dataset as Chen's[5]. Those photos were taken by a Sony camera in an extreme low-light environment. By keeping other conditions almost the same, using a tripod to stay stable, each image is taken with a short exposure time(1/10s, 1/25s and 1/30s) as training data, and long exposure time (100, 250 and 300 times longer) as reference data. Because of short exposure time, the training image is discernible and full of noise. Some short exposure images are from burst mode, they can also be used to train.

Sony α7S II	Filter array	Exposure time (s)	# images
x300	Bayer	1/10, 1/30	1190
x250	Bayer	1/25	699
x100	Bayer	1/10	808

Figure 1. Image data from a sony camera from SID.

Method

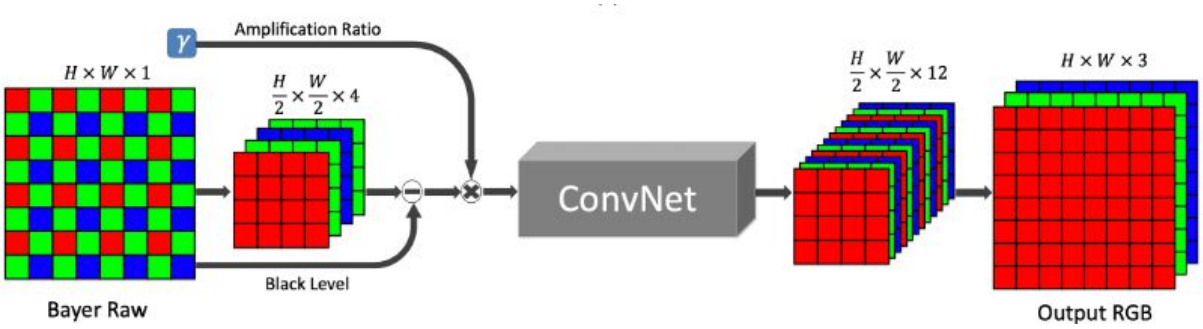


Figure 2. Pipeline structure of SID.

Here is what SID does in figure 2. First pack the input image of bayer array format into four channels and correspondingly reduce the spatial resolution by a factor of two in each dimension. Then subtract the black level and scale the data by the desired amplification ration, e.g. x100, x250, x300. Then put the current step result to a FCN to extract features. The output will be the same size but 12 channels. Next step, use a sub-pixel layer to recover the output to original resolution. At the training phase, they compute the L1 Loss of the output RGB image and the long exposure image(ground truth), which is substracing pixel by pixel and take absolute value.At the inference phase, the “output RGB image” is the model applied to the raw sensor data output result.

In figure 2, the “ConvNet” is a U-Net[6][7] structure of FCN, which gets down-sampling by applying deconvolution to get feature maps at different levels. Then, upsample the feature map with the smallest size by using deconvolution, and concatenate with the previous feature map to get feature maps with the same size as the input. Finally, use a subpixel layer to return current feature maps to the original size. Here is a framework table as follows:

Input Size	Conv Layer	In_Channels	Out_Channels	Output Name
H/2 x W/2	2 * Conv2D(kernel 3*3)+LRelu	4	32	x1
H/2 x W/2	Maxpool2d(kernel 2*2)	32	32	x1_downsample

H/4 x W/4	2 * Conv2D(kernel 3*3)+LRelu	32	64	x2
H/4 x W/4	Maxpool2d(kernel 2*2)	64	64	x2_downsample
H/8 x W/8	2 * Conv2D(kernel 3*3)+LRelu	64	128	x3
H/8 x W/8	Maxpool2d(kernel 2*2)	128	128	x3_downsample
H/16 x W/16	2 * Conv2D(kernel 3*3)+LRelu	128	256	x4
H/16 x W/16	Maxpool2d(kernel 2*2)	256	256	x4_downsample
H/32 x W/32	2 * Conv2D(kernel 3*3)+LRelu	256	512	x5
H/32 x W/32	ConvTranspose2d(2, 2)	512	256	x5_upsample
H/16 x W/16	Concate(x5_upsample, x4)	256+256	512	x5_concat
H/16 x W/16	2 * Conv2D(kernel 3*3)+LRelu	512	256	x4_upsample
H/16 x W/16	ConvTranspose2d(2, 2)	256	128	x4_upsample
H/8 x W/8	Concate(x4_upsample, x3)	128+128	256	x4_concat
H/8 x W/8	2 * Conv2D(kernel 3*3)+LRelu	256	128	x3_upsample
H/8 x W/8	ConvTranspose2d(2, 2)	128	64	x3_upsample
...	…	…	…	…
H/2 x W/2	Concate(x2_upsample, x1)	32+32	64	x1_concat
H/2 x W/2	2 * Conv2D(kernel 3*3)+LRelu	64	32	x1_upsample
H/2 x W/2	1 * Conv2D(kernel 1*1)	32	12	output
H/2 x W/2	Pixel shuffle	12	3	Output(H×W)

Here is what we modified. We first reimplement the pipeline according to the method discussed in the paper, then we realized the training speed and forward propagation speed of the original's pipeline is slow and it is required to run 4000 epochs. Thus we improve the structure by group convolution to accelerate the speed, and reimplement several new versions:

Structure 1: In order to decrease the parameters, change all the "2*Conv2D(kernel 3*3)+LReLU" to "1*Conv2D(kernel 1*1)+LReLU + 1*Depthwise Conv2D(kernel 3*3)+LReLU + 1*Conv2D(kernel 1*1)".

Structure 2: In order to further decrease the parameters, change all the "ConvTranspose2d" to "UpsamplingBilinear2d" and use Conv2D(kernel 1*1) to reduce channel dimensions.

Structure 3: In order to further decrease the parameters, change all the "2*Conv2D(kernel 3*3)+LReLU to 1*Conv2D(kernel 1*1)+LReLU + 1*Depthwise Conv2D(kernel 3*3)+LReLU, and change all the "ConvTranspose2d" to "UpsamplingBilinear2d + Conv2D(kernel 1*1)".

Experiments

We will use 5 groups to do the 3 experiments.

Group 1: original pipeline from (SID) with 4000 epochs and 0.0001 learning rate.

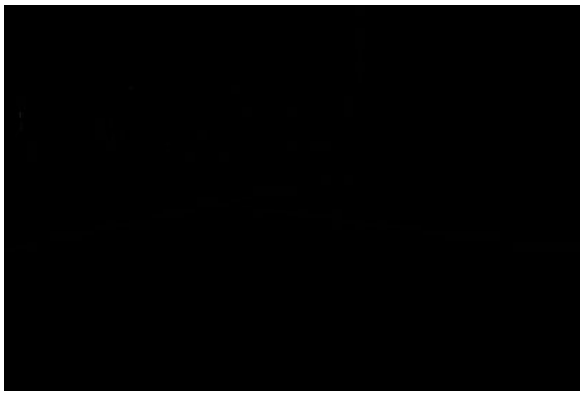
Group 2: structure 1 with 4000 epochs and 0.0001 learning rate.

Group 3: structure 2 with 4000 epochs and 0.0001 learning rate.

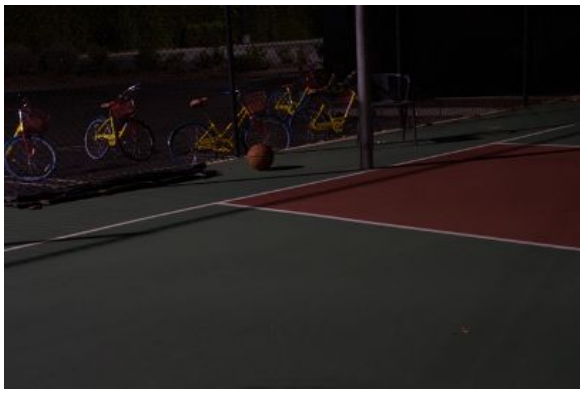
Group 4: structure 3 with 4000 epochs and 0.0001 learning rate.

Group 5: structure 3 with 1000 epochs and 0.0001 learning rate.

Experiment 1: In order to know the quality of processed images, we choose an image (ID 10006) with x100 exposure time to visualize and compare it with other groups. Scale version simply uses origin*(long exposure / origin) to enhance the brightness.

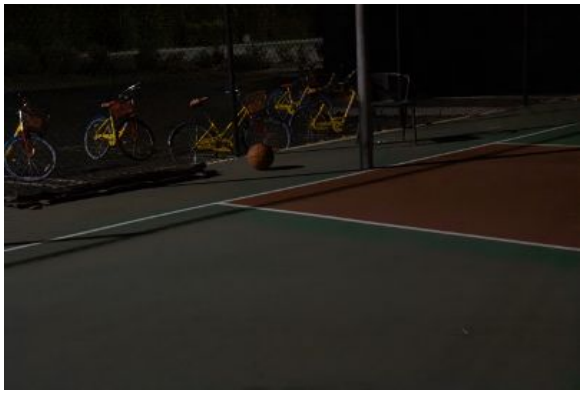


Origin Raw



Long Exposure

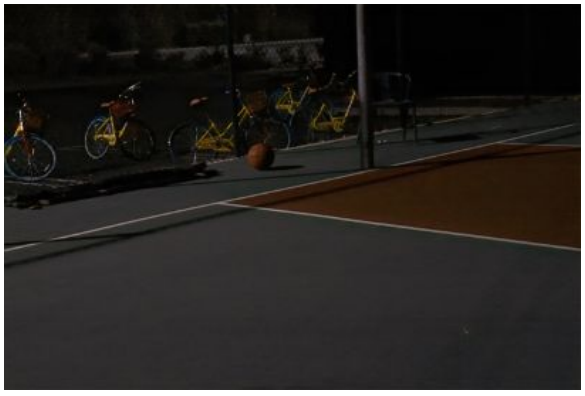
Scale



Group 1

Group 2

Group 3



Group 4

Group 5

Discussion 1: From those processed images, we can claim that group 1, 2, 3 and 4 look almost the same, but group 5 loses some color, since group 5 only trained for 1000 epochs. So we discard group 5.

Experiment 2: In order to know the accurate processed quality of this image, we apply PSNR and SSIM algorithms to examine. We analyze the PSNR and SSIM[8] of group 1(original method from SID) and group 4(our best method in theory).

	SID(Group 1)	Our model(Group 4)
PSNR	28.81	27.74
SSIM	0.781	0.752

Discussion: According to the PSNR and the SSIM, SID method's quality is 3.7% better than our method.

Experiment3: In order to know how much better our approach is, we compare the inference of our approach(group 4) with SID approach(group 1). And we compare the model parameter size as well. Since our GPU(1080ti) cannot process a whole image at once, we plan to run 512x512 patch 1000times for inference time to compute an average time.

	SID(Group 1)	Our model(Group 4)
Inference time	0.01534s per patch	0.01454s per patch
Model parameters	31.1MB	2.2MB

Discussion: Our model is around 5.2% faster than SID for inference time. And our model parameter size is 13 times smaller than SID.

Conclusion

The quality of our approach is a little bit worse than SID's approach, but almost the same. However our approach's inference(training) time and model parameters size is better than SID's approach. We successfully reimplement and modify the SID approach.

Reference

[1] T. Remez, O. Litany, R. Giryes, and A. M. Bron- stein. Deep convolutional denoising of low-light images. *arXiv:1701.01687*, 2017.

[2] Z. Hu, S. Cho, J. Wang, and M.-H. Yang. Deblurring low- light images with light streaks. In *CVPR*, 2014.

[3] X. Zhang, P. Shen, L. Luo, L. Zhang, and J. Song. Enhance- ment and noise reduction of very low light level images. In *ICPR*, 2012.

[4] Chen, Chen, et al. "Learning to See in the Dark." *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2018, doi:10.1109/cvpr.2018.00347.

[5] cchen156. "cchen156/Learning-to-See-in-the-Dark." *GitHub*, github.com/cchen156/Learning-to-See-in-the-Dark/tree/master/dataset.

[6] "U-Net." *Wikipedia*, Wikimedia Foundation, 13 Dec. 2020, en.wikipedia.org/wiki/U-Net.

[7] Ronneberger, Olaf; Fischer, Philipp; Brox, Thomas (2015). "U-Net: Convolutional Networks for Biomedical Image Segmentation". arXiv:1505.04597

[8] Z. Wang, A. C. Bovik, H. R. Sheikh, and E. P. Simoncelli. Image quality assessment: From error visibility to structural similarity. *IEEE Transactions on Image Processing*, 13(4), 2004.

[9] Bai, Kunlun. “A Comprehensive Introduction to Different Types of Convolutions in Deep Learning.” *Medium*, Towards Data Science, 11 Feb. 2019, towardsdatascience.com/a-comprehensive-introduction-to-different-types-of-convolutions-in-deep-learning-669281e58215.