

HDR image reconstruction from a single short-exposure image

ZhuoYi Cai
zycai@stanford.edu

Abstract—Cameras on mobile devices, such as cell phones, have small sensor pixels which limit the number of photos capture, leading to limited dynamic range images. Conventionally, in order to create high dynamic range (HDR) images, a set of images with different exposures is captured and then combined in post-processing. This approach faces a challenge in image fusion where different parts of the fused image represent the scene at different times. Another challenge faced by mobile devices is power-constraint. Capturing a burst of images results in cleaner and less noisy images but it consumes power, time and memory space. In this project, we propose a new approach to reconstruct a HDR image from a single low exposure image using a deep neural network. We will see and compare the results from the networks trained using different image patch sizes. We will also discuss what we have learned from this project and the potential improvement to the algorithm.

Index Terms—Computational Photography

1 INTRODUCTION

THE real-world scenes are high dynamic range by nature. But consumer digital camera sensors have limited capacity to capture a wide range of luminance. As a result, the produced images very often contain under or over-exposed regions. The conventional approaches proposed to generate high dynamic range images by combining set of low dynamic range images of the scene at different exposure levels. However, these conventional methods need to handle the scene motion which gives a huge challenge to image fusion process. Another challenge is power and memory space. For mobile devices which are battery-powered, power and memory consumptions are critical in the real-world application. With single image dynamic range expansion, it not only avoids image fusion issue but also save power and require less memory space.

In this project, we conducted a training experiment on an architecture which is a modified version of U-Net to demonstrate the possibility to reconstruct a high dynamic range image from a raw RGGB image. We built the network from scratch and trained the network with random patches from the raw image. We will discuss several observations and common issues we have from reconstructed image from neural network. However, the result shows the potential of this approach becoming an alternative to the other existing methods.

2 RELATED WORK

As mentioned in the abstract, conventionally, in order to create high dynamic range (HDR) images, a set of images with different exposures is captured and then combined in post-processing. There are a couple challenges digital cameras on consumer mobile devices in particular face with this approach. Due to the different exposure times for different capture, producing good image fusion where different parts of the fused image represent the scene at different times is particularly difficult. To solve this problem,

one modern approach in Google’s HDR+ [1] is to capture a burst of images with constant exposure and combining them with dynamic range compression. Another challenge faced by mobile devices is power-constraint. Capturing a burst of images results in cleaner and less noisy images but it consumes power, time and memory space. Other approach to reconstruct HDR images from a single exposure using deep convolutional neural network has also been proposed [2][3]. These approaches attempt to hallucinate the overblown regions of the images. However, they may introduce patterns that do not exist in the scene or not always able to reconstruct all the details.

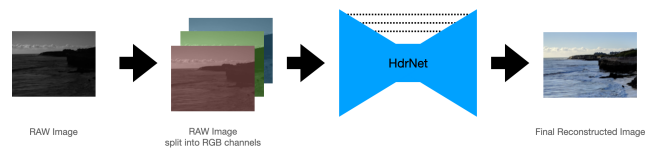


Fig. 1: Architecture Overview

3 PROPOSED METHOD

In this project, we studied one proposed network architecture – HdrNet. HdrNet is a modified version of U-Net architecture which includes encoder and decoder networks with skip connections. We conducted experiments on training the network with different size of image patches. The idea is to attempt to reconstruct a HDR image from a single low exposure image using a deep neural network. We are using a low exposure image to avoid blowing out highlights as in HDR+ method. This may solve the issues faced in highlight hallucination. The input raw image is first “split” into Red, Green, and Blue channels with missing pixel initialized to 0 without any demosaicing operation. This “split” image

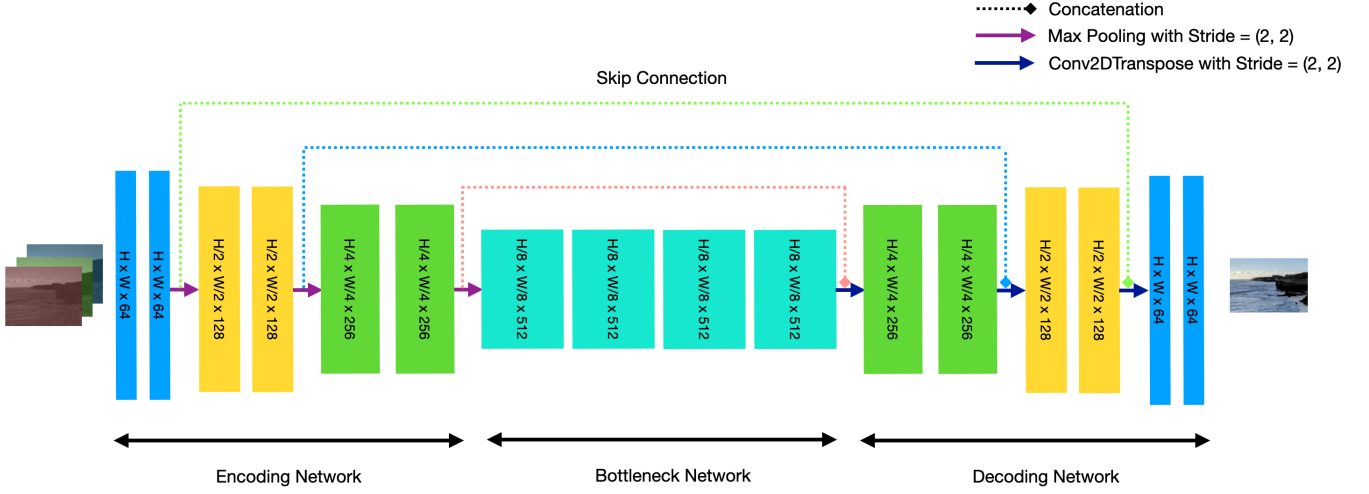


Fig. 2: HdrNet Architecture

is then passed through the network to allow the network to hallucinate the missing information and produce HDR image result. Since only one low exposure image is needed, we can save time, power and memory in the capturing process. In this session, we will discuss all the details of the implementation and the proposed architecture. In the next section, we will demonstrate some observations and results.

3.1 Datasets

The proposed network is trained and evaluated using the dataset from Google HDR+ Burst Photography Dataset. Specifically, for this project, only a subset of the dataset (153 raw images) will be used. Each image set includes a burst between 2 and 10 12-13 Mpixels raw photos captured with the same exposure time and gain. They are captured using a variety of Android mobile cameras with the public Android Camera2 API. All the mobile cameras used for capturing the dataset have a rolling shutter with a readout time of 1/30 second and should never be larger than 33 ms between successive frames as specified. Only the reference frame (as HDR+ algorithm defines it) of a burst is used in this project. The ground truth is the final result generated from google HDR+ algorithm.

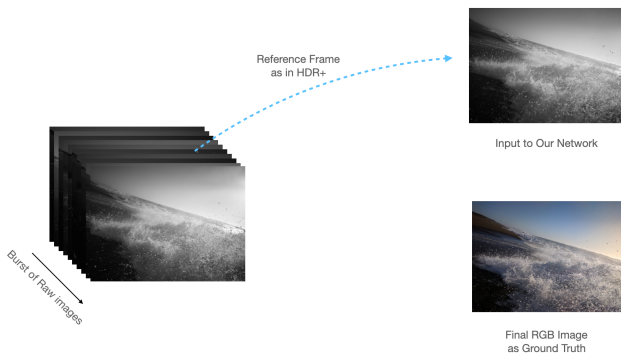


Fig. 3: Dataset

3.2 Architecture

3.2.1 HdrNet

HdrNet is built based on the framework of U-Net[4]. It consists of encoding network, bottleneck network and decoding network with skip connections at each down/up sampling point.

As shown in Fig. 2, there are three stages at the encoding (downsampling) and decoding (upsampling) networks. Each stage consists of two 2D convolution layer with same size configuration. After each downsampling, the number of the filters for these 2D convolution layers is double to maintain the network capacity for retaining image information. Each downsampling step is performed by using a max pooling layer with stride (2, 2) setting. Symmetrically, after each upsampling point, the number of filters is reduced to half. Each upsampling step is performed by using a 2D transpose convolution layer with stride (2, 2).

The bottleneck network consists of 4 2D convolution layers with 512 filters. This section of the network attempts to hallucinate the missing information such that the decoding network can properly reconstruct the image.

The skip connection connects the corresponding encoding and decoding points with concatenation operation at the decoding end. All the layers use Relu as the activation functions. For regularization, L2 regularization with strength 0.2 is used for all layers. And batch normalization is applied to all layers before the activation operation.

3.2.2 Skip Connection

Following U-Net architecture, the proposed networks are equipped with skip connections borrowing the ideas from [8]. The goal is to address two main problems in deep neural network for image reconstruction. First, with deep network, more layers of convolution and de-convolution result in a significant amount of details from the original image lost and corrupted. This phenomenon significant reduces image construction quality. In addition, since our input data is "split" raw image with missing pixel initialized to 0, we want to maintain the data integrity over the entire network

to avoid further image corruption due to encoding process. Second, deep networks, in general, suffer from vanishing gradients and are very difficult to train. This eventually contributes to the network performance degradation. In order to address these issues, we added skip connections between every corresponding downsampling and upsampling point within the network.

3.2.3 Loss Function

For the loss function, we use mean squared error (MSE) between the reconstructed image and the corresponding ground truth HDR image.

$$L(G(p), Y(p)) = \frac{1}{N} \sum_{p=0}^N (G(p) - Y(p)_i)^2$$

where $Y(\cdot)$ represents reconstructed function and N is the number of pixels. In our experiment, all the training images are random $64 \times 64 \times 3$ or $128 \times 128 \times 3$ patches from the raw image. $G(\cdot)$ represents the ground truth function from Google HDR+ algorithm.

3.2.4 Metrics

The evaluation metrics used in this project are SSIM and PSNR. SSIM [7] provides a way to measure the similarity in perceived quality between the reconstructed and ground truth images while PSNR measures the reconstruction quality for the generated images.

4 EXPERIMENTAL RESULTS

4.1 Compute Environment

Colab compute environment is used for all the experiments and network training in this study. GPU and HIGH RAM (26GB) runtime setting are deployed. However, the GPU assignment is solely random based on Google algorithm.

4.2 2D Convolution vs Max Pooling

Initially, we built the encoding downsampling layer with 2D convolution with stride = (2, 2). However, this resulted in images with lost details and artifacts, such as undesired patterns and color distortion. This is shown in the middle image of Fig. 4. There are square-like pattern and color defects all around.



Fig. 4: 2D Conv vs Max Pooling

After we changed the downsampling layer to max pooling with stride (2, 2), the resulting image looks much cleaner in terms of artifacts seen previously. The sky color of the reconstructed image looks decent too.

In terms of metrics, with 2D convolution and Max pooling as downsampling layer, SSIM and PSNR of the image is listed below.

	2D Conv	MaxPooling
SSIM	0.4535	0.5852
PSNR	18.8373	20.1738

TABLE 1. Metrics: 2D Conv vs Max Pooling

4.3 Training Patch Size 64×64 vs 128×128

We only use 40 images (due to time and GPU memory capacity) in this project. The network was trained using 512 64×64 patches per raw image and using 256 128×128 patches per raw image. The red, green and blue pixels are separated into three channels with the missing pixels initialized to 0. The final goal is to let the network hallucinate the missing information for each channel and reconstruct the image with higher dynamic range.

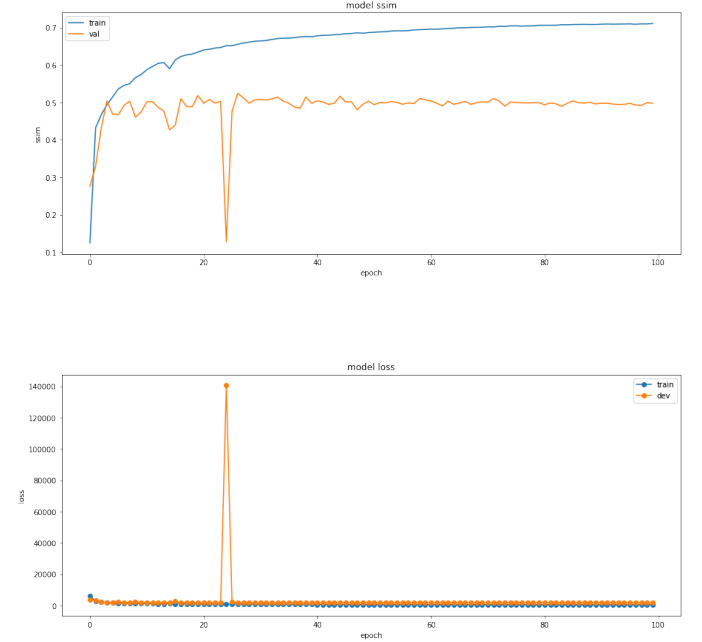


Fig. 5: 128x128 training plot

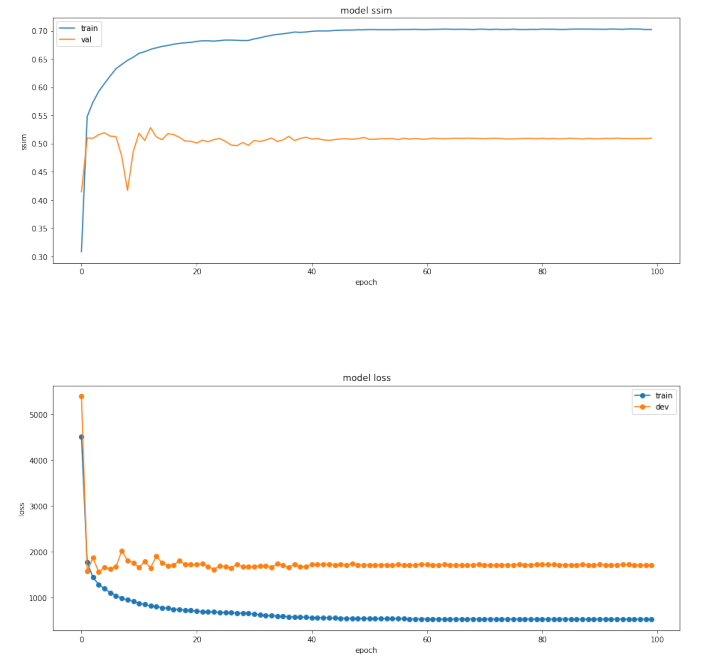


Fig. 6: 64x64 training plot

From the training plot, we see that the network tends to overfit the data. This is due to the fact that there are only 40 images. Even though we have thousands of patches in the training process, this is not sufficient to have a good network. Also, with 64x64 patches, the training process tends to be more stable the making steady progress while with 128x128 patches, the process seems to be more noisy in general. This may be due to the number of patches for 128x128 is 4x less than the batch of 64x64 patches.

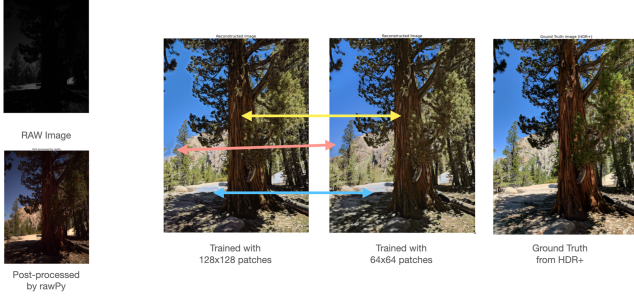


Fig. 7: 128x128 vs 64x64

The resulting images from the network trained with 128x128 patches and 64x64 patches are shown in Fig. 7. The yellow arrow compares the color of tree trunk. It seems the network trained with 128x128 patches produces image with more saturated color. However, it also results in more color distortion in some area (see the blue arrow). In addition, it produces artifacts near the edges of the texture while it looks much cleaner in the image produced by the network trained with 64x64 patches. (see red arrow) Overall, it seems the network trained with 64x64 patches produces more pleasant result perceptually.

5 DISCUSSION

5.1 GPU OOM issue

Since the network was trained using patches of an image, we were not able to include too many data into our training set due to the time constraint. We tried to include as many as images into the training set. As a result, we can only fit 40 images with 512 64x64 patches per image or 256 128x128 patches per image without hitting GPU Out Of Memory (OOM) issue. This limits the network's generalization to validation and test dataset.

5.2 Image Color Distortion

Even though we can observe the color mostly accurate in the reconstructed image with reasonably high in dynamic range, we can also spot some areas with the color distortion which is common among the CNN approaches [3].



Fig. 8: Color Distortion

We can see the color of the stone ground in Fig. 8 becomes blueish in the reconstructed image. This is the result

from the network trained with 128x128 patches. In general, even though there are some artifacts from the reconstructed images, the network trained with 64x64 patches results in much better overall quality of reconstructed images perceptually. And the color distortion is less severe.

5.3 Training Datasets and Regularization

As mentioned above, the network has the potential to be more generalized to the unseen data if there are enough data in the training dataset. However, due to the time limit and memory resource constraint, we can only demonstrate the potential of this HDR image construction approach with DNN. In addition, there is more we can do on the regularization, for example, increasing the weight L2 regularizer or using other technique, such as dropout. One side note is that the raw images and the corresponding ground truth images require manually check to make sure their orientations are aligned.

5.4 Overfitting the Image

Since we only have limited time and data, we attempted to overfit the data and see how much this proposed architecture can potentially produce. So we only trained the network with two images with 4096 64x64 patches per image.

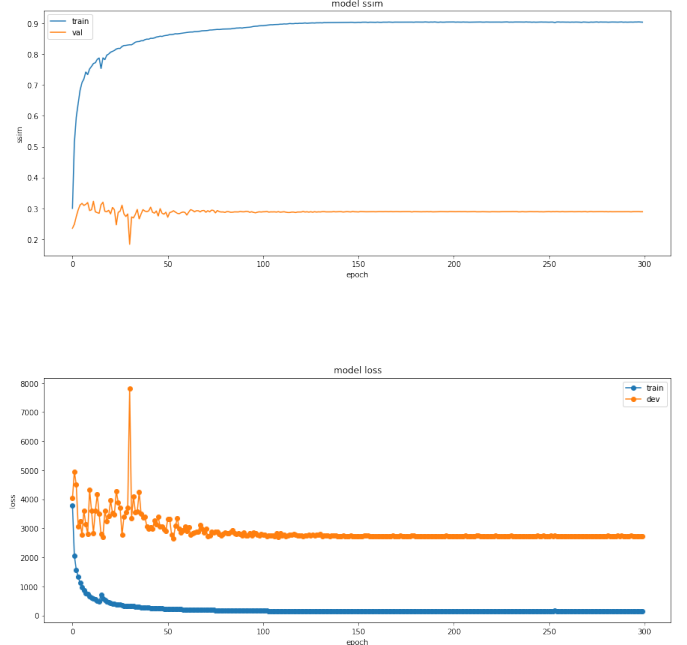


Fig. 9: loss and ssim plot

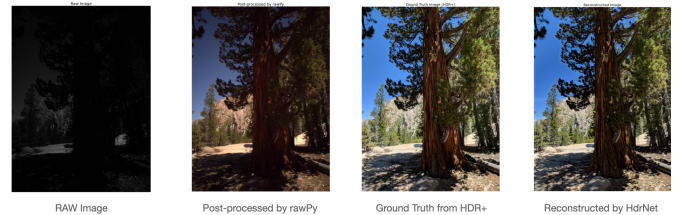


Fig. 10: overfit image demo

From the example shown in Fig. 10, we can see that the network is capable to reproduce the HDR image from a short exposure raw image. A couple issues still exists.

SSIM	0.6475
PSNR	21.0040

TABLE 2. Metrics: Overfitting image

First, the color of some area, such as the tree trunk, is not as saturated as the ground truth image. Also, there is some color distortion around the edges of the tree. See Fig. 11. Since this example is from an overfit image, we would expect this side effect may also exist if the network is generalized to other image data.



Fig. 11: overfit image color distortion

6 CONCLUSION

Finally, in this short project, we can see the potential of the proposed network to be a low power and low memory consumption solution to reproduce HDR image in real time. Given the insufficient training data and a few observed issues, the network demonstrates the possibility of becoming an effective alternative producing HDR image in the mobile device application. This is supported by the fact that there are techniques we can further experiment to solve the observed issues. Certainly, it also requires further work to improve the model in order to achieve the state of the art result presented in other literatures.

7 SOURCE CODE

Source code can be downloaded at [here](#). Access granted to Stanford University Community only. Note that due to the size of the dataset files, this code package doesn't include dataset data. Please see README for instructions to download dataset from Google HDR+ dataset website.

REFERENCES

- [1] Samuel W. Hasinoff, Dillon Sharlet, Ryan Geiss, Andrew Adams, Jonathan T. Barron, Florian Kainz, Jiawen Chen, and Marc Levoy, *Burst photography for high dynamic range and low-light imaging on mobile cameras*, ACM Transactions on Graphics (Proc. SIGGRAPH Asia 2016), 35(6), 12 pp.
- [2] GABRIEL EILERTSEN, JOEL KRONANDER, GYORGY DENES, RAFAŁ K. MANTIUK, JONAS UNGER, *HDR image reconstruction from a single exposure using deep CNNs*, ACM Transactions on Graphics, Vol. 36, No. 6, Article 178. Publication date: November 2017.
- [3] MARCEL SANTANA SANTOS, TSANG ING REN, NIMA KHADEMI KALANTARI, *Single Image HDR Reconstruction Using a CNN with Masked Features and Perceptual Loss*, ACM Trans. Graph., Vol. 39, No. 4, Article 80. Publication date: July 2020.
- [4] Ronneberger, O., Fischer, P., Brox, T., *U-Net: Convolutional Networks for Biomedical Image Segmentation*, MICCAI 2015, arXiv:1505.04597
- [5] Xiao-Jiao Mao, Chunhua Shen, Yu-Bin Yang, *Image Restoration Using Convolutional Auto-encoders with Symmetric Skip Connections*, In CVPR, 2016.
- [6] Schwartz, E., Giryes, R., and Bronstein, A., *DeepISP: Toward Learning an End-to-End Image Processing Pipeline*, IEEE Transactions on Image Processing 28.2 (2019): 912-923 arXiv:1801.06724
- [7] 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):600–612, 2004.
- [8] Xiao-Jiao Mao, Chunhua Shen, Yu-Bin Yang *Image Restoration Using Convolutional Auto-encoders with Symmetric Skip Connections* In CVPR, 2016.

