Rutgers University

ECE579 Efficient Machine Learning

Learning To See In The Dark

Authors:
Huy Phan (hvp24)

Instructors:
Prof. Bo Yuan

May, 2019

Contents

1 Definition

1.1 Project Overview

One of the most exciting development in Deep Learning has been the rise of end-to-end Deep Learning. End-to-end deep learning has the potential to replace many data processing systems with multiple stages with just a single neural network. At the moment, there are many applications of end-to-end Deep Learning currently in use:

- Speech Recognition: Traditional pipeline usually consist of many stages from using Mel-frequency cepstral coefficients (MFCCs) to extract hand designed features, then from those features extract phonemes. After that strings together phonemes to form words, the finally from words gets transcript. End-to-end DL approaches can map raw audio signal to transcript directly.

- Face Recognition: Non-DL approach requires many stages from face detection, face normalization, feature extraction (vector), DB based feature matching, then finally face ID. Modern end-to-end DL could shorten the whole pipeline by just using a single neural network.

- Image Noise Reduction: Traditional pipeline that is currently in use by many applications consists of white balancing raw data, the applying demosaic, sharpen and denoise, color space conversion, gamma correction, then finally render the output. End-to-end DL has the potential to replace this pipeline by using convolutional neural network.

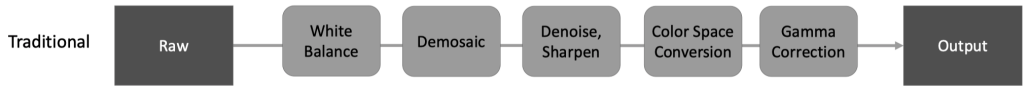


Figure 1: Traditional denoising pipeline [4]

In in project, I focused on the noise reduction application of end-to-end deep learning. Other work related to end-to-end deep learning includes:

- Stacked sparse denoising auto-encoders (SSDA) [8]

- Trainable nonlinear reaction diffusion (TNRD) [3]

- Multi-layer perceptrons [1]
- Deep autoencoders [6]
- Convolutional networks [5]

As pointed out by Chen et al., most existing methods have been evaluated on synthetic data, such as images with added Gaussian or salt and pepper noise. The "Learning in the Dark" paper collected its own data, with short-exposure images and corresponding ground true long-exposure images.

1.2 Problem Statement

Traditional image noise reduction pipeline has many drawbacks such as its effectiveness is limited in extreme conditions, such as video-rate imaging at night. The challenge of fast imaging in low light is wellknown in the computational photography community, but remains open. Hence the need for a better approach arises. In this project, I attempted to implement the "Learning to See in the Dark" CVPR 2018 paper written by Chen Chen, Qifeng Chen, Jia Xu, and Vladlen Koltun [2]. The paper's code was released in GitHub using Tensor Flow deep learning framework, however I chose to re-implement the paper in PyTorch because it is more intuitive to use.

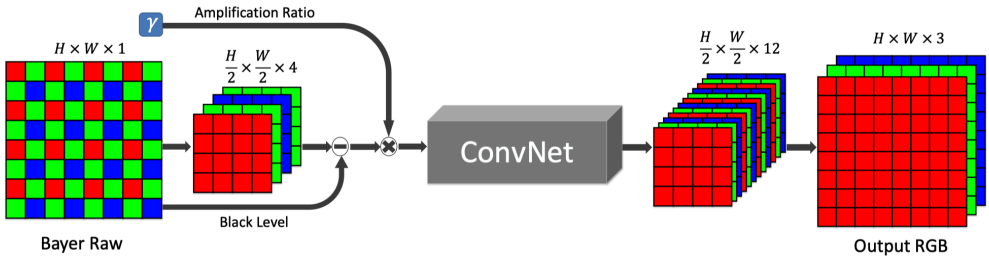


Figure 2: End-to-End Deep Learning Denoising [4]

1.3 Metrics

Similar to Chen et al, I used Peak signal-to-noise ratio (PSNR) to measure the performance of the model. PSNR is an engineering term for the ratio between the maximum possible power of a signal and the power of corrupting noise that affects the fidelity of its representation. Because many signals

have a very wide dynamic range, PSNR is usually expressed in terms of the logarithmic decibel scale.

PSNR is most defined via the *Mean Squared Error (MSE)*. Given a noise-free $m \times n$ monochrome image I and its noisy approximation K, MSE is defined as:

$$MSE = \frac{1}{m \cdot n} \sum_{i=0}^{m-1} \sum_{j=0}^{n-1} [I(i,j) - K(i,j)]^2$$

The PSNR (in decibel dB) is defined as:

$$PSNR = 10 \cdot \log_{10} \left(\frac{MAX_I^2}{MSE} \right) \tag{1}$$

$$= 20 \cdot \log_{10} \left(\frac{MAX_I}{\sqrt{MSE}} \right) \tag{2}$$

$$= 20 \cdot \log_{10} (MAX_I) - 10 \cdot \log_{10} (MSE) \tag{3}$$

Here, MAX_i is the maximum possible pixel value of the image. When the pixels are represented using 8 bits per sample, this is 255. More generally, when samples are represented using linear. For color images with three RGB values per pixel, the definition of PSNR is the same except the MSE is the sum over all squared value differences (now for each color, i.e. three as much differences as in a monochrome image) divided by image size and by three.

2 Analysis

2.1 Data Exploration

The authors of "Learning To See In The Dark" provides 2 datasets for the paper: Sony (Bayer sensor) and Fuji (X-Trans sensor). I used the Sony dataset in my implementation. The See-in-the-Dark (SID) Sony dataset contains multiple raw short exposure images, each with a corresponding long-exposure reference image. Multiple short-exposure images can correspond to the same long-exposure reference image. The number of distinct long-exposure reference images in SID is 424. The images are captured both outdoor and indoor with the illuminance at the camera ranging from 0.03 to 5 lux.

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. 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.

2.2 Exploratory Visualization

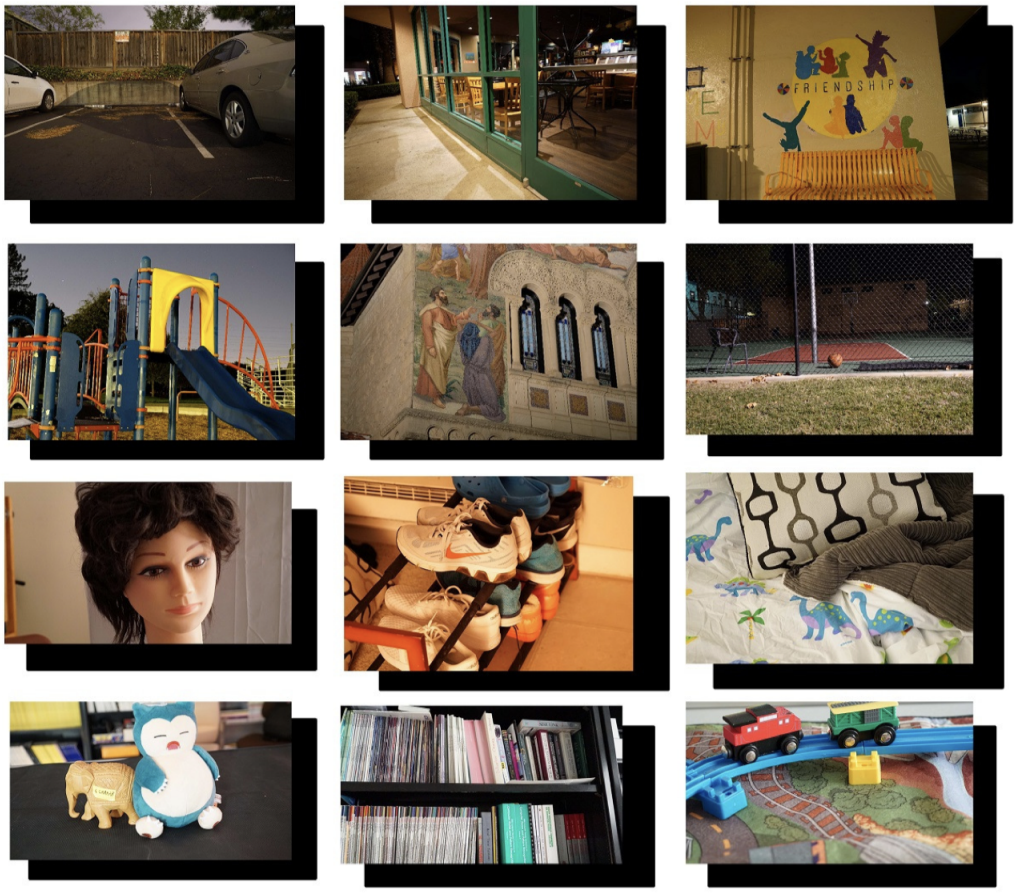


Figure 3: Images in SID dataset. Long exposure reference (ground truth) images are shown in front. Short-exposure input images (essentially black) are shown in the back.

2.3 Algorithms and Techniques

The neural network architecture used in the paper was the U-net [7]. Basically, U-net is a deep convolutional neural network with 2 parts. The left side of the "U-net" are multiple down-sample convolutional layers. As we go deeper, the size of the input images decrease and the number of filters increases. The right side of the "U-net" does the opposite, it consist of multiple up-sample convolutional layers. As we go up the U shape, the size of the input images increase and the number of filters decreases.

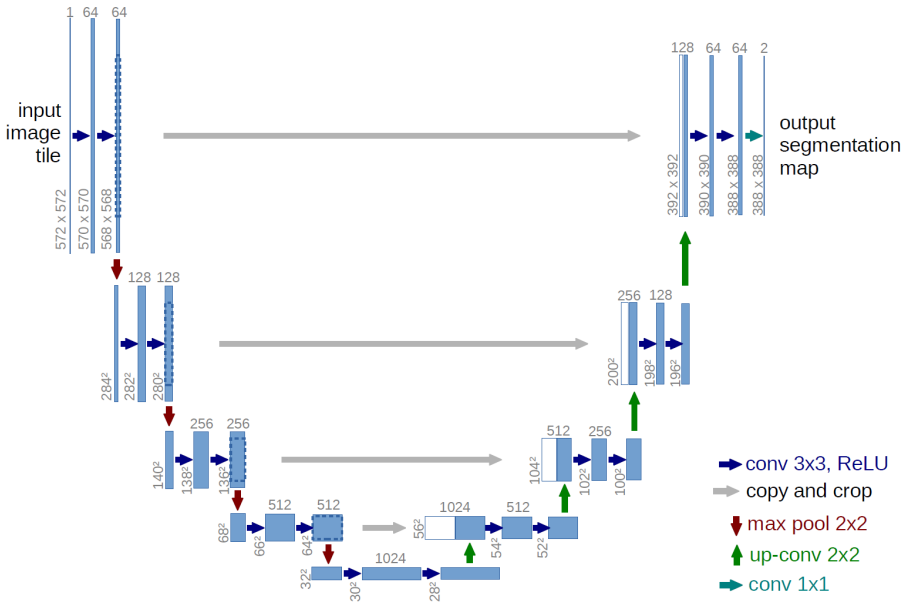


Figure 4: U-net architecture [7]

2.4 Benchmark

To determine the performances obtained by my solution, I compared my results to with the results produced by Chen et al [2].

	Peak Signal to Noise Ratio
Chen et al.	28.88

3 Methodology

3.1 Data Preprocessing

Bayer raw sensor data is packed into 4 channels (Red - Green - Blue - Green) and then multiply (element-wise) by the amplification ratio. There is a library call RawPy which makes this process easy.

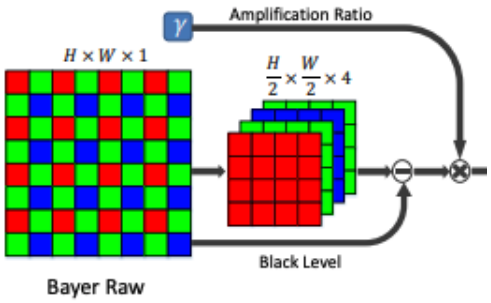


Figure 5: Unpack Bayer sensor data into 4 channels and amplify the data [7]

The amplification ratios is the ratio between the the shutter speed of the long exposure image (ground truth) and the short exposure image. Chen et al. provides a text file of corresponding short exposure and long exposure images with the shutter speed in the file names:

	X	Y	ISO	F-stop
0	./Sony/short/00001_00_0.04s.ARW	./Sony/long/00001_00_10s.ARW	ISO200	F8
1	./Sony/short/00001_00_0.1s.ARW	./Sony/long/00001_00_10s.ARW	ISO200	F8
2	./Sony/short/00001_01_0.04s.ARW	./Sony/long/00001_00_10s.ARW	ISO200	F8
3	./Sony/short/00001_01_0.1s.ARW	./Sony/long/00001_00_10s.ARW	ISO200	F8
4	./Sony/short/00001_02_0.1s.ARW	./Sony/long/00001_00_10s.ARW	ISO200	F8

Figure 6: Correspond short exposure and long exposure images [7]

For ground truth data (long exposure images), I rendered the raw sensor data to PNG format file because the convolutional neural network will output tensor in that format. All tensors are normalize to range of [0, 1]. Next, I applied a random crop of 512 x 512 patch for training and apply random

ipping and rotation for data augmentation. I used a batch size of of 32 for training. The corresponding ground truth images size are two time as that. So the training input and ground truth tensor have the following shape

- X: (32 x 4 x 512 x 512)
- Y: (32 x 3 x 1024 x 1024)

3.2 Implementation

The model architecture consists of several blocks:

- DoubleConvolution(in-ch, out-ch):
    1. Conv2d(in-ch, out-ch, kernel-size=3, stride=1, padding=1)
    2. BatchNorm2d(out-ch)
    3. LeakyRelu(a=0.2)
    4. Conv2d(in-ch, out-ch, kernel-size=3, stride=1, padding=1)
    5. BatchNorm2d(out-ch)
    6. LeakyRelu(a=0.2)
- DownSample(in-ch, out-ch):
    1. Maxpool2d(size=2)
    2. DoubleConvolution(in-ch, out-ch)
- Upsample(in-ch, out-ch):
    1. ConvTranspose2d(in-ch, in-ch//2, 2, stride=2): Conv2d with respect to its input. It is also known as a fractionally-strided convolution or a deconvolution (although it is not an actual deconvolution operation).
    2. DoubleConvolution(in-ch, out-ch)
    3. Concatenate
- PixelShuffle: re-arrange 4 channels (Red-Green-Blue-Green) to a final image.

The U-net was implemented by stacking these blocks as follow:

1. x1 = DoubleConvolution(4, 32) (X)
2. x2 = DownSample(32, 64) (x1)
3. x3 = DownSample(64, 128) (x2)
4. x4 = DownSample(128, 256) (x3)
5. x5 = DownSample(256, 512) (x4)
6. x = UpSample(512, 256) (x5, x4)
7. x = UpSample(256, 128) (x, x3)
8. x = Upsample(128, 64) (x, x2)
9. x = Upsample(64, 32((x, x1)
10. x = Conv2d(32, 12) (x)
11. x = PixelSuffle(2) (x)

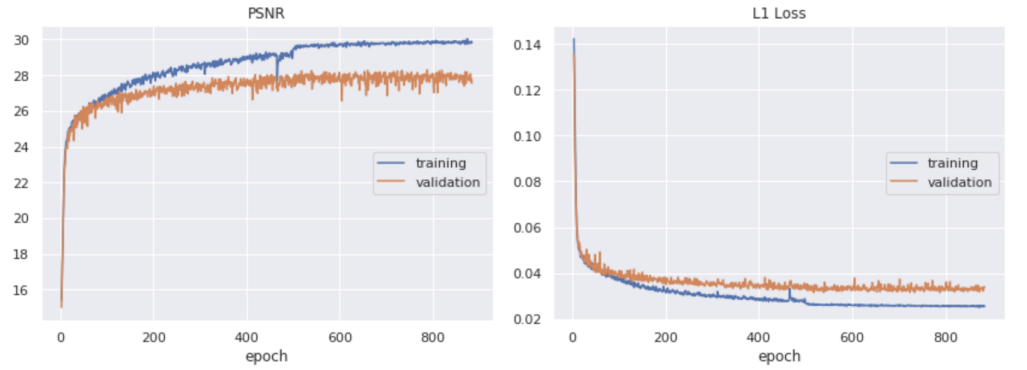
These are the hyper-parameters choice I made:

- Loss: L1 loss
- Optimizer: Adam optimizer with AMSgrad
- Learning rate: 1e-4 during first 1000 epochs and 1e-5 during epoch 1000 to 2000 epochs
- Batch Size: 32

4 IV. Results

4.1 Model Evaluation and Validation

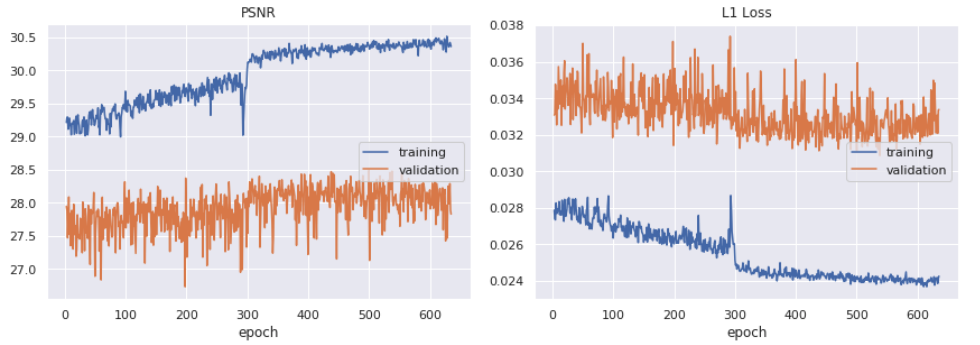
I trained the model for 2000 epochs. Total training time was 50 hours on a NVIDIA Tesla V100.




```
 88%|████████  | 884/1001 [24:37:05<3:14:40, 99.84s/it]
PSNR:
training   (min:   13.218, max:   30.039, cur:   29.824)
validation (min:   13.152, max:   28.293, cur:   27.553)

L1 Loss:
training   (min:    0.025, max:    0.193, cur:    0.026)
validation (min:    0.032, max:    0.178, cur:    0.034)
```

Figure 7: Train the model (Epoch 0-1000/2000)




```
 70%|███████   | 635/901 [17:41:33<7:24:03, 100.16s/it]
PSNR:
training   (min:   28.993, max:   30.512, cur:   30.358)
validation (min:   26.729, max:   28.483, cur:   27.826)

L1 Loss:
training   (min:    0.024, max:    0.029, cur:    0.024)
validation (min:    0.031, max:    0.037, cur:    0.033)
```

Figure 8: Train the model (Epoch 1000-2000/2000)

Figure 9: Example 1: Images produced by the model

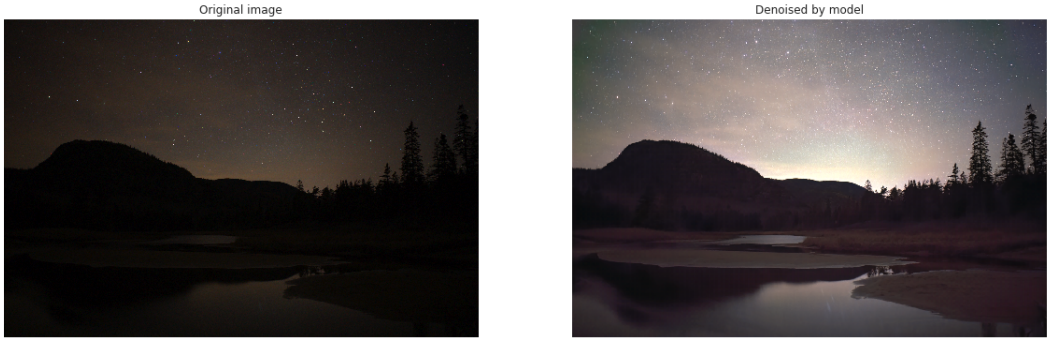


Figure 10: Example 2: Images produced by the model

Cross-sensor results are worse, as expected, but still better than traditional pipeline. This is the image of my friend I took with my iPhone and processed with the model.

Figure 11: Example 3: Cross-sensor generalization

4.2 Justification

Compared with the PSNR of Chen et al.

	Peak Signal to Noise Ratio
Chen et al.	28.88
My model	28.83

5 Conclusion

Given the baseline PSNR of 28.88, my model's PSNR of 28.83 is quite close. I think I could beat the original paper implementation if I train the model for a few hundreds more epochs. Deep neural network has a very promising in denoising and enhancing very low light images. The future of computational photography looks very bright (in the dark)!

References

[1] HC. Burger, CJ. Schuler, and S. Harmeling. "Image denoising: Can plain Neural Networks compete with BM3D?" In: June 2012, pp. 2392–2399.

[2] Chen Chen et al. "Learning to See in the Dark". In: *CoRR* abs/1805.01934 (2018). arXiv: 1805.01934. URL: http://arxiv.org/abs/1805.01934.

[3] Yunjin Chen and Thomas Pock. "Trainable Nonlinear Reaction Diffusion: A Flexible Framework for Fast and Effective Image Restoration". In: *CoRR* abs/1508.02848 (2015). arXiv: 1508.02848. URL: http://arxiv.org/abs/1508.02848.

[4] Xiaojie Guo, Yu Li, and Haibin Ling. "LIME: Low-Light Image Enhancement via Illumination Map Estimation". In: *Trans. Img. Proc.* 26.2 (Feb. 2017), pp. 982–993. ISSN: 1057-7149. DOI: 10.1109/TIP.2016.2639450. URL: https://doi.org/10.1109/TIP.2016.2639450.

[5] Viren Jain and H. Sebastian Seung. "Natural Image Denoising with Convolutional Networks". In: *Proceedings of the 21st International Conference on Neural Information Processing Systems*. NIPS'08. Vancouver, British Columbia, Canada: Curran Associates Inc., 2008, pp. 769–776. ISBN: 978-1-6056-0-949-2. URL: http://dl.acm.org/citation.cfm?id=2981780.2981876.

[6] Kin Gwn Lore, Adedotun Akintayo, and Soumik Sarkar. "LLNet: A Deep Autoencoder Approach to Natural Low-light Image Enhancement". In: *CoRR* abs/1511.03995 (2015). arXiv: 1511.03995. URL: http://arxiv.org/abs/1511.03995.

[7] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. "U-Net: Convolutional Networks for Biomedical Image Segmentation". In: *CoRR* abs/1505.04597 (2015). arXiv: 1505.04597. URL: http://arxiv.org/abs/1505.04597.

[8] Li Xu et al. "Deep Edge-Aware Filters". In: *Proceedings of the 32nd International Conference on Machine Learning*. Ed. by Francis Bach and David Blei. Vol. 37. Proceedings of Machine Learning Research. Lille, France: PMLR, July 2015, pp. 1669–1678. URL: http://proceedings.mlr.press/v37/xub15.html.