Learning To See In The Dark

Chen Chen, Qifeng Chen, Jia Xu, and Vladlen Koltun,
"Learning to See in the Dark", in CVPR, 2018.
Implementation by Ajinkya Ambatwar(EE16B104)

December 9, 2019

1 Introduction

Imaging in low light is challenging due to low photon count and low SNR. Short-exposure images suffer from noise, while long exposure can induce blur and is often impractical. A variety of denoising, deblurring, and enhancement techniques have been proposed, but their effectiveness is limited in extreme conditions, such as video-rate imaging at night. To support the development of learning based pipelines for low-light image processing, the paper introduces a dataset of raw short-exposure low-light images, with corresponding long-exposure reference images. Using the presented dataset, developed is a pipeline for processing low-light images, based on end-to-end training of a fully- convolutional network. In this report, the original paper titled **Learning To See In The Dark[1]** is implemented. The original implementation is available in Tensorflow 1.8 while here we have tried to implement the same in PyTorch. The dataset we have used is "See In the Dark"(SID)(Sony Set) dataset provided by the authors of the paper. The dataset size is 72GB in size. More details of the dataset is provided below in the report. The paper reorts some promising results and that were observed during training as well as testing this network by us.

2 Dataset details

The SID dataset(Sony set) contains 232 training images with varying exposures of 0.1s, 0.4s, 0.04s and 0.33s. The input also consists of images with varying burst length of 0 to 9. With this the length of dataset is 2697 images. The ground truth of respective images is the same image captured with an exposure of 10s and higher ISO.

The dataset contains both indoor and outdoor images. The outdoor images were generally captured at night, under moonlight or street lighting. The illuminance at the camera in the outdoor scenes is generally between 0.2 lux and 5 lux. The indoor images are even darker. They were captured in closed rooms with regular lights turned off and with faint in direct illumination set up for

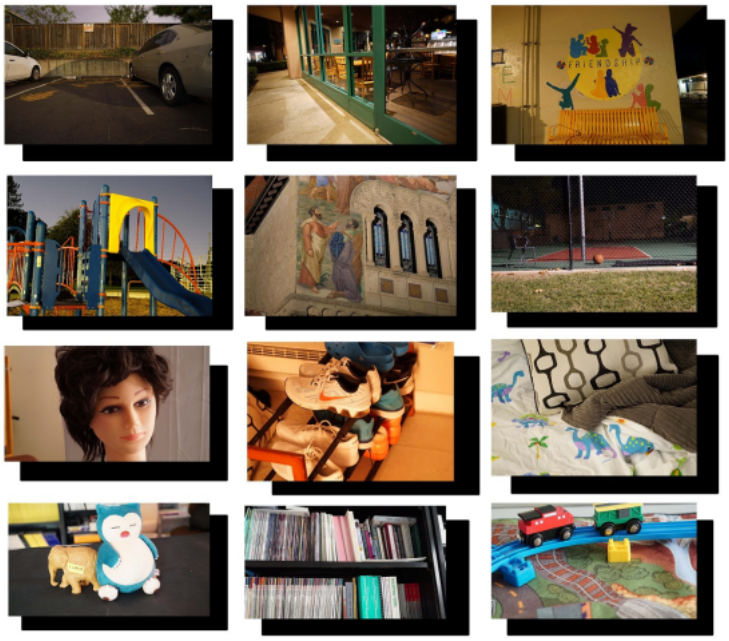

Figure 1: Example images in the SID dataset. Outdoor images in the top two rows, indoor images in the bottom rows. Long- exposure reference (ground truth) images are shown in front. Short-exposure input images (essentially black) are shown in the back. The illuminance at the camera is generally between 0.2 and 5 lux outdoors and between 0.03 and 0.3 lux indoors.

this purpose. The illuminance at the camera in the indoor scenes is generally between 0.03 lux and 0.3 lux.

The exposure for the input images was set between 1/30 and 1/10 seconds. The corresponding reference (ground truth) images were captured with 100 to 300 times longer exposure: i.e., 10 to 30 seconds. Since exposure times for the reference images are necessarily long, all the scenes in the dataset are static. The dataset is summarized in Table 1. A small sample of reference images is shown in Figure 1. Approximately 20% of the images in each condition are randomly selected to form the test set, and another 10% are selected for the validation set.

Each sample of image(both input and ground truth is a RAW file generated from the sensor of **Sony α7SII camera.** It possesses a full-frame Bayer sensor. This supports the evaluation of low-light image processing pipelines on images produces by different filter arrays. The resolution is 4240×2832.

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

Table 1: The SID(Sony) dataset contains 2697 images with varying exposure and burst length

3 Training Pipeline

The default network used to train is UNet[2]. The network architecure is shown below. It is a fully convolutional network with downconv and then upconv layers. Similar to ResNet, this network hosts a cross connection like branches that add up information from the previous layer while upconv.

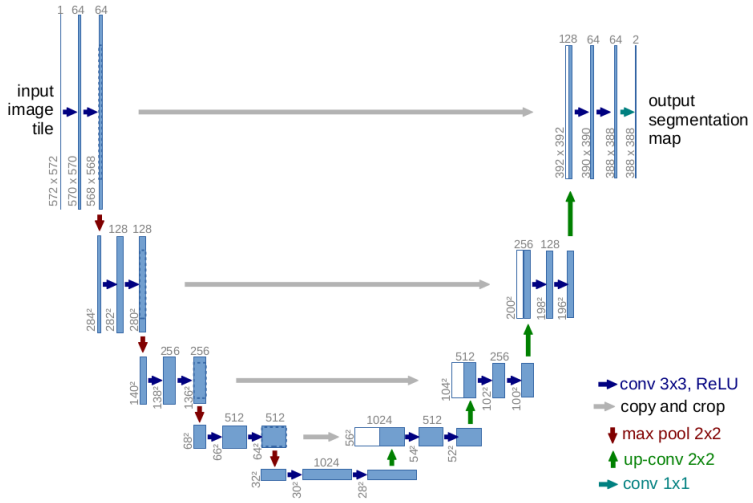

Figure 2: UNet architecture hosts a down conv as well as upconv. There are cross connections coming from previous layers.

The network is trained end to end unlike the traditional pipeline which involves manually hand created white balance, demosaic, Denoise, Deblur filters etc. Rather than operating on normal sRGB images produced by traditional camera processing pipelines, the network operates on raw sensor data. Figure 3 llustrates the structure of the presented pipeline. For Bayer array filter(for Sony camera), the input raw data(2D array) is packed into 4 channels and correspondingly reduce the spatial resolution by a factor of 2 in each dimension. The mean black levels are subtracted and the data is scaled by desired amplification ratio(e.g x100 or x300). The packed and amplified data is fed into a fully-convolutional network. The output is a 12-channel image with half the

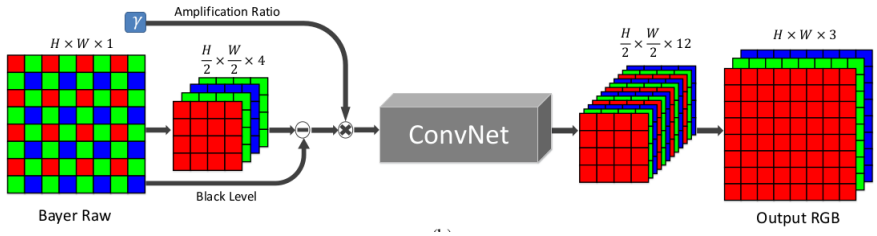

Figure 3: The training pipeline involves taking the input raw data and packing it into 4 channels and then is amplified and is sent as the input to the UNet network. The output is a 12 channel output with half the dimension. Using a pixel level **depthToSpace** layer the 12 channel output is turned into a sRGB image.

spatial resolution. This half-sized output is processed by a sub-pixel layer to recover the original resolution.

The reasons to choose this network over others -

- Provides fully convolutional property
- Displayed state of the art results in image segmetation[2]
- Other work has explored residual connections, but it is not needed here as the input and output come from different color spaces
- Another consideration is memory consumption, the architecture was chosen such that it should be able to process a full-resolution image(4240×2832) in GPU memory. Hence fully-connected layers are avoided.

4 Training

The network is trained from scratch with mean $L1$ loss and Adam Optimizer. Other types of losses including $L2$ loss was tried out but none of them(including L1 loss) performed remarkably better than other. The network displayed above is summarized in Table 2 and 3. During training, the input to the network is the raw data of the short-exposed images and the ground truth is the corresponding long-exposed image in sRGB space(processed by python-libraw(rawpy) library). The amplification ratio is set to be the ratio of exposure between the ground truth image and the input image. During each iteration, a random 512×512 sample is cropped from the input and applied random flip, vertical flip and transpose(each with a probability of 0.5)is applied. Similar processing is done wrt the ground truth image to maintain consistency. The learning rate is set at 10^{-4}.

Layer(type)	Output shape
Conv2d	[-1, 32, 512, 512]
LeakyRelu	[-1, 32, 512, 512]
BatchNorm	[-1, 32, 512, 512]
Conv2d	[-1, 32, 512, 512]
LeakyRelu	[-1, 32, 512, 512]
BatchNorm	[-1, 32, 512, 512]
MaxPool2d	[-1, 32, 512, 512]
Conv2d	[-1, 64, 256, 256]
LeakyRelu	[-1, 64, 256, 256]
BatchNorm	[-1, 64, 256, 256]
Conv2d	[-1, 64, 256, 256]
LeakyRelu	[-1, 64, 256, 256]
BatchNorm	[-1, 64, 256, 256]
MaxPool2d	[-1, 64, 256, 256]
Conv2d	[-1, 128, 128, 128]
LeakyRelu	[-1, 128, 128, 128]
BatchNorm	[-1, 128, 128, 128]
Conv2d	[-1, 128, 128, 128]
LeakyRelu	[-1, 128, 128, 128]
BatchNorm	[-1, 128, 128, 128]
MaxPool2d	[-1, 128, 128, 128]
Conv2d	[-1, 256, 64, 64]
LeakyRelu	[-1, 256, 64, 64]
BatchNorm	[-1, 256, 64, 64]
Conv2d	[-1, 256, 64, 64]
LeakyRelu	[-1, 256, 64, 64]
BatchNorm	[-1, 256, 64, 64]
MaxPool2d	[-1, 256, 64, 64]
Conv2d	[-1, 512, 32, 32]
LeakyRelu	[-1, 512, 32, 32]
BatchNorm	[-1, 512, 32, 32]
Conv2d	[-1, 512, 32, 32]
LeakyRelu	[-1, 512, 32, 32]
BatchNorm	[-1, 512, 32, 32]

Table 2: Network Architecture(DownSample)

Layer(Type)	Output Shape
ConvTranspose2d	[-1, 256, 64, 64]
Conv2d	[-1, 256, 64, 64]
LeakyRelu	[-1, 256, 64, 64]
BatchNorm	[-1, 256, 64, 64]
Conv2d	[-1, 256, 64, 64]
LeakyRelu	[-1, 256, 64, 64]
BatchNorm	[-1, 256, 64, 64]
ConvTranspose2d	[-1, 128, 128, 128]
Conv2d	[-1, 128, 128, 128]
LeakyRelu	[-1, 128, 128, 128]
BatchNorm	[-1, 128, 128, 128]
Conv2d	[-1, 128, 128, 128]
LeakyRelu	[-1, 128, 128, 128]
BatchNorm	[-1, 128, 128, 128]
ConvTranspose2d	[-1, 64, 256, 256]
Conv2d	[-1, 64, 256, 256]
LeakyRelu	[-1, 64, 256, 256]
BatchNorm	[-1, 64, 256, 256]
Conv2d	[-1, 64, 256, 256]
LeakyRelu	[-1, 64, 256, 256]
BatchNorm	[-1, 64, 256, 256]
ConvTranspose2d	[-1, 32, 512, 512]
Conv2d	[-1, 32, 512, 512]
LeakyRelu	[-1, 32, 512, 512]
BatchNorm	[-1, 32, 512, 512]
Conv2d	[-1, 32, 512, 512]
LeakyRelu	[-1, 32, 512, 512]
BatchNorm	[-1, 32, 512, 512]
Conv2d	[-1, 12, 512, 512]
DepthToSpace	[-1, 3, 1024, 1024]

Table 3: Network Architecure(Upsample)

5 Results

Because of enormous dataset size, we couldn't train the network for 4000 epochs. Instead, it was trained for 335 epochs. During training after every 10 epochs the patches were saved as images. We observed some decent improvement in the image quality generation. It can be observed in the Figures below.

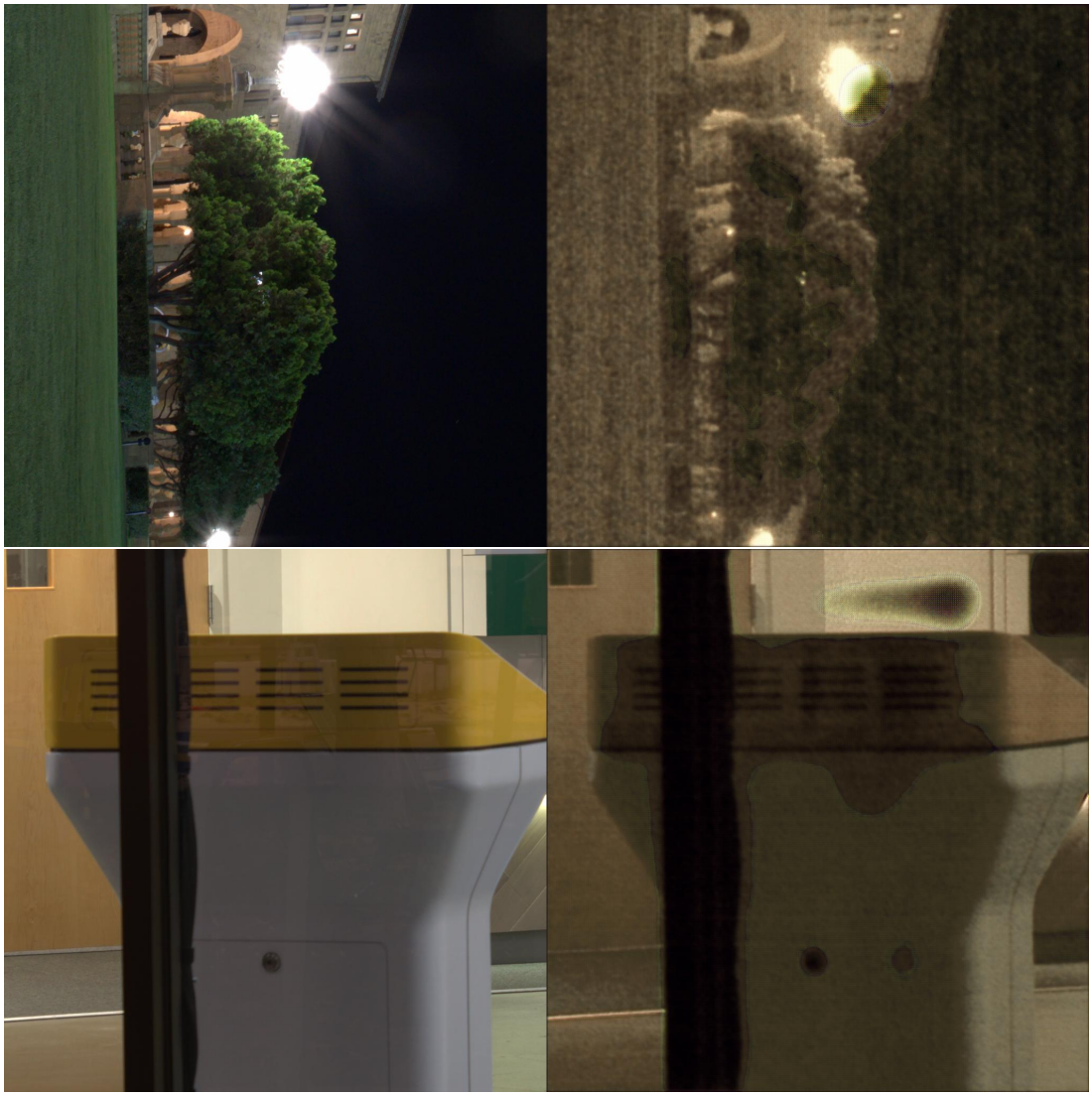

Figure 4: After 10 epochs of training. Left image is ground truth high exposure shot while right image is generated from the network

The generated images blurry and much uncolored.

Figure 5: After 100 epochs of training. Left image is ground truth high exposure shot while right image is generated from the network

The noise in the images starts to reduce. Color starts generating.

Figure 6: After 150 epochs of training. Left image is ground truth high exposure shot while right image is generated from the network

Images are much less noisy and the color reproduction is more accurate

Figure 7: After 200 epochs of training. Left image is ground truth high exposure shot while right image is generated from the network

The network starts to learn generating scene colors in high contrast

Figure 8: After 300 epochs of training. Left image is ground truth high exposure shot while right image is generated from the network

Network starts doing a good job in multicolor scenaries.

As we can observe that the network has started to generalize well with multi color scenaries, we test it out on the test set images in variety of backgrounds, lighting conditions etc. The results even after $<10\%$ of recommended training were quite promising.

So as can be clearly seen through the images that the mentioned network is learning to make the extremely dim images visible clear. As claimed and shown

Figure 9: MidDark Input, Ground Truth, Output image

Figure 10: Less Dark Input, Ground Truth, Output image

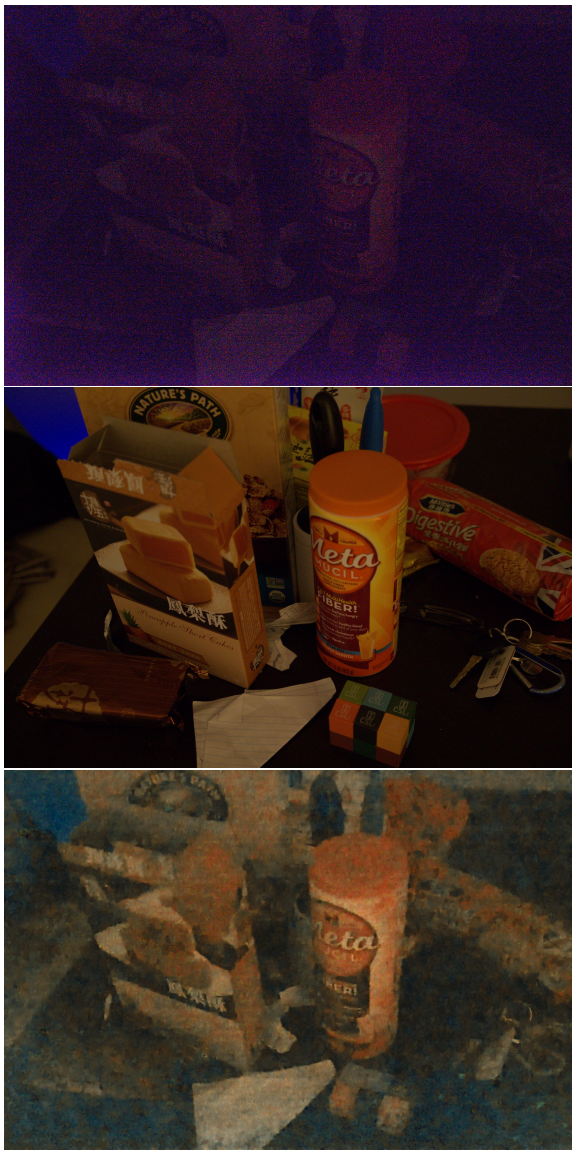

Figure 11: Extremely Dark Input, Ground Truth, Output image

in the [paper of given more training time, the network should be able to create real looking night images with more visual clearity and surpass the traditional computer vision pipeline based results and set up a new benchmark.

6 Conclusion

With the fore-mentioned end-to-end learnt pipeline, we saw that we can a train a FCN network(UNet in this case) to do remarkably great job at making extremely dim night images to visibly clear images. There are still performance issues few of them can be attributed to lack of training time while few can be attributed to ineffiicency in the pipeline itself. I believe the mentioned results can be improved by one or more of the following methods -

1. Changing the network architecture to more advanced and recent architecture like DeepLab v3[3] etc.
2. This being a long training problem hence using Adam optimizer might not be a proper choice due to it's dependence on past gradient values
3. The authors use separate networks for each kind of sensor array. There is a future possible research in unifying it and training it end-to-end.
4. This implemetation might suffer a hard-hit in situations when the object in the image vary in scales as it might not be able to distinguish and identify it separately, hence this network can be trained with an RPN for better results. These are intuitive thoughts and will need to be implemented and verified outs

References

[1] Chen Chen, Qifeng Chen, Jia Xu, and Vladlen Koltun. Learning to see in the dark. In *CVPR*, pages 3291–3300. IEEE Computer Society, 2018.

[2] O. Ronneberger, P.Fischer, and T. Brox. U-net: Convolutional networks for biomedical image segmentation. In *Medical Image Computing and Computer-Assisted Intervention (MICCAI)*, volume 9351 of *LNCS*, pages 234–241. Springer, 2015. (available on arXiv:1505.04597 [cs.CV]).

[3] Yujie Wang, Simon Sun, Jahow Yu, and Limin Yu. Skin lesion segmentation using atrous convolution via deeplab v3. *CoRR*, abs/1807.08891, 2018.