Learning to see in the dark reproduction

Liam Rees - 5735726, Charles Downs - 4705467, Rico van Buren - 4705467

April 2023

1 Introduction

Achieving high-quality photos taken in areas with little light has always been a challenge in photography. In traditional signal processing, techniques such as HDR tone mapping have been the norm [2]. More recent approaches have adopted a learning approach, where a deep learning network attempts to find a mapping between low-light (short exposure) and normal-light (longer exposure) photos. In particular, we shall discuss the work of Chen et al. [1]. The original paper attempts to learn a network to brighten images based on a dataset of roughly 1000 images for the training set (see Data section below). The authors propose to train the network with two different architectures (UNet and CAN) and on images procured via two different cameras. The authors then compare and tune the (hyper) parameters in order to see how the results vary. These include changing the loss function, changing the data arrangement (packing colors into low-resolution channels, vs masking and duplicating colors), different network architecture, and training on different color modes (raw -¿ sRGB). The different methods used achieve overall impressive results, and are compared using PSNR (Peak Signal-Noise Ratio) and SSIM (Structured Similarity Index). PSNR is used as a metric in order to evaluate overall reconstruction and image quality, and SSIM (which roughly models how humans perceive image similarity) measures on a scale of [0..1] how similar the reconstructed image is to its ground truth.

However, it is important to note, that similarly as super-resolution, this area of image processing is ill-posed, so it is difficult to gauge quality via specific loss.

Throughout this post, we propose to discuss the various results achieved with our reproduction attempt of this paper. The original idea was to use the authors original code (which is openly available) and attempt to train the network on different color channels, and different formats, the motivation being that it is well established, for example, that luminance information is mostly captured in the y-channel, however, the authors do not investigate this. However, our attempts were quickly thwarted when we realize the authors code wouldn't compile with modern packages. The paper, which dates back to 2018, was written using the previous version of Tensorflow (V1) which is now largely deprecated, meaning that the code likely would be difficult to port to Tensorflow V2. Furthermore, in recent years, PyTorch has become the most commonly used DL

Python library, being favoured by academics as well as in industry. Bearing this in mind, we decided that it would be a more productive idea to entirely re-build the code-base in PyTorch, and attempt to reproduce a subset of the results in order to validate the author's findings. A measure proposed by [the authors of troubling trends] suggests that a paper is 'reproducible' if 75% of the claims can be validated, with results only different by 10%.

In addition to re-building the entirety of the original code-base into a more modern framework, we also decided to update the style of the code. The style of the original code-base was not very elegant, did not adopt commonly used standards (such as OOP, code cleanness) and the code was not very 'pythonic', and not adherant to common good-practice.

With this in mind, we proceeded with our reproduction goals. In terms of specific results, we decided to attempt to reproduce a subset of their results, more specifically, we decided to reproduce with the following parameters/choices: architecture: UNet, loss: L1 loss, dataset: the Sony dataset. The motivation behind this reproduction is to see whether the author's claim can be accurately reproduced in terms of PSNR and SSIM. The code to reproduce the results can be found here: `https://github.com/charlesalec/DL-Seeing-In-The-Dark-Reproduction`.

2 Data

The original paper included two large datasets shot on two different cameras. For our reproduction of the code we used the dataset shot on the Sony camera, as it is smaller in size and easier to work with.

The data includes training images and label images. The training images are short-exposure raw image files. There are several (typically 10 or so) training images corresponding to each label image. The training images are taken with shutter times of 0.1s, 0.04s, and 0.033s. Because the exposure time is so short, these raw images are very dark, as can be seen in Figure 1 the goal of this paper and our reproduction is to brighten them.

Figure 1: Sample Training Image

The label images are long exposure (10-second or 30-second shutter time) images able to take in a great deal of light. The loss function when training will compare these label images against images brightened by the network to train it. Separate label images put aside in a test set will be compared against the trained network output to evaluate the performance of the network using SSIN. An example of the label can be found in Figure 2.

Figure 2: Sample Label Image

3 Reproduction

The original code is written in Python 2.7 with TensorFlow 1.1. This means that the original code can not be used with the Python 3 and TensorFlow 2. We opted to reproduce the paper in Python 3 with PyTorch 2.0, since we are more familiar with this. The network is a convolutional neural network with

a U-Net architecture. The paper does not list the sizes of the layers, so we extracted them from the original code. Setting up the network was quite doable once we figured out how to read the TensorFlow code. However, the network is not the only thing that needs to be reproduced. A very important thing is the pre-processing of the data. The input data is in the Bayer Raw format which is converted to 4 channels: red, green, blue and green. So there are two green channels. This is because the Bayer Raw contains two green channels. Converting the images is done with the following code:

```
def pack_raw_sony(raw):
        im = raw.raw_image_visible.astype(np.float32)
        im = np.maximum(im - 512, 0) / (16383 - 512)

        im = np.expand_dims(im, axis=2)
        img_shape = im.shape
        H = img_shape[0]
        W = img_shape[1]

        out = np.concatenate((im[0:H:2, 0:W:2, :],
                              im[0:H:2, 1:W:2, :],
                              im[1:H:2, 1:W:2, :],
                              im[1:H:2, 0:W:2, :]), axis=2)
        return out
```

From this conversion the black level is subtracted, this value is also found in the original code, and is multiplied with the ratio between the exposure time of the sample and truth label. An overview of the architecture can be found in Figure 3.

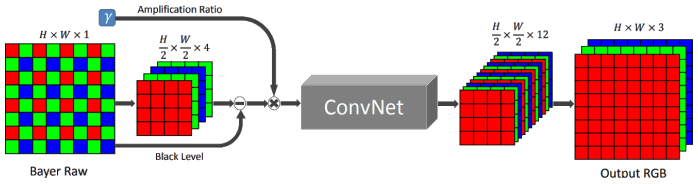

Figure 3: Architecture

To train the network, the truth labels are processed with rawpy without auto-brightening the image. The training samples are fed in random order through the network, but only a random patch of 512 by 512 is used of the image. The loss is calculated with the L1 loss. The original paper trains for 4000 epochs. We only trained for 100 epochs since this already took 10 hours using rather decent hardware with CUDA (1080 Ti). The original paper also flips and translates the patches, which we did not do, we kept to simply random crops of a fixed size. To reproduce this paper we should come close to a PSNR

of 28.88 and a SSIM of 0.787. These were the values achieved by the original authors with the default pipeline.

4 Experimental Results

The results start to get satisfying after about 80 epochs, out of the total of 100 epochs for which we let the network train. Under 80 epochs, the results are not very ideal – particularly, it takes the network at least 80 epochs before it begins to start learning color mappings from the dark images to the bright images. Beyond 95 iterations, overall results are strong as can be seen in Figure 4.

Figure 4: Sample test set result. Label Left, Reconstruction Right

Something that we are not happy with is that there is a sort of artifacting effect when bright lights are in the image as can be seen in Figure 6; we think this may be due to the fact that we only train our network for 100 epochs (as opposed to 4000 in the paper) or our slightly simpler training loop. From the testing we ran, it appears that colors are only learned after about 1000 epochs; it is possible that the high light areas are also learned much later, meaning we can assume that there's some form of linear progression of learning: colors first, then the network learns a wider range of colors (since the range at first is quite narrow), followed by extremely bright colors, or white-light.

Figure 5: A different test result showing artifacts when bright lights are in the image. Label image Left, Reconstruction image Right

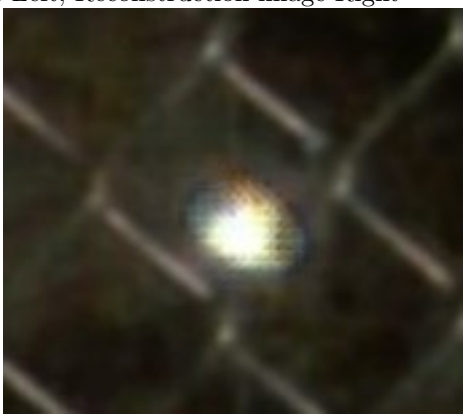

Figure 6: zoomed in on the artifacting

The loss is calculated with the L1 loss. For every image that is used during training the loss is calculated. The training set consists of 944 images, so for 100 epochs the loss is calculated 94400 times. The training loss is found in Figure 7.

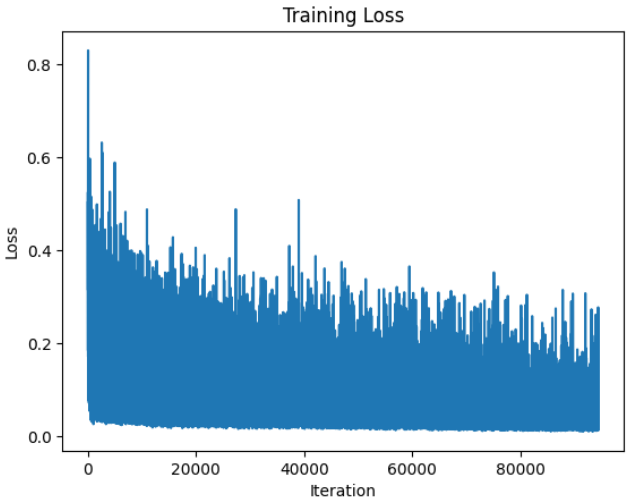

Figure 7: Train Loss

The loss for the test is also calculated with the L1 loss after the network has finished training. The plot in Figure 8 depicts the loss of all the test images.

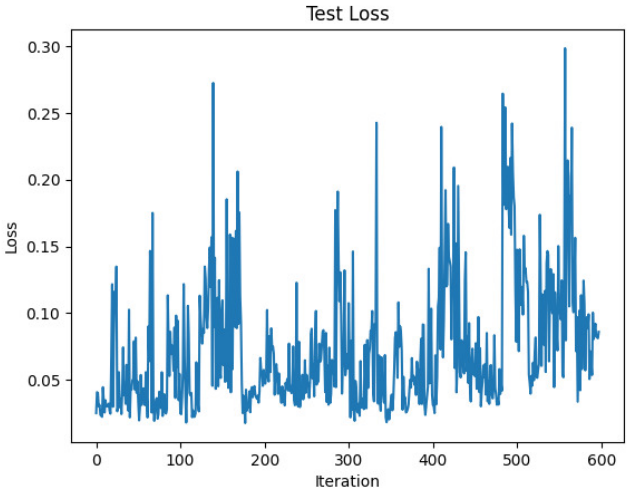

Figure 8: Test Loss

Finally, in order to compare the results to the original paper, we calculated the mean PSNR and SSIM on the test set. Since we only considered a subset of parameters, the following is with: L1 loss, 100 epochs, UNet architecture, and `lr=1e-4`, and yielding:

- PSNR: 65.64700446662502
- SSIM: 0.21268047

The first aspect to note is that these results are highly conflicting; both amongst themselves and when compared to the original papers results. The original paper purports a PSNR of 28.88, where a higher PSNR means a better reconstruction. This would suggest that somehow, our results outperform the results of the paper in terms of reconstruction, however, when manually evaluating the quality of our results, it is clear that the author has significantly better performance. But this once again attests to the fact that there is no perfect metric for evaluating image similarity, since it is very difficult to model the human visual system. Looking at the SSIM, the original paper reports an SSIM of 0.78, meaning that that their results are significantly better than ours. Which is a stark contrast to the PSNR values we reported above. With these results in mind, we have to note that these results are clearly inconclusive from a 'similarity metric' point of view. Leaving us with the only other option of visual similarity and reconstruction performance: the human eye.

Figure 9: Sample Label

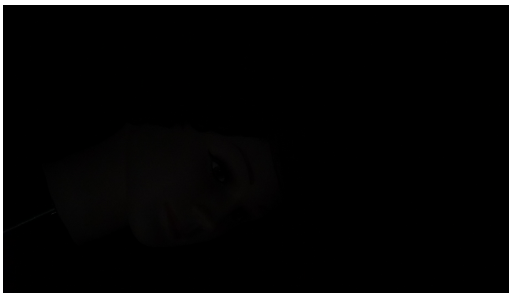

Figure 10: Corresponding Sample test image to be brightened

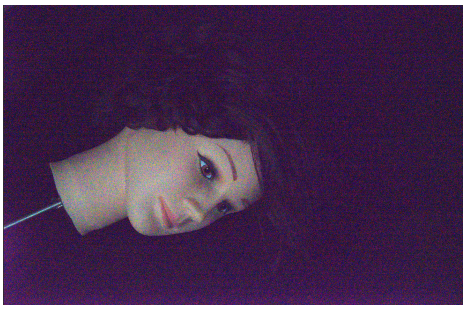

Figure 11: Windows built-in image enhancer

Figure 12: Their Result (Figure 4 in their paper)

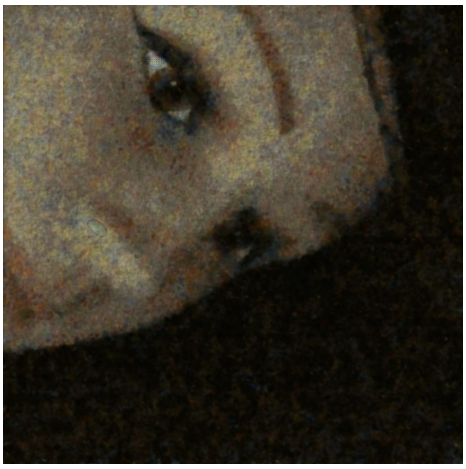

Figure 13: Our Result

On select images (see Figure 13), the performance is not as strong as the

authors, we do, however, achieve, an improvement in terms of lightening the image, when compared to the input, which seems completely black to the human eye. This suggests the network is indeed learning and achieving the goal set by the authors.

The network however does seem to perform much better on photos that were taken outdoors, as we can see below when compared to the figure above. This seems to suggest that the network seems to work well when there is some level of natural lighting, as opposed to almost pitch-black photos, such as the example above.

Figure 14: Another of our results, much better than the other

5 Conclusion

We successfully reproduced the results of the default pipeline of the paper. Even by training for only 100 epochs, we produced good results that are in line with the results of the paper. However some images still contain artifacts when bright lights are present in the image, however, this is most likely due to the fact the network learns a wider range of colors later on in training. An important note is that our reconstruction quality is purely based on human evaluation, as discussed in the section above. Due to the unreliable results reported by the two metrics used (PSNR and SSIM), our results are gauged by the authors as being acceptable. Since there concretely does not exist any metric for evaluating 'how well a photo is un-darkened', the human eye seems like an acceptable result. Note, however, that this is of course for only a tiny subset of images for which we checked the reconstruction. A more rigorous approach would of course go over all sorts of images, which may contain artefacts that we did not see, and which severely harm the networks ability to reconstruct images.

6 Discussion

Given the total training for 100 epochs (versus the 4000 epochs in the original paper), the results are acceptable on the test set (as seen in Figure 4). We could

have trained for more epochs and lowered the learning rate after a certain number of epochs like the original paper. This most likely will improve the results since the training loss still decreases. Another thing that we could have done is the flipping and translating of the patches. This might have decreased the bright light artifacts since the network will be able to see them in more diverse spots. Next to these improvements we could have experimented with the network, for example we could have trained the network on SSIM or different color channels, as well as with different architectures (such as the CAN architecture, as proposed by the original paper). However, due to the extensive time it takes to train the network on 100 epochs, re-training on multiple occasions would not have been feasible, especially considering the network seems to converge only after 90 epochs (which corresponds to roughly 8-9 hours of training). Alternatively, we could have attempted to fine tune the network hyperparameters, as well as the training itself, in order to see if the network would converge after fewer iterations. Despite being an interesting experiment, and a quantitative approach to fine-tuning, it is unlikely that any significant changes would have been discovered, since our thesis is that the network can only learn colors after an extensive training period. However, small changes could lead to large differences over the course of the whole training time, since the training time is in the magnitude of days, making this experiment a potentially interesting candidate for further research into seeing in the dark.

Another thing to note is, that we should have kept track of the test loss during training. This would have given a look into the generalisation of the network. Unfortunately we did not do this. It would have also been smart to keep track of the lowest loss per epoch, this would have given a better look into the loss decrease per epoch.

References

[1] Chen Chen, Qifeng Chen, Jia Xu, and Vladlen Koltun. Learning to see in the dark, 2018.

[2] Raanan Fattal, Dani Lischinski, and Michael Werman. Gradient domain high dynamic range compression. *ACM Trans. Graph.*, 21(3):249–256, jul 2002.