CS 7180 Milestone 2

Nalin Gupta NUID: 001497315
Christopher Botica NUID: 001726854
Tyler Brown NUID: 001684955

1 Introduction

Images taken in low-light conditions are often too dark, noisy, and distorted to be used in industrial purposes. We propose a deep-learning model that processes low-light images to improve image brightness and increase their overall quality. The problem with imaging in low-light conditions is challenging due to low-photon count and low Signal-to-Noise (SNR) ratio. These yield very dark and noisy images. The most common technique to overcome this problem is long exposure shot. However, this method yields blurry images with the slightest camera shake or object motion[1]. Common post-processing techniques brighten the image at the expense of image quality. Being able to "see in the dark" provides a number of real-world benefits such as photography, computer vision, and social networking.

2 Related Work

In the past, the problem of enhancing low light images has been tackled via noise reduction. This noise becomes dominant especially in low-light images due to low SNR. Remez et. al. proposed a deep CNN for noise reduction under the assumption that this low-light noise belongs to a Poisson distribution [2]. They used images from ImageNet [3] as their ground truth data and added synthetic Poisson noise to simulate corrupted images. Even though their model outperform the state-of-the art de-noiser "BM3D", it does not scale well to real world images, due to their underlying assumptions. Furthermore, their model only denoises images but does not brighten them. Motivated by these downfalls, Chen et. al., proposed an end-to-end CNN, "See-in-the-Dark" (SID), which brightens extremely low light images and removes noise without making any underlying assumptions [1]. However these advances come with the added expense of collecting large amounts of low light and bright light images. In the absence of a true vs noisy image dataset, the team captured scenes using various exposure times to generate true (bright light) and corrupted (low light) image pairs called "See-in-the-Dark Dataset" (SID Dataset [1]). Furthermore, their model is camera specific and not easily generalizable.

We propose a transferable CNN for image brightening and denoising. Instead of training our model on actual true (bright light) and corrupted (low light) image pairs, we use images from the publicly available RAISE raw images dataset [4]. We use the RAISE raw images dataset as our true images and corrupt these by simulating low-light conditions for use in our model. We train our CNN on this synthetic data to obtain our initial model parameters. A small fraction of the real images paired from the SID dataset is then used in our transfer learning [5] approach to update our model parameters. These model parameters are updated for the particular camera used in the SID dataset. We then use this model to test on the SID Dataset. In addition, we aim to test various transfer learning approaches, such as the traditional transfer learning and zero shot learning [6, 7, 8].

The novelty of our approach stems from the idea of "more for less". Our model drastically reduces the overhead costs of data collection by synthesizing readily available training data (RAISE raw images

[1] https://github.com/cchen156/Learning-to-See-in-the-Dark

32nd Conference on Neural Information Processing Systems (NeurIPS 2018), Montréal, Canada.

dataset). This is particularly beneficial in domains where collecting images pairs is expensive/time consuming.

3 Methods

Our contributions were to simulate the data and replicate their model using simulated data.

3.1 Dataset

We plan to use the RAISE raw images dataset which includes 6,000 high definition images taken by professional photographers. Since these are raw images, they have not been touched or processed in any way. This gives us a true representation of the scene.

3.2 Model

Our model is based on the one developed for SID, with the addition of 2 fully-connected layers. Note that we may increase this if training runtime permits. Using the transfer learning approach, we will first train our model on the RAISE dataset, then erase the learned weights from the last two fully-connected layers, and retrain on the SID dataset. An example of a training pair (image taken in low-light and normal light conditions) is shown in Figure 5.This is highlighted in our model framework in Figure 2.

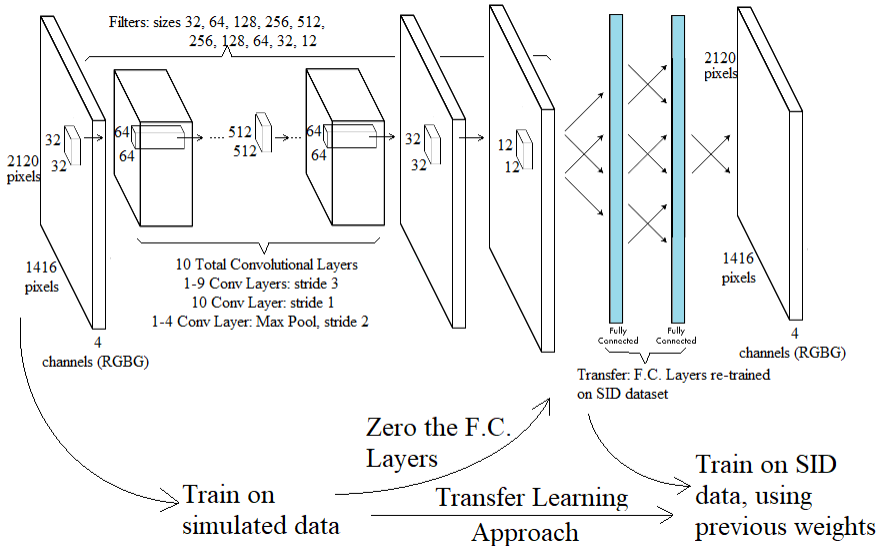


Figure 1: Our proposed model framework

We tried to replicate their model to using other readily available data such as CIFAR10 [9] and ImageNet [3]. The CIFAR10 ad ImageNet data is augmented to simulate properties of the images used by Chen et. al. (2018) [1]. The CIFAR10 and ImageNet datasets are distinctly different because they are not *raw* images. This means the dimensionality of CIFAR10 is $32 \times 32 \times 3$ rather than a raw image which could be $512 \times 512 \times 4$. We not only need to simulate images but also account for the *change in dimensionality* across image types.

As illustrated in Figure 1, the traditional pipeline takes a corrupted image, and applies the following sequence of modules: Reduce Black Level, Denoising, White Balance, and Gamma Correction. The Black Level refers to refers to the level of brightness at the darkest parts of the image, and is reduced by subtracting the minimum pixel value. Denoising is reduced using common algorithms such as BM3D. White Balance refers to the color balance in the image (i.e., white should be true white) and is corrected by re-balancing the intensities of each color RGB. Finally, Gamma Correction controls the overall brightness of the image. We synthetically generate corrupted images by applying the reverse of this pipeline. Gamma Distortion: decrease the brightness of the image, White Imbalance: skew the

color-space by multiplying each level of RGB by a random weight, Poisson Noise: add Poisson noise to the image, Black Level: add a negative bias to the pixel values (i.e., random black level).

Our model is based on the one developed for SID, with the addition of 2 fully-connected layers. Note that we may increase this if training runtime permits. Using the transfer learning approach, we will first train our model on the MIT dataset, then erase the learned weights from the last two fully-connected layers, and retrain on the SID dataset. This is highlighted in our model framework in Figure 2.

We tried to replicate their model to using other readily available data such as CIFAR10 [9] and ImageNet [3]. The CIFAR10 ad ImageNet data is augmented to simulate properties of the images used by Chen et. al. (2018) [1]. The CIFAR10 and ImageNet datasets are distinctly different because they are not *raw* images. This means the dimensionality of CIFAR10 is $32 \times 32 \times 3$ rather than a raw image which could be $512 \times 512 \times 4$. We not only need to simulate images but also account for the *change in dimensionality* across image types.

3.3 Computational Resources

Using AWSEducate did not work for us. We were unable to create roles with IAM authentication so it's really hard or impossible to move data from a S3 Bucket to an EC2 Instance. We tried to create a $p3.8xlarge$ instance but these instances are not allowed even though they are listed. Using regular AWS does work but is costly. Ran a single AWS EC2 $p3.8xlarge$ instance with 32 CPU, 244 GB of Memory, 4 Tesla V100 GPUs, and 64 GPU Memory. This costs $12.24 an hour. This is the amount of GPU Memory requested by the paper authors as a minimum amount.

We have also been able to use Google Cloud for some of our workload. This resource is well integrated with Tensorflow. Leading up to Milestone 2, we extensively used Google Cloud Compute with 2 CPU and 7.5GB of memory for debugging and working with image datasets; primarily CIFAR10 and Tiny ImageNet.

In regards to training our model, one of our group members has also received permission to use computational resources from Lincoln Laboratory's Super Computer (LLSC) for our project. We are currently waiting for complete access and expect to be able to use it in the coming days.

3.4 Low Light Simulation

As illustrated in Figure 1, the traditional pipeline takes a corrupted image, and applies the following sequence of modules: Reduce Black Level, Denoising, White Balance, and Gamma Correction. The Black Level refers to refers to the level of brightness at the darkest parts of the image, and is reduced by subtracting the minimum pixel value. Denoising is reduced using common algorithms such as BM3D. White Balance refers to the color balance in the image (i.e., white should be true white) and is corrected by re-balancing the intensities of each color RGB. Finally, Gamma Correction controls the overall brightness of the image. We synthetically generate corrupted images by applying the reverse of this pipeline. Gamma Distortion: decrease the brightness of the image, White Imbalance: skew the color-space by multiplying each level of RGB by a random weight, Poisson Noise: add Poisson noise to the image, Black Level: add a negative bias to the pixel values (i.e., random black level).

We simulate our low light images by successively applying gamma distortion, white imbalance, Poisson noise, and black level to a normal lighted. An example of two of such images is represented in Figure 3.

For a given bright image, B, a matrix of size mxnx3, we want to simulate a low-light image D, or the same. We define gamma distortion as follows:

$$D = B^{1/\gamma},$$

where γ controls the brightness of the image and the exponentiation is computed element-wise.

Furthermore, for the three channels of B, namely B_R, B_G, and B_B, and the three channels of D, namely D_R, D_G, and D_B, respectively, we define color distortion as follows:

$$D_R = w_R * B_R,$$

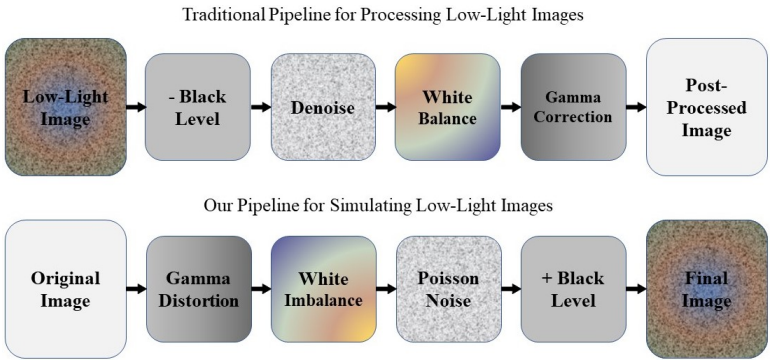


Figure 2: Top: Traditional Pipeline for processing low-light images. Bottom: Our pipeline to simulate low-light images based on the traditional, only in reverse.

$$D_G = w_G * B_G,$$
$$D_B = w_B * B_B,$$

where w_R, w_G, w_B are small weights that distort the color space.

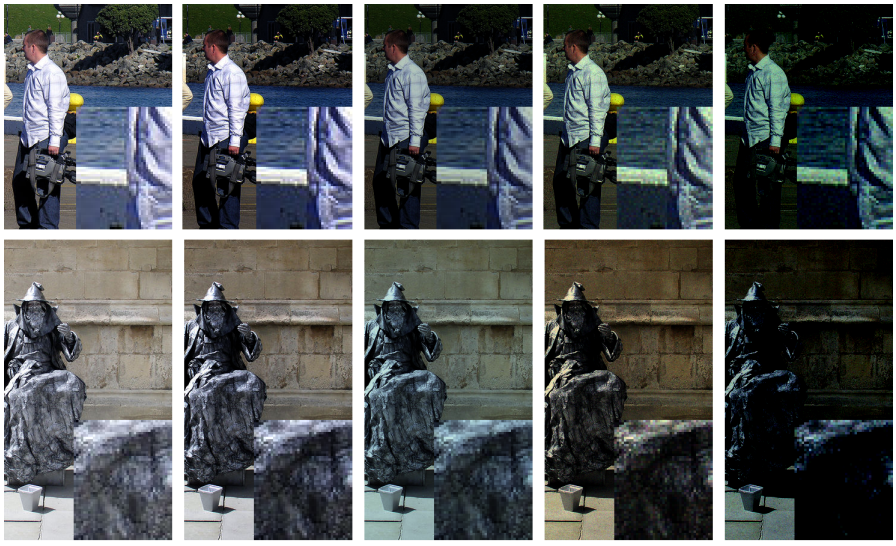
We define the black level as follows:

$$D = max(B - bl, 0),$$

where bl represents the amount of black level added to the image. Note that the closer the pixel values are to 0, the darker they are. The maximum of $B - bl$ and 0 is taken so that no pixels are negative.

Finally, we add Poison noise. Since this outputs values between $[0, 1]$, we first scale our image, apply Poison, and then re-scale as follows:

$$D/255 = Poisson(B/255).$$



(a) (b) (c) (d) (e)

Figure 3: Simulating two low-light images via our pipeline: (a) Gamma Distortion, (b) White Imbalance, (c) Poisson Noise addition, (d) Black Level, (e) Final low-light simulation

Our initial approach to creating simulated low light images was to randomly select values for γ, w_R, w_G, w_B, and bl was to choose these randomly and validate them by visualizing the distorted image. However, this process is not good enough to yield accurate results. The purpose of the

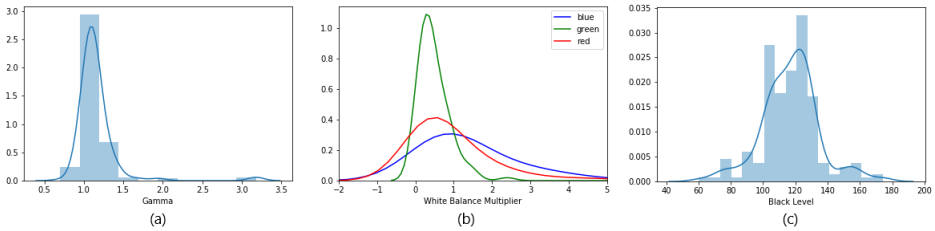
simulation step is to simulate low light images the best we can, in order to then provide a well-trained model for transfer. Therefore, we assume values for these hyper-parameters for natural low-light images belong to certain distributions. In the next step, we aim to approximate these distributions.

Let $b_{i,j,k}$ be a pixel value from B and $d_{i,j,k}$ be a pixel value from D. We approximate γ finding the value such, when gamma-distortion is applied to the bright image, the average pixel value equals that to the darker image:

$$\frac{1}{3mn}\sum_{i,j,k} b_{i,j,k}^{1/\gamma} = \frac{1}{3mn}\sum_{i,j,k} d_{i,j,k}$$

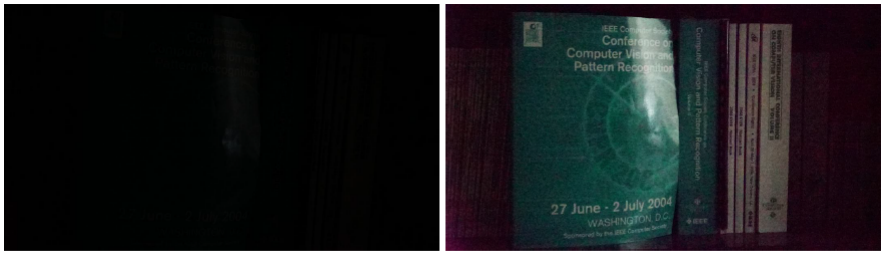
Similarly, for a given pair B and D, we find the optimal values for w_R, w_G, w_B, and bl, such that, after applying color-distortion, and adding black-level, respectively, the average pixel value for B equals that of D.

We repeat this process for 200 random pairs B and D from SID and compute the distribution for each value of γ, w_R, w_G, w_B, and bl. These distributions are exemplified in Figure 4. Our assumption is that, by sampling these values from their respective distribution, we can create more accurate low-light images. We find these distributions by computing the Kernel Density Estimation (KDE) and then randomly sample.



(a) (b) (c)

Figure 4: Distributions for (a) gamma, (b) RGB weights, and (c) black level.



(a) (b)

Figure 5: SID Images in Normal and Low Light Conditions

4 Experiment

Our experiments consist of recreating the see-in-the-dark model and getting results from our own model. We also compare results and provide insight into the model implementation details.

4.1 Model Implementation

We took inspiration from the SID model and wrote our own model which had similar functionality. The SID model is mostly written in a higher level Tensorflow contribution module called "slim". The "slim" contribution module does not appear to have any documentation and requires one to read the module code embedded within Tensorflow. We initially tried writing a model in parallel using

Keras but found that moving from Keras objects back down to lower level Tensorflow for some blocks within forward propagation to not be documented well. We then rewrote the Chen et. al. [1] model using only low level Tensorflow code. This approach has the disadvantage of requiring the dimensions of all weight matrices to be specified and initialized in a separate function we called 'initialize_parameters()'.

These efforts allowed us to understand the dimensionality of both models at each of the 10 blocks within their forward propagation. We used these models to experiment with pooling window size, and stride length for the convolutional layers. We also added up to three additional blocks with different parameters in an attempt to get the output dimensions to match the dimensionality of our input image.

We also tried using the CIFAR10 and ImageNet datasets as input. The primary issue when replicating their model was related to dimensionality. The RGB images in CIFAR10 and ImageNet have $m \times n \times 3$ dimensions. However, the SID model takes as input Raw images which have $m \times n \times 4$ dimensions. We were able to confirm that output dimensions from Chen et. al. [1] follow the pattern $2m \times 2n \times 3$ irrespective of the image size or type used. During our experiment, we focused our debugging efforts on understanding whether the dimensionality of a raw image as an input generated incompatible dimensions with the target vector after model output. This error resulted in our replicated model not being able to compute the cost function.

Currently, we are working with raw images as input which have been corrupted using our low light simulation; mentioned in Section 3.4. The results for our model are shown in Section 4.3. At this point, we have not been able to use RGB images, having $m \times n \times 3$ dimensions, to work with our model. It is important for us to develop this capability in order to qualitatively verify that we are correctly augmenting image data. We are unable to view simulated images in raw format because it requires us to know proprietary information about how a camera generated an image. In order to generate output for this milestone, we modified images from Chen et. al. [1], such that

$$X = g(y_{train})$$
$$Y = y_{train}$$

where function g is an application of artificial noise to a clean image in y_{train} which is then trained against a corresponding image where g has not been applied.

4.2 State of the Art Model Results

The model used by Chen et. al. [1] trains on raw images for 4,000 epochs. It beats the current state of the art denoising technique, BM3D. An example of a result for this model is shown in Figure 6. A drawback of this model is that it learns a separate model for each device used to capture the raw images.

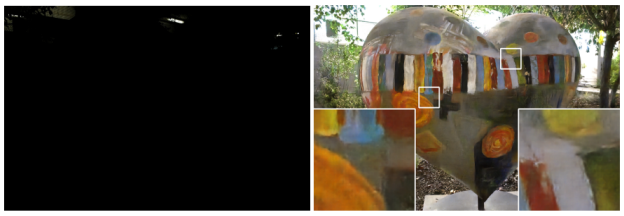


Figure 6: State of the Art Model Results

We can see in Figure 6, a nearly dark image in sub-figure (a) is converted to an image shot in normal light conditions; shown in sub-figure (b).

4.3 Our Model Results

Currently, our model has been trained on a subset of simulated lowlight images using the SID dataset as mentioned in Section 4.1. We ran the model for 25 epochs due to limited resources. However, we were still able to obtain images which showed some enhancement. We are able to show some initial results but they are not of the same quality compared to state-of-the-art. [1].

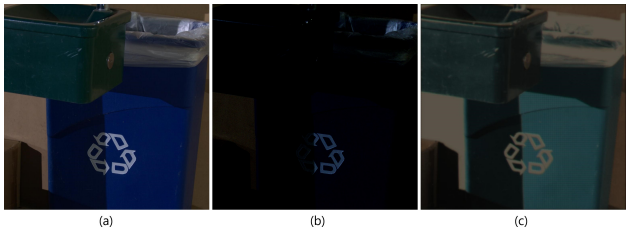


Figure 7: Predicted Results

Figure 7 shows the steps followed by our approach. Sub-figure (a) is the original image, sub-figure (b) is the simulated low light image described in Section 3.4, and sub-figure (c) shows the result of our model. We can clearly observe from Figure 7 that our model was able to identify the colors while simultaneously brightening the image.

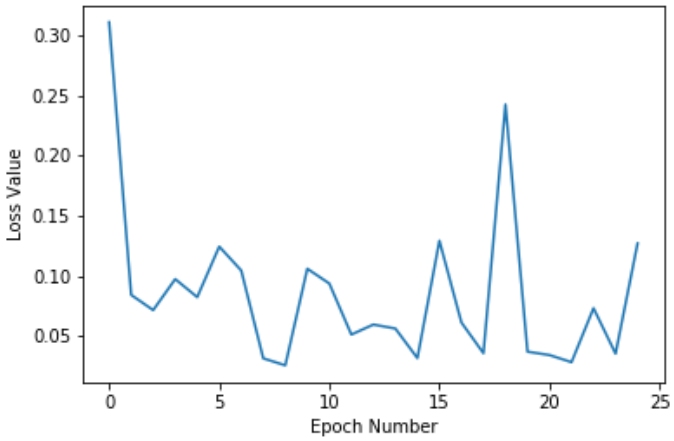


Figure 8: Loss vs. Epochs

Figure 8 plots our loss function at each epoch. We can see a downward trend in the loss function output. However, we can also see that our model is having trouble converging after 25 epochs. We believe this problem can be solved when we use the entire dataset to train the model.

5 Discussion

Currently, we are simulating low light images using (Section 3.4) using raw images from the SID dataset. A drawback of this approach is that we can modify the pixels of the image to simulate low light but we cannot view these modifications as a RGB image. Our proposed next step is to modify our model is so it is compatible with RGB images as input. We have identified blocks within the model which most likely need to be changed. After updating our model to accept RGB input, we will be implementing transfer learning to obtain the final result.

References

[1] Chen Chen, Qifeng Chen, Jia Xu, and Vladlen Koltun. Learning to see in the dark. *arXiv preprint arXiv:1805.01934*, 2018.

[2] Tal Remez, Or Litany, Raja Giryes, and Alex M Bronstein. Deep convolutional denoising of low-light images. *arXiv preprint arXiv:1701.01687*, 2017.

[3] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei. ImageNet: A Large-Scale Hierarchical Image Database. In *CVPR09*, 2009.

[4] Duc-Tien Dang-Nguyen, Cecilia Pasquini, Valentina Conotter, and Giulia Boato. Raise: A raw images dataset for digital image forensics. In *Proceedings of the 6th ACM Multimedia Systems Conference*, MMSys '15, pages 219–224, New York, NY, USA, 2015. ACM.

[5] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT Press, 2016. `http://www.deeplearningbook.org`.

[6] Hugo Larochelle and Y Bengio. Classification using discriminative restricted boltzmann machines. pages 536–543, 01 2008.

[7] Mark Palatucci, Dean Pomerleau, Geoffrey E Hinton, and Tom M Mitchell. Zero-shot learning with semantic output codes. In Y. Bengio, D. Schuurmans, J. D. Lafferty, C. K. I. Williams, and A. Culotta, editors, *Advances in Neural Information Processing Systems 22*, pages 1410–1418. Curran Associates, Inc., 2009.

[8] Richard Socher, Milind Ganjoo, Hamsa Sridhar, Osbert Bastani, Christopher D. Manning, and Andrew Y. Ng. Zero-shot learning through cross-modal transfer, 2013.

[9] Alex Krizhevsky, Vinod Nair, and Geoffrey Hinton. Cifar-10 (canadian institute for advanced research).