Image Enhancement for Unconstrained Environments

by

Sougato Bagchi

Aug 7th, 2023

A thesis submitted to the
faculty of the Graduate School of
the University at Buffalo, The State University of New York
in partial fulfillment of the requirements for the
degree of

Master of Science
Department of Computer Science & Engineering

Acknowledgements

I want to start by expressing my gratitude to Professor Nalini Ratha for his guidance throughout the past year. He provided me with the opportunity and encouragement I needed to pursue the intriguing problems I was interested in, and he also inspired me to persevere through the challenges. I also thank Professor Karthik Dantu for giving me valuable suggestions and academic support thought my entire time at UB.

Being a member of the Distributed Robotics and Networked Embedded Systems group has been incredibly fulfilling. I want to thank everyone for the time we shared and for our discussions. I will always be grateful for Ninad's invaluable suggestions and contributions. I would like to thank Shaoshu for the time we shared together while working and for his technical assistance whenever I needed it.

I would like to express my gratitude and acknowledgement to my friends especially Vivek, Vignesh, Sharath and Anusha who have supported me throughout the completion of my thesis with their encouragement, support, and the dinner parties.

Most of all, I am grateful to my family and my parents and my uncle for showing me an interest in engineering and guiding me to be a better person.

Abstract

In this research we have discussed how modern techniques use Deep Neural Network for processing low-light images. It involves de-noising and exposure correction which greatly improves the usability of these images.

De-Noising and exposure-correction both are well studied in the field of computer vision. Significant progress has been made by researchers to handle both of the problems separately. The ultimate goal of our work is to combine these into a single problem as most realistic low-light images inherently come with noise. Traditionally tailor-made solutions like custom de-noising filters were used to image enhancement. Recent applications have replaced these with DNN models due to their generalizability in different noise and exposure scenarios. But these models need a sheer amount of realistic data for training, which is an issue as it requires human labor and it's infeasible to capture images that model all types of irregularities.

Therefore, our discussions extend towards investigating ways for generating realistic noisy low-light images considering both denoising and exposure-correction factors, which can be used for training any suitable DNN model. The data generation process requires a detailed understanding of the noise architecture with a digital camera's image acquisition process. Our experimentations conclude that irrespective of the DNN model being used the performance gets better when trained on our synthetic noisy dataset. We have utilized an already existing SOTA (state of the art) DNN model named LLFLOW [1], and its generalizability in recovering the images has been greatly improved in terms of metrices like PNSR(Peak Signal to Noise Ratio), SSIM [42] and LPIPS [15]. It's also able to avoid over-exposure circumstances during our testing which is

clearly visible for our lab images. We have also tested the model on EarthCam images of New York City between June 6th & 8th, when the city was clouded with Canadian Wildfire Smoke. Though our model was not provided with that kind of data for training, it performed fairly well in which it's supposed to i.e., improving the illumination of the images. This proves the immense importance of data on which we train our models.

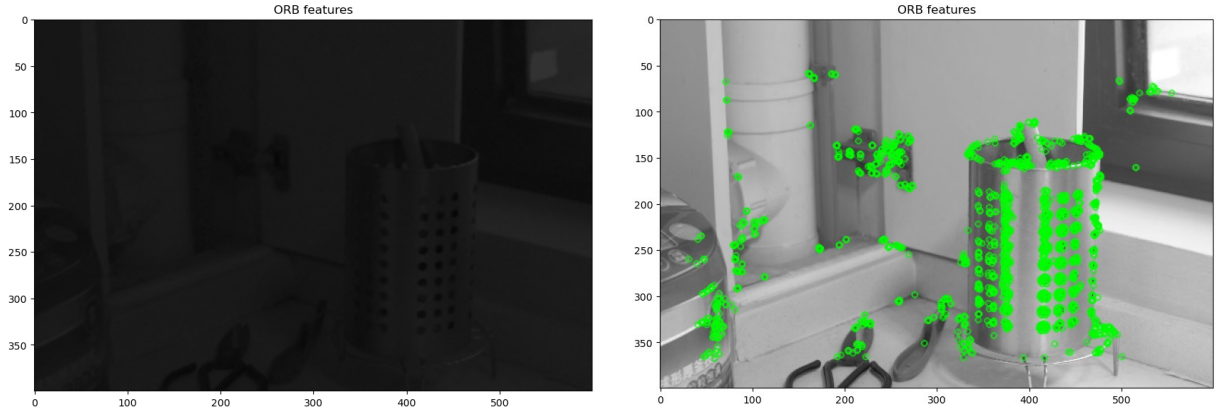
Table of Contents

List of Tables

List of Figures

Chapter 1 Introduction

Images captured in low-light circumstances are always difficult for a human to perceive. Images are required in many applications such as surveillance, autonomous driving, vision-based mapping, and others. Some of these applications need to be robust in all weather conditions, and for that these will require good-quality images. And as low-light images are always difficult to infer, we need to modify these images for better visibility. Many computer-vision related applications are specifically tasked to enhance these low-light images.



(a) No feature detected in low light conditions (b) Features detected in bright light conditions

Figure 1.1: Comparison of the ORB features detected for a same scene under different circumstances.

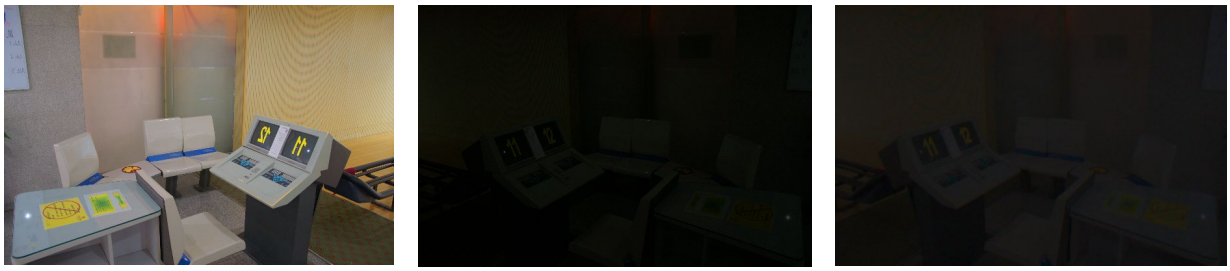
The low light image hampers feature detection, these images are tough to infer and lead to severe degradation of a particular application's performance. Image src: LOL dataset, with the low light image being degraded using our custom process described later.

Typically, a digital image is a collection of pixels with different values ranging from 0-255. For a low light image all the pixels collectively have very low intensities, and therefore the difference between neighboring pixels is also low which makes it look like every part of the image

is similar and simply dark. So, our objective is to increase the relative difference in the values for these pixels, which leads to the process of low-light image enhancement.

One of the best ways to deal with low-image quality problems is to train a Deep Learning model for creating a map [1] between low and high-quality image features. These models typically need image pairs with one being the undesired one and the other a good-quality image for the same scene as shown in Figure 1.2(a-b, a-c). Chen, et al. [2] have developed a paired dataset by reducing the exposure of bright images (Figure 1.2, a) to create poorly exposed images (Figure 1.2, b).

Real low-light images have noise in them due to several factors and the approach mentioned above doesn't model that well and so we need to come up with solutions which can create realistic noisy images in a better way. Another approach could be instead of generating synthetic images use the same camera for capturing bright and low-light image pairs targeted by some denoising/enhancement algorithm [3], [4], [5].



(a) Bright Image (b) Image captured using low exposure time (c) Image b after adding noise

Figure 1.2: (a) & (b) represent the image pairs generated in traditional way without considering the noise factor. Image (c) is our custom generated image which will be used instead of (b) to create the modified pair.

The latter method might prove to be a difficult implementation because of the tremendous human effort needed. It also requires long exposure shots for capturing noise-free images where some of the parameters might change, this may lead to a discrepancy in the noisy & noise-free image pairs for the same scene. The viable solution which might be scalable is developing better ways of

modeling noise on the generated low-light images by modifying the exposure of the bright images from different datasets, and then train models which leads us in better cross-dataset results. Here our focus was to first create an appropriate dataset by utilizing the low-light images from the LOL dataset. We have inverted all the image acquisition steps from the sRGB format to the RAW equivalent and then added noise to match the real-world scenarios. Then we again converted those noisy RAW images into their sRGB formats to get our desired low-light noisy images. These images were paired with the bright images from the dataset for training a neural network which in our case is a Super-Resolution based GAN named LLFLOW [1]. A detailed overview of the image acquisition process has been discussed in the **Image Acquisition Pipeline** which is required to understand our research methodology.

Chapter 2 Related Works

2.1. Noise Modeling

Due to differences in sensor size, sensor type, and other aspects of the imaging pipeline, different cameras emit distinct types of noise. Moreover, in low-light situations, image noise follows the Poisson-Gaussian pattern instead of the simple Gaussian pattern [5]. So, there are two ways of creating a noisy-clean image pair dataset i.e., one is to create synthetic noisy images by artificially providing the required noise on the clean images, and the other method is to capture the image pairs by changing the parameters like light sensitivity (ISO) and exposure time of a particular camera. The latter method models the noise more accurately than the other method as there are real images.

Anaya et al. [6] have implemented the latter method using different camera/image sensor systems but this method is tedious and may not model the noise signature of an unknown sensor on which the dataset was not created. Implementation of the learning-based denoising methods is becoming increasingly popular in the research community [7] over traditional models like BM3D [8] & WNNM [9], as they can learn the infinitely possible noise signatures given that we can feed enough data. For this, it's not feasible to follow the latter method, instead, we have to find ways to develop synthetic noisy images.

One of the ways of creating noisy synthetic images as proposed by Brooks, et al. [10] is to take the final image from a particular system and then undo all of the image processing steps which typically take place in a camera system, illustrated in Fig 3. This results in getting synthetic RAW

images and changing the parameters, adding noise to these generates RAW noisy images. Also, they have pointed out from their experimental results that a neural network trained on these reverse-engineered raw image pairs works better than the one trained on the processed/final image pairs.

2.2. Image Enhancement

The Retinex model is an effective method utilized for lowlight image enhancement. This states that an image can be decomposed into two components, reflectance & illumination.
This can be denoted as

$$S = R \circ I \tag{2.1}$$

where S is the source image, R is the reflectance, and I is the illumination. Here ◦ denotes the mathematical operation of element-wise multiplication. Many traditional low-light image enhancement methods have utilized this theory. This theory has been the inspiration for many of the deep learning methods created for this task. A Progressive Retinex framework that trains the illumination and reflection maps in a mutually reinforcing manner is proposed by Wang et al. 2019b [11]. This model has also inspired the framework proposed by Yufei, et al. [1] to extract the illumination invariant color map that takes care of the color saturation and distortions as the prior for the low-light image enhancement task. But for better illumination adjustments and noise & artifact suppression they have utilized a conditional normalizing flow based on Normalizing Flow [12], which is the transformation of a simple probability distribution (e.g., a standard normal) into a more complex distribution.

Most of the neural networks for denoising also need a huge amount of image pairs. But in this field, not many attempts have been made but the one made by Wei, Chen, et al. [2] collected RAW images from the RAISE [13] dataset and matched its Y channel with that of the real low-light images collected from different datasets.

2.3. Brief overview of GAN

With the introduction of Generative Adversarial Networks by Goodfellow et al. [14] we have seen a whole new field of applications. These networks do have the ability to enhance the input data from what they have learned previously, given that it finds some similarity between its input and the learned features. Due to this unique ability, it's one of the most suitable networks for our purpose of image enhancement and denoising.

The implementation of Adversarial Modeling is made on two layers of multilayer perceptron.

The first one being the generator whose distribution p_g over data x being initially defined as a prior over input noise variable $p_z(z)$ and i.e., represented to the dataspace using $G(z;\theta_g)$ where G is the differentiable function with parameters θ_g.

The second multilayer perceptron is called the discriminator $D(x;\theta_d)$, which has a single scalar as output. Here its task is to find the probability that x came from the original data rather than from the 1st perceptron p_g.

Therefore, both of these networks are being trained with different motives. D is trained to maximize the likelihood that training examples and samples/outputs from G being assigned the

correct label. In addition, we train G to reduce the distance between its output and the input data, mathematically its task is to minimize $\log\left(1 - D\big(G(z)\big)\right)$.

So, we can call these two modules taking active part in a two-player minimax game with the value function being $V(G, D)$, defined as:

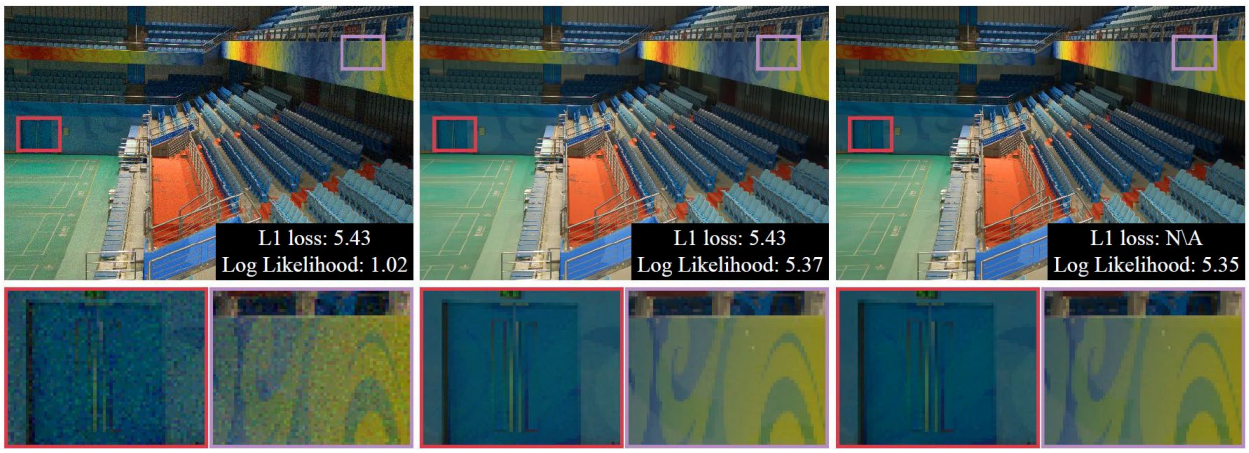
$$\min_G \max_D V(D, G) = \mathbb{E}_{x \sim p_{data}(x)}[\log D(x)] + \mathbb{E}_{z \sim p_z(z)}\left[\log\left(1 - D\big(G(z)\big)\right)\right] \tag{2.2}$$

2.4. LLFLOW

The primary goal of Image enhancement task is to improve the visibility of low-light images, suppress the source noise and visual artifacts. GANs are one of the most widely used networks for this purpose for their inherent design. The performance of G depends on how well $\log\left(1 - D\big(G(z)\big)\right)$ is reduced. As this is a mathematical distance, the method implemented for calculating the distance greatly affects the performance. Utilizing a pixel-wise loss function for the Deep learning methods by Wang, Yang, et al. 2019 [11], Chen et al. 2018 [3] have achieved promising results, but they also subject to some serious issues.

First, a low-light image may correspond to several reference images with different exposure. Mapping the low-lit pixels and the fusion of bright pixels from these different reference images may result in improper exposure and visual artifacts. Second, considering pixel-wise loss as the primary factor may lose contextual information as features are not preserved instead the pixel values are modified individually. This may lead to deterioration in visual perception quality.

Second, for a particular region in an image, pixel-wise loss may fail to differentiate between the reference image and the enhanced image if the mean value turns out to be similar. Illustration in Figure 2.1.



(a) noisy image **(b) image with slight brightness variance** **(c) reference image**

Figure 2.1: Here although image (b) is more similar with (c) which is the reference image, but if we are supposed to consider only L1 loss/ pixel-wise loss as the factor then it is likely to fail in recognizing the difference between image (a) & (b) both of which has the same L1 loss. Source: [1]

The authors of LLFlow have addressed these issues. They have avoided L1 loss and instead used the NLL loss and for preserving the perceptual similarity as well as structural similarity they have added the LPIPS (Learned Perceptual Image Patch Similarity) [15] metric with the traditional PSNR and SSIM. They have also structured their algorithm/model according to the reinforced Retinex Theory [11] which incorporates the noise factor. All these make this model robust and perfect for our use case.

2.4.1. Overview of LLFlow Architecture

We know that the Retinex Theory states that an image can be disintegrated into its reflectance and illumination components [2], and this is the holy grain for LLFlow' s image

enhancement. The theory's equation (2.1) has further been reinforced by Y. Wang, et. al. [11] by incorporating the factor of noise. This greatly improves the robustness of this theory.

$$S = R \circ I + n \tag{2.3}$$

Here n denotes noise in the image and LLFlow has taken care of these 3 factors using an Enhanced Super Resolution GAN (SRGAN) network for training named as Residual-in-Residual Dense Block (RRDB) [16]. The NLL loss for LLFlow has been optimized in such a way that it takes care of the components of equ. (2.3).

For the "I" factor, images are histogram equalized which increases global contrast by ensuring the values stay in the range of [0,1]. This reduces the difference between x_l and x_{ref} and ensures that the images stay illumination invariant for the model Θ.

For the "R" factor which aims to minimize the difference due to different color intensities, a color map $C(x)$ is calculated. $C(x)$ is a ratio which makes it easier for the model Θ to compare x_l and x_{ref} as instead of comparing them directly $g(C(x_l))$ and $C(x_{ref})$ ratios are compared, therefore generating better x_h which is more similar to x_{ref}.

$$C(x) = \frac{x}{\text{mean}_c(x)} \tag{2.4}$$

Here mean_c calculates the mean value of each pixel among the RGB channels. For taking care of n or noise factor in equation (2.3), a noise map is created and fed to the encoder of the generator g. The noise map $N(x_l)$ is estimated as follows:

$$N(x) = \max\left(abs\left(\nabla_x C(x)\right), abs\left(\nabla_y C(x)\right)\right) \tag{2.5}$$

Here, ∇_x & ∇_y denotes the gradient maps in the x & y directions and max (x, y) function returns the max value between x & y at pixel channel level.

Chapter 3 Image Acquisition Pipeline

As discussed in the previous sections, the efficient way of gathering large amount image data is by generating synthetic images as it does not need much human labor. The only limitation which we might face is how good we are at replicating the real-life noisy low-quality images. And to improve our ways to generate synthetic image we need to understand what goes on behind the scenes of an image sensor. We also need to understand the type of noise and the stage where it gets associated in.

Noise gets associated mostly in the first step of the image processing pipeline. This pipeline consists of various steps which are mentioned Figure 3.1.

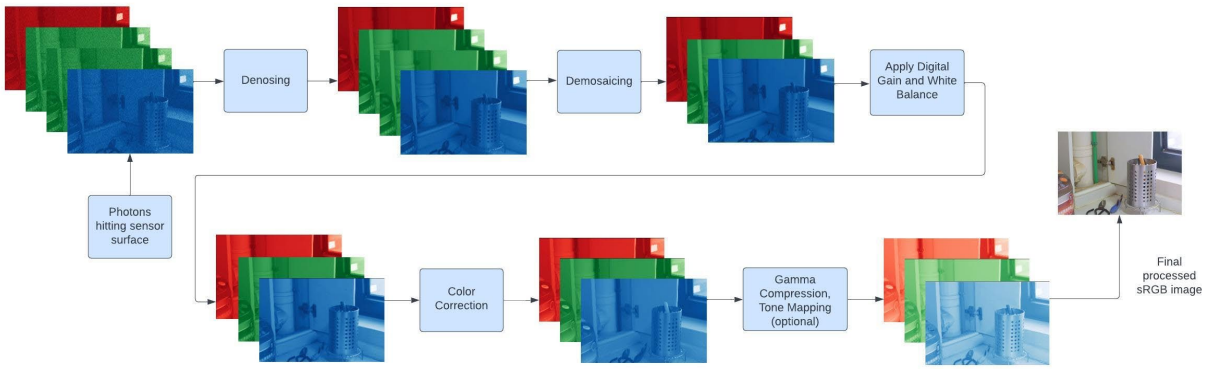


Figure 3.1: Image Acquisition Pipeline

3.1. Noise In an Image

Noise is an inherent property of the image sensors and it's generally dominated by photon noise and sensor noise [17]. Every image is result of the pattern formed from the photoelectric effect on the image sensors, and the uncertainty in photons hitting the surface of the image sensor leads to photon noise. The other kind of noise is due to the properties of the image sensor which is dependent on the physical characteristics of the sensor. There is also another kind of noise called the dark current noise which is caused by the release of photons from the sensor itself due to heat. In lowlight conditions, the sensors are unable to capture detail due to the lack of photons and in these situations, the sensor noise becomes a dominant force in the image distortion.

3.1.1 Modeling Noise

In the field of computer vision noise in an image is often modeled using zero-mean additive Gaussian distributions, which is quite often signal independent. So, in simple application this kind of approximation does work but it's mostly unrealistic. In real image capturing systems Shot Noise (photon noise), and sensor-based noise contributes in varying proportions to the final RAW image.

This means that the noise pattern is dependent on the scene brightness i.e., number of photons collected in a particular sensor pixel, and also some sensor-based variables like the sensor size, number of pixels in the sensor, irregularities in reading the values due to circuit characteristics.

3.1.2. Shot/Photon Noise

This type of noise is associated with the uncertainty in the measurement of light, which is inherent to the quantized nature of light. Capturing an image is the result of photelectric effect which occurs in the image sensor, therefore this effect depends on the randomness in photon

detections and can be treated as independent events following a random temporal distribution. As a result, we can call the photon counting problem following a Poisson distribution with the distribution being described below in equ 3.1.

$$Pr(N = k) = \frac{e^{-\lambda t}(\lambda t)^k}{k!} \tag{3.1}$$

Here this is a discrete probability distribution with N being the no. of photons which is measured on a sensor over time interval t, λ being the expected no. of photons per unit interval of time, therefore,

$$\lambda \propto H \tag{3.2}$$

where (H) is irradiance and it is measured as power per unit area, λt corresponds to the expected photon count and can be called as the signal. As λt or photon count follows Poisson distribution, its variance is equal to its expectation.

$$E[N] = Var[N] = \lambda t \tag{3.3}$$

or

$$\mu = \sigma^2 = signal \tag{3.4}$$

Equ 3.3 and 3.4 shows that this type of noise is signal dependent and its standard deviation (σ) is proportional to the $\sqrt{(\text{signal})}$.

3.1.3. Gaussian Approximation of Shot/Photon Noise

As the gaussian distribution of this noise is typically accurate, its widely used. Due to equ. 3.4 we can say that if we have weak signal or if the photon count is small, the photon noise will also be less and the overall noise will be dominated by other signal-independent noise like the readout/sensor noise or the dark current noise.

But while assuming the distribution of photon noise we write the distribution as a Normal distribution with the mean and variance being the same with the value being equal to λt.

$$N \sim \mathcal{N}(\lambda t, \lambda t) \tag{3.5}$$

As per equ 3.3 and 3.4 we see that Photon/shot noise is proportional to the $\sqrt{(\text{signal})}$, that means though this noise increases in absolute term and its absolute ratio is high when the signal level is low, and it gets weak when we have higher signal level. So, the remedy behind reducing this noise is to capture images with more photons and we can do that by increasing the exposure time of the sensor capturing light. But also, there is a limitation in the number of photons that can be captured by the sensor for a single shot and that is dependent on the type of image sensor that we use.

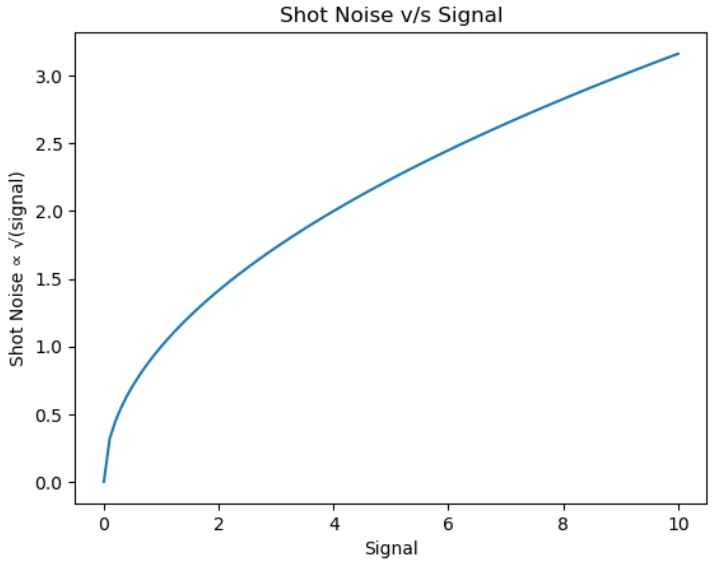


Figure 3.2: Characteristics curve of shot noise. The square root relationship signifies that in low-light circumstances this noise will dominate cumulative noise pattern.

3.1.4. Noise due to sensor characteristic

There are several types of noise associated with an image sensor, two of them have been described here. The first type is due to the internal readout-circuitry, which means that every sensor has irregularities going on and that is typically specific for each and every sensor. This is mostly known to the manufacturer and dealt with by them, so these values are mostly proprietary.

Everything which has mass releases or absorbs heat, so while we use these sensors these do heat up and releases photons and this is the cause for the second type of noise related to the sensors. This noise is called the dark current noise and usually its value is insignificant compared to the other noise. There are some scenarios where this noise becomes a significant player and i.e., when the signal is too like when we are capturing the image of something which is too far away and the emitted light reaching the sensor is extremely limited. Examples like telescopes capturing celestial images like the Milky Way Galaxy, in these scenarios to avoid dark current noise the sensor is kept in a cool environment with temperature being close to 0 Kelvin.

3.2. Demosaicing

An image sensor consists of millions of individual pixels with each being sensitive to either one of the Red, Green, or Blue spectra of the Electromagnetic wave. So colloquially we call these as either Red, Green, or Blue pixels, according to the color-filter placed on that particular pixel. Every sensor has 2x the number of Green pixels as compared to the Red/Blue pixels; this is due to the fact that the human retina using designated cone cells, during daylight vision are most sensitive to green light. The RGB pixels of the camera sensor is arranged in a typical RGGB/BGGR Bayer pattern mentioned in Fig 3.2.

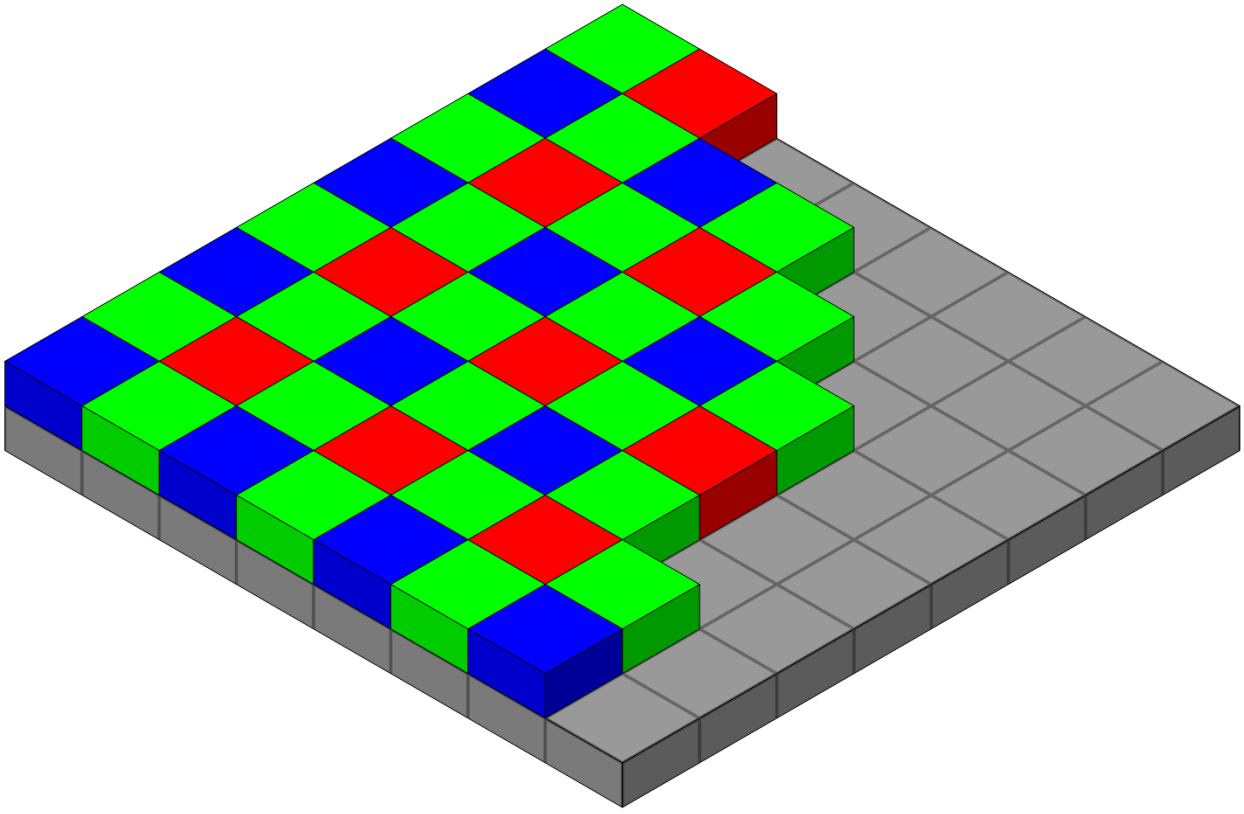


Figure 3.3: RGGB Bayer Pattern

3.2.1. Finding the RGB image from RGGB combination

We know that the no. of Red & Blue pixels is half than that of the Green pixels. So, to get a color image which should be in the format of RGB, a form of interpolation is needed to fill in the blanks. There are different mathematical ways for this interpolation and the implementation type is subject to an individual's choice. This mathematics is known as demosaicing.

For each and every pixel of the RGB image we are either interpolating the value for the red pixel or the blue pixel. In this whole image processing pipeline section, our research work been based on the implementation made by the authors of [10] and [18]. And for that reason, bilinear interpolation has been used for demosaicing.

3.3. Digital Gain

Gain refers to the ratio between the input signal and the output signal, which in our case will be numeric value multiplied with the RAW sensor signal to amplify this brightness and contrast. If we have low gain values, it will make the image dark and reduce the contrast. This is generally applied once the exposure time has been set for taking the image.

3.3.1. Relation between Gain and Exposure

For a camera system if we increase the exposure time, it will lead to an increase in the amount of light captured by the sensor and on the other hand if the gain is increased it will increase the voltage applied to the pixels on the sensor. Both of these factors lead to a brighter image but have different considerable drawbacks. Long exposure may lead to motion blur if any subject moves while the picture is being captured and if we increase the gain it will lead to amplification of the background noise with possible reduction in dynamic range.

3.3.2. Calculating Digital Gain

These days most of the digital cameras do apply digital gain to the image they capture, and the value is set by the manufacturer's auto exposure software. Though this whole system being a black box for us, we need calculate the digital gain for creating our own synthetic dataset.

3.4. White Balance

The colors captured in an image depend on two factors. Its color is the product of the material/object color and the color of the illuminating light. This second factor here is what we need to take care of, and we counter that with white balance. Our objective is to make the image look like it's captured under white-neutral lighting conditions.

3.4.1. Calculating White Balance

The camera systems use a statistical/heuristic approach [19] [20] for calculating the white balance by estimating the Red & Blue channel gains. Likewise Digital Gain, it's also a black box The Darmstadt noise dataset [18] has kept the record of the values and that has been used for our image inversion process. Though my work has incorporated other datasets, I believe that these values will lead to somewhat realistic synthetic data.

3.5. Color Correction

Most of the time color spectra that we get from the camera sensor for an image doesn't match with the standard color spectra like the sRGB. This mismatch led people to incorporate the step of color correction by multiplying the image vector with a color correction matrix, which is specific to every camera sensor.

Thankfully the Darmstadt dataset has these values collected and for our use case it's been used though being different dataset. To generate synthetic images, using the values from a completely different dataset where a different camera sensor might have been used may lead to some imperfections. But our experimental results prove that the generated synthetic images model the real-life images very closely.

3.6. Gamma Compression

Our vision system is typically more sensitive to the gradient in the darker shades than the brighter shades. Therefore, gamma compression typically is the process of allocating more bits of data for low intensity pixels, leading to better dynamic range. For our purpose the standard gamma curve mentioned in the Darmstadt dataset [18] has been used.

$$\Gamma(x) = \max(x, \epsilon)^{1/2.2} \tag{3.6}$$

$$\Gamma^{-1}(y) = \max(y, \epsilon)^{2.2} \tag{3.7}$$

In equ. (3.6), $\epsilon = 10^{-8}$ and while inverting the images equ. (3.7) which is the inverse of (3.6) is being used. Here y denotes the pixel values of the source image while inversion.

3.7. Tone Mapping

Tone mapping is an image processing and computer graphics technique that simulates the appearance of high dynamic range images in a medium with a more constrained dynamic range by mapping one set of colors to another. The limited dynamic range of printouts, CRT or LCD monitors, and projectors makes it impossible to accurately depict the complete spectrum of light intensities found in real-world scenarios. While maintaining the image features and color look necessary to fully appreciate the original scene material, tone mapping solves the issue of significant contrast reduction from the scene radiance to the displayable range.

3.7.1. Inverse Tone Mapping

An image with a low dynamic range can be mapped into an image with a higher dynamic range using the inverse tone mapping technique. Notably, it is employed to convert SDR videos to HDR videos.

3.7.2. Calculation

Standard images with low-dynamic range, are often processed using S-shaped curved which mimics the “characteristics curve” of a film [21]. Considering this to be our case we use a simple “smoothstep” curve while creating the processed RGB images from the RAW images and while inversion from RGB images we use the inverse of equ. (3.8).

$$smoothstep(x) = 3x^2 - 2x^3 \tag{3.8}$$

$$smoothstep^1(y) = \frac{1}{2} - \sin\left(\frac{\sin^{-1}(1-2y)}{3}\right) \tag{3.9}$$

Here in both of the equations (3.8) & (3.9), the inputs $\boldsymbol{x}$ & $\boldsymbol{y}$ are defined in the range of $[\mathbf{0}, \mathbf{1}]$. In real life a simple smoothstep function might not be used. There are many complex edge-aware local tone mapping algorithms [22] and also for HDR images extreme tone mapping is used. As these steps are difficult to reverse engineer, we have kept our calculation simple and followed the steps taken by these authors [10].

Chapter 4 Our Approach

As we know, images are formed due to the pattern formed from photoelectric effect on the image sensors. And these patterns are affected by different kinds of irregularities. There is always some kind of randomness in the photons which hit the sensor, this causes a type of noise which is known as the shot noise. As mentioned in "3.1. Noise In an Image", we know that shot noise is proportional to the $\sqrt{(\text{signal})}$ which means that when intensity of the signal is low, proportionately the intensity of shot noise will be high and will dominate the noise pattern. We also need to take a note of the difference in the electronic pattern formed on different sensors due to their own characteristics signature and image sensitivity, causing the sensor-dependent readout noise. Our present research of image enhancement considers these two types of noise.

Also, from the previous section "Image Acquisition Pipeline" we know in the whole process of image acquisition where the noise gets mixed. Most of the recent image capturing systems de-noise them immediately. But in our case, we want to create a system which can de-noise as well as perform exposure correction irrespective of the camera being used for capturing the image. This is due to the fact that not every camera system does have a good de-noising system and so the method we choose is to take low-light images and convert them into an enhanced version where the exposure is proper and also its denoised so that those images can be used elsewhere.

For our purpose, we need to train a suitable neural network and more importantly we need to feed it with realistic images. These days researchers choose some form of Super-Resolution Generative Adversarial Networks, in this field. This kind of network boasts the

capability of generating good quality images given that it is able to learn the co-relation between the factors which differentiate a good and a bad image. Getting a sheer amount of real low-light images with their corresponding bright image for the same scene is nearly impossible. So, our job is to create a dataset first which models the real-life irregularities.

4.1. Creating Augmented Dataset

Our experimentation involved inverting the ground truth images which were captured using low exposure time. In the case of the LOL dataset, image pairs were already present with one being captured using low exposure time. These images were inverted for converting them to their RAW format, then synthetic noise as mentioned in section "3.1. Noise In an Image" was added and these noisy RAW images were processed to get back their sRGB variants.

In case images are not captured using low exposure time, we have implemented a module which is supposed to change the brightness of the image. It divides the pixel values of the image by a simple integer and its RAW counterpart is generated using our traditional image un-processing module. Noise is added on the RAW format and then all the steps are performed similarly irrespective of the source image being dark or not which is inverting all the un-processing module's steps as mentioned in the "Image Acquisition Pipeline" except of "3.2. Demosaicing".

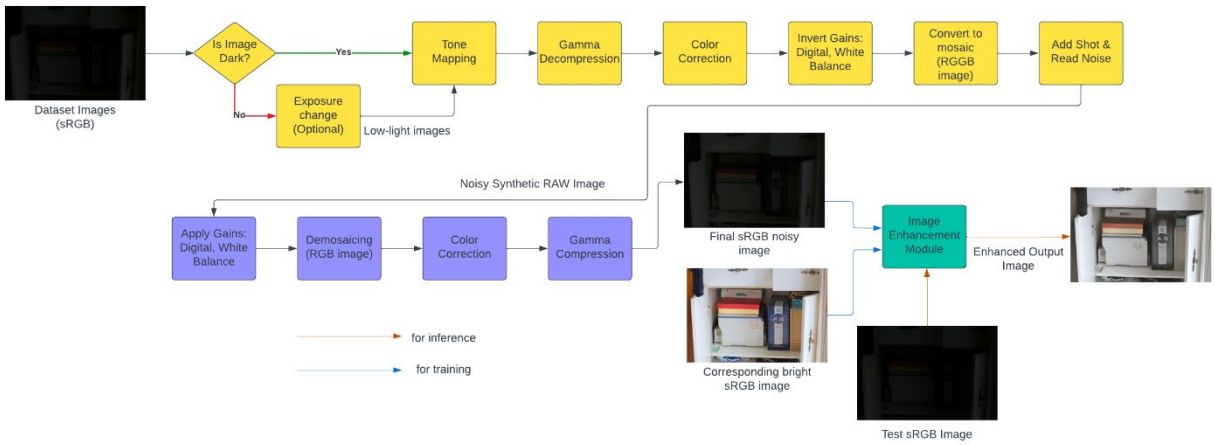


Figure 4.1: The architecture of our proposed method.

The "yellow" boxes are part of the image **un-processing module** which reverts the steps which undergoes in a sensor system while acquiring an image. The "violet" boxes collectively convert the RAW images into their processed sRGB images using the parameters from Darmstadt noise dataset [18], or we can call that as the **processing module**. The green box is the **Image enhancement neural network**, and here LLFLOW is used. The train & test image are same here only for illustration purpose so that we can compare the difference between the bright sRGB image which is the groundtruth vs the enhanced output. Neglecting that bit of a smoothening the enhanced output image is very similar to the groundtruth.

The generated RAW images' mosaic form has less no. of R & B data, precisely half than their original sRGB counterparts. To get the RGB matrix from the RGGB we need a kind of approximation, which in our case is "bilinear interpolation". This method has also been used in the Darmstadt noise Dataset, so we believe this method is fair to use.

The image un-processing module in Figure 4.2 denoted with "yellow" boxes has many processes which are camera system specific and many a times proprietary. So, inverting those steps from an sRGB image would be a difficult task. For our process we have used the parameters from the Darmstadt noise dataset [18]. Though the camera parameters will differ for every system, we assume here that for our purpose the error will be within tolerance limit.

Adding realistic noise is important, but also where we are adding in the pipeline is also important. This is clearly understood from “Figure 3.1”, its shows us exactly when a sensor encounters noise and so what we need to perform in order of mimicking the process. The importance of each and every step of the unprocess-process pipeline from “Figure 4.1” has been justified with the ablation study of those steps.

4.2. Image Enhancement & Denoising Network

The neural network implemented in our research is based on the one developed by Yufei, et al. [1]. Their network follows the Super Resolution GAN architecture which is inspired from the improved Retinex Theory by Wang, et. al. [11] which takes the illumination, reflectance, and noise of the low-light images as conditions for learning a one: many relations map from a low-light/ill exposed image to multiple normal-exposed images of the same scene and it’s due to the fact that a particular scene may be captured with different illuminations and this mapping will help us transform the ill-exposed noisy images to properly exposed image with features intact.

The network is based on the improved Retinex theory which incorporates noise as a major factor (Equation (2.3)) with the help of gradient maps (Equation (2.5). This made us believe that if we incorporate noise in the train images the network should be able to distinguish the difference between noise and useful features, given a reference image is provided. The choice of this particular network is solely due to the reason being that it has already been a state of the art on the LOL dataset. Our purpose is to check that if we did incorporate artificial noise while training, does this network’s cross-dataset as well as LOL dataset performance improve.

4.3. Robustness Evaluation

The network has been evaluated against LOL dataset with numerous modifications as well as on other two datasets which we created, named Pepper and EarthCam. We also evaluated our model for face detection, on the VE-LOL-H dataset which consists of low-light images of people walking on busy streets. This gave us the idea of how useful our architecture can be in real-world applications. We have applied our architecture to numerous different scenarios so that we understand in general the potential and the characteristics of these models. Our experimentation on the Pepper dataset was crucial, and the severe degradation in the performance of the LLFLOW network on this dataset as compared to LOL motivated us in developing our architecture. The Pepper dataset is specifically curated for our purpose, as it has paired images where the low light images are not captured with less exposure time, instead captured in real dark environment. Whereas the EarthCam dataset is a bit different, it may not serve our exact purpose due to the difference in the nature of the images, but we definitely can infer the characteristics of our model.

Chapter 5 Description of Datasets

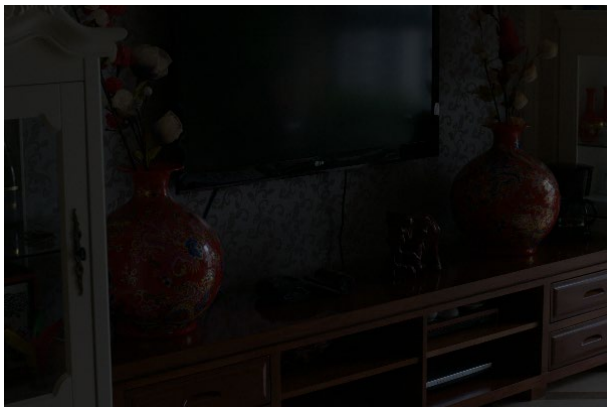
A model/theory can be called robust, only if it works well on different standardized data and on different kinds of setup. This inspired us to find quantitative results using different metrices on various datasets. Some of the datasets which we used are publicly available, we also evaluated using our own dataset which is tailor made for our purpose.

5.1. LOL Dataset

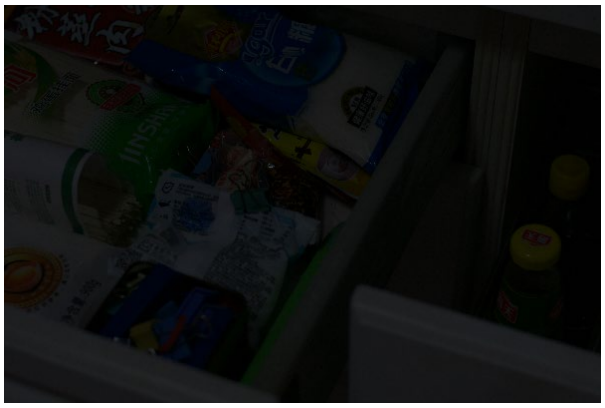
Low **L**ight paired dataset contains 500 image pairs with low-light as well as normal-light images. For a particular scene most of the low-light images are captured by changing two parameters of the camera which the authors [2] have used. These parameters are exposure time and ISO, to get a low-light image. All of these images were resized to the dimensions of 400x600 and converted to PNG (Portable Network Graphics) format. The authors have extended their dataset by adding 1000 modified RAW images collected from the RAISE [13] dataset. The properties of these 1000 extra images are matched with that of 250 real-life dark images collected from public datasets like MEF [23], NPE [24], LIME [25], DICM [26] and Fusion [27]. These 1000 RAW images were transformed to YCbCr, and their Y channel histogram was calculated. Their Y channel was modified according to the 250 reference images' Y channel histogram using Adobe Lightroom. As the authors of the LLFlow network did work with the original 500 images from the LOL dataset, we also chose to follow the same for having a fair comparison.

(a) Scene A: Image with normal exposure

(b) Scene B: Image with normal exposure



(c) Scene A: Image with low exposure



(d) Scene B: Image with low exposure

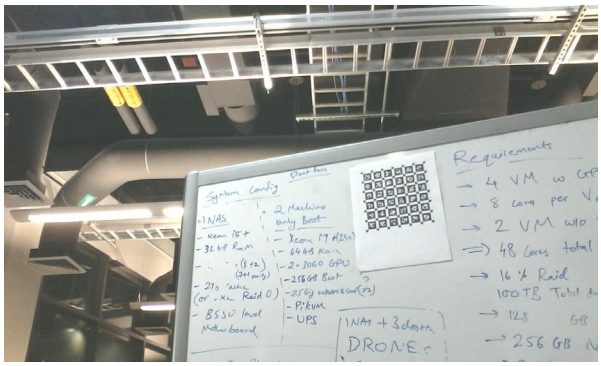
Figure 5.1: Image pairs from the LOL dataset which contains 500 captured image pairs. Here each pair (a)-(c) and (b)-(d) represent an image captured using different camera settings for the same scene. Here (c) and (d) are captured using low-exposure time.

5.2. DronesLab Images from Robot Pepper

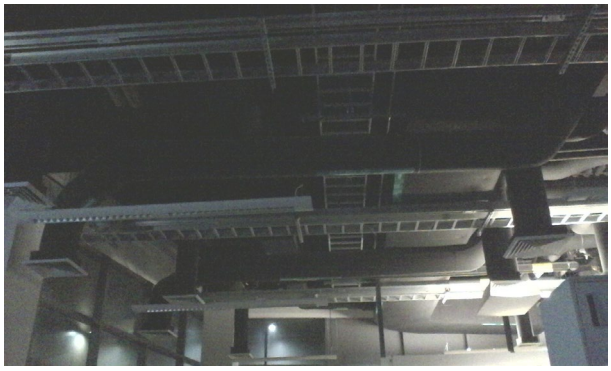
KAI is the name given to the department's **SoftBank Pepper Robot**. This humanoid robot is one of the new members of our lab. It has two 5 megapixels camera sensors, one mounted in the mouth and the other one on the forehead. There is a 3rd camera integrated with the tablet which sits on the chest of the robot. We have captured some low light images with its tablet and executed our model to see how well it's able to enhance these images.

Here our attempt in creating this small, paired dataset is to verify whether the behavior of our network architecture our on the modified LOL dataset (explained in Chapter 4) tallies with other datasets. This dataset has only 2 image pairs displayed in Figure 5.2, where the pair is denoted with the scene name. While creating the paired images, the robot has been kept stationary, and the images are captured while keeping the lights on for the normal images and off for the low light images. The images are captured in normal mode (not pro) that helped us in not knowing the image processing parameters like White Balance, exposure time and Digital Gain. This might work better for our purpose of evaluating the trained model θ in a completely blind circumstance. The brute force method of creating low-light image under a real low-light environment ensures that noise characteristics have been well characterized. Due to this reason the dataset has been directly evaluated without any modification by adding synthetic noise. This may help us in verifying whether our synthetic noise characterizes the real noise well.

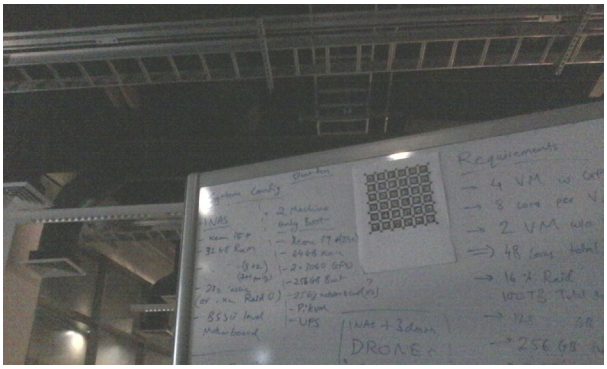
(a) Scene A: Image with normal illumination



(b) Scene B: Image with normal illumination



(c) Scene A: Image with low illumination



(d) Scene B: Image with low illumination

Figure 5.2: Image captured in Davis Hall Room-105/DronesLab.

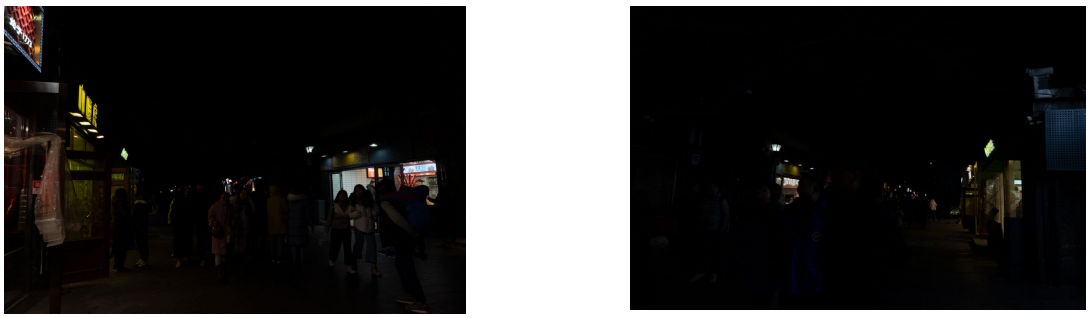
5.3. VE-LOL-H Dataset

Vision **E**nhancement in the **LO**w-**L**ight condition **(VE-LOL)** is a dataset created by Jiaying et al. [28] to explore the area of low light image enhancement. While creating the dataset they have considered an important factor for these image enhancement models, and i.e., to evaluate how well these work for face recognition. They have two separate modules in the dataset, one has the image pairs similar to the one we used for LOL dataset, those are mainly for training the network and the other part is for evaluating on low-light images which do contain human face,

with the face coordinates being manually labelled. This will greatly help us to understand how well these models presently work for real-life applications, and are they fit to be deployed in the commercial space or do we need to improve these models a lot.

VE-LOL-L is the first module of their dataset which do contain 2500 paired images and among them 1000 pairs are synthesized from the RAISE [13] dataset. While the others are real-image pairs degraded similarly like ours using the noise modeling by Brooks et al. 2019 [10].

VE-LOL-H is the model which has been utilized by us for understanding the behavior of our model in areas which require high precision like face recognition. Here 10,940 images in total (6940 for training and validation and 4000 for testing) were captured in low light and manually annotated with bounding boxes to identify human faces.



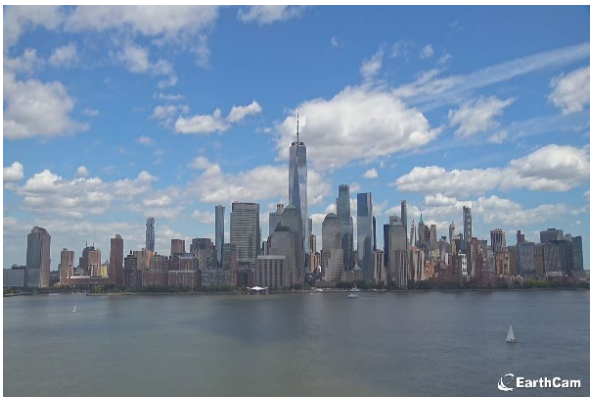
(a) (b)

Figure 5.3: Typical images from the VE-LOL-H dataset.

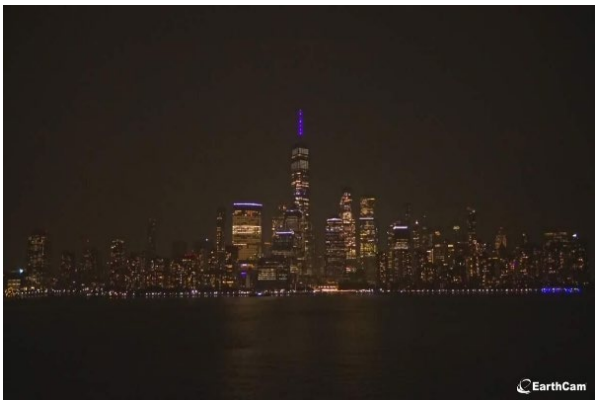
The images in this dataset were taken using Sony α6000 and Sony α7 E-mount cameras with varied capturing settings on a number of busy streets in the Beijing area, catching faces with a various scale, stances, and appearances making it a truly unconstrained dataset. These images have a resolution of 1080 x 720, which were downscaled from 6 k x 4 k for our convenience.

5.4. EarthCam Dataset

Canadian Wildfire Smoke affected North America, mainly the East Coast during the month of July. Major cities were engulfed with the some and led to extremely unhealthy air quality. EarthCam pictures taken between June 6th and 8th show huge variance image texture. So, we thought about putting those images into our chosen network to observe how those will be processed.



(a) June 5th



(b) June 6th 11:30pm

(a) June 7th 12pm

(b) June 7th 2pm

Figure 5.4: Some of the image from our dataset “EarthCam”. The images are captured from “https://www.earthcam.com/usa/newyork/worldtradecenter" and pose vastly different texture, artifacts, and color tone. For our purpose we have kept image (a) as the reference for all the metrics calculations.

Chapter 6 Experiments

6.1. Analyzing Experimental Data

The LOL dataset as described earlier is a paired dataset, and all of our modifications are done on the images captured with low exposure time. Our experiments involve different ways of adding shot & read noise to these images for training the network. These methods are described below:

- Noise added after the images are passed through the unprocessing module to get their RAW variants, hence the name "**RAW_noisy**" been used. After noise being added the images are converted back to their sRGB format using the processing module.
- Only some of the parts of the unprocessing module were used like leaving the color correction matrix, white balance, and digital gain while creating the dataset, leaving only the mosaicking/demosaicing process aside. Hence the name "**no_ccm_wb_gain**" been used for this variant of our dataset. Here the noise has been added to images in their RGGB format without being converted to their respective RAW variants.
- **pretrained**: This denotes the original LLFlow model which was trained by the authors on the unmodified LOL dataset.

The column heading "**model type**" for the tables in this section denotes the way LLFLOW network was trained, like "**custom v2_RAW_noisy**" means that it was trained on the modified dataset where noise was added to the generated RAW images and then again, those images were converted

back to sRGB. For all these tables here ↑ signifies that higher the value its better and similarly ↓ is vice versa. The v2 or v3 for the model signifies:

- **v2**: - train-validate datasets are modified similarly.
- **v3**: - train dataset modified but validated using unmodified LOL [28] data.
- **pretrained**: - pretrained LLFLOW smallNet model.

Model type	Eval dataset	PSNR (↑)	SSIM (↑)	LPIPS (↓)
custom v2_RAW_noisy	LOL	17.45	0.71	0.32
custom v2_RAW_noisy	RAW_noisy_eval	22.71	0.81	0.34
custom v2_RAW_noisy	no_ccm_wb_gain	20.49	0.60	0.47
custom v3_RAW_noisy	LOL	19.23	0.78	0.34
custom v3_RAW_noisy	RAW_noisy_eval	21.47	0.79	0.40
custom v3_RAW_noisy	no_ccm_wb_gain	20.59	0.66	0.48
pretrained	LOL	24.06	0.91	0.14
pretrained	RAW_noisy_eval	18.82	0.58	0.71
pretrained	no_ccm_wb_gain	19.25	0.61	0.64
custom v2_ no_ccm_wb_gain	LOL	16.61	0.66	0.33
custom v2_no_ccm_wb_gain	RAW_noisy_eval	17.81	0.63	0.45
custom v2_no_ccm_wb_gain	no_ccm_wb_gain	22.82	0.81	0.31
custom v3_no_ccm_wb_gain	LOL	17.73	0.69	0.38
custom v3_no_ccm_wb_gain	RAW_noisy_eval	19.15	0.69	0.49
custom v3_no_ccm_wb_gain	no_ccm_wb_gain	21.52	0.78	0.37

Table 6.1: Comparison of LLFlow models trained on different versions of the LOL dataset.

Here each model has been evaluated against different variants of the LOL dataset using the metrices PSNR, SSIM and LPIPS. This gives us an idea about each model's performance. As from this table its not very idea to conclude which one is the best model so we need to conduct certain calculations on data from this table.

Model type	PSNR (↑)	SSIM (↑)	LPIPS (↓)
custom v2_RAW_noisy	μ = 20.21 σ = 2.640 σ^2 = 6.973	μ = 0.70 σ = 0.105 σ^2 = 0.011	μ = 0.37 σ = 0.081 σ^2 = 0.006
custom v3_RAW_noisy	μ = 20.43 σ = 1.128 σ^2 = 1.27	**μ = 0.743** σ = 0.072 σ^2 = 0.005	μ = 0.406 σ = 0.070 σ^2 = 0.005
pretrained	**μ = 20.71** σ = 2.909 σ^2 = 8.46	μ = 0.70 σ = 0.182 σ^2 = 0.033	μ = 0.496 σ = 0.311 σ^2 = 0.096
custom v2_ no_ccm_wb_gain	μ = 19.45 σ = 3.294 σ^2 = 10.851	μ = 0.71 σ = 0.096 σ^2 = 0.009	**μ = 0.36** σ = 0.076 σ^2 = 0.006
custom v3_no_ccm_wb_gain	μ = 19.46 σ = 1.914 σ^2 = 3.666	μ = 0.72 σ = 0.052 σ^2 = 0.003	μ = 0.41 σ = 0.066 σ^2 = 0.004

Table 6.2: Mean performance of the LLFlow models. Here the mean performance on different variants of LOL datasets for each model is summarized.

For each metric the best performing model is highlighted in green. The mean results are inconclusive for selecting the best model, so we select the models with best mean values and use standard deviation as an elimination process.

Model type	PSNR (↑)	SSIM (↑)	LPIPS (↓)
custom v3_RAW_noisy	μ = 20.43 **σ = 1.128** **$\sigma^2 = 1.27$**	μ = 0.743 **σ = 0.072** **$\sigma^2 = 0.005$**	μ = 0.406 **σ = 0.070** **$\sigma^2 = 0.005$**
pretrained	μ = 20.71 σ = 2.909 $\sigma^2 = 8.46$	μ = 0.70 σ = 0.182 $\sigma^2 = 0.033$	μ = 0.496 σ = 0.311 $\sigma^2 = 0.096$
custom v2_ no_ccm_wb_gain	μ = 19.45 σ = 3.294 σ^2 =10.851	μ = 0.71 σ = 0.096 $\sigma^2 = 0.009$	μ = 0.36 σ = 0.076 $\sigma^2 = 0.006$

Table 6.3: We have selected 3 of the best models based on mean performance across all the modifications on the LOL dataset.

Using standard deviation and variance helped us in deciding which model would be ideal in our scenario. It's important to choose a model which has a good mean performance but also, we need to consider consistency and that's where the variance and standard deviation comes in. Using these parameters, we are able to determine which model would be ideal in our use case.

Model type	Eval dataset	PSNR (↑)	SSIM (↑)	LPIPS (↓)
custom v3_RAW_noisy	LOL	19.23	**0.78**	**0.34**
custom v3_LOL_AWGN	LOL	**20.16**	0.77	0.45

Table 6.4: Model performance under different type of Noise on the LOL dataset.

Comparing the type of noise being added for the models displayed in Table 6.4, we conclude that the one trained on Additive White Gaussian Noise (AWGN) gets outperformed in majority of the metrics, namely SSIM and LPIPS. Though the difference in performance for the metrics PSNR and SSIM is marginal, there's a significant lead for the model which has been trained on the custom noise for the metric LPIPS. LPIPS is currently the most sophisticated way for comparing image quality and has been explained later. This means that our way of adding shot & read noise leads to better feature preservation and proves that noise doesn't always follow the simple Gaussian distribution.

Model type	Eval dataset	PSNR (↑)	SSIM (↑)	LPIPS (↓)
custom v3_RAW_noisy	Pepper	**13.56**	0.53	**0.49**
pretrained	Pepper	12.45	**0.54**	0.57
custom v2_ no_ccm_wb_gain	Pepper	13.04	0.44	0.52
custom v3_RAW_noisy	EarthCam	11.71	0.53	**0.68**
pretrained	EarthCam	**13.31**	**0.63**	0.69
custom v2_ no_ccm_wb_gain	EarthCam	12.31	**0.63**	0.73

Table 6.5: Comparison of LLFlow models trained on our datasets Pepper and EarthCam. Here we have tested the three best performing models from Table 6.2, to understand the cross-dataset performance, and have a better conclusive result of which one to be preferred.

Table 6.3 lists the models which has the best mean performance across all the modifications of the LOL dataset. We found different evaluation metrices gave us different results, and we got 3 models which are the best in terms of "PSNR", "SSIM" and "LPIPS". Therefore, we had to resort to other quantitative methods like standard deviation and variance to understand which model

would be the ideal one for our scenario. Applying that we found, that the "**custom v3_RAW_noisy**" model is more consistent than other models. Our results on a handful of images from the Pepper robot also made us believe that our way of evaluation in the previous experimental section was in the right direction. Results on the EarthCam dataset (from Table 6.5) may not be very related to our low-light image enhancement work, but it definitely helps us in better understanding the network behavior.

6.1.1 Metrics Explanation

Here 3 metrics have been used extensively to compare the performance. PSNR means Peak-Signal-to-Noise-Ratio. PSNR is meant to calculate the mean squared error (MSE) comparing the pixel value difference between the ground truth (I) and the generated image(K).

$$MSE = \frac{1}{mn}\sum_{i=0}^{m-1}\sum_{j=0}^{n-1}[I(i,j) - k(i,j)]^2 \tag{6.1}$$

SSIM or Structural Similarity Index Measure is meant to compare the structural similarity on various windows of an image. It's the weighted sum of image components like luminance (l), contrast (c) and structure (s) with their respective weights being α, β, γ and is calculated between different image windows (x and y) one from image I and the other from image K. The simplified formula for SSIM:

$$SSIM(x,y) = l(x,y)^{\alpha} \cdot c(x,y)^{\beta} \cdot s(x,y)^{\gamma} \tag{6.2}$$

Both of the above-mentioned methods are more related with the comparing image similarity based on structure of the image or difference of the pixel value, whereas the 3rd metric LPIPS (Learned Perceptual Image Patch Similarity) [15] , is better used to compare the feature

between the ground truth and the generated image. This is done by computing the distance between the activations of two image patches using some pre-trained network, such as VGG or AlexNet. This is used to compare images keeping the human perception in mind rather than pixel-value differences.

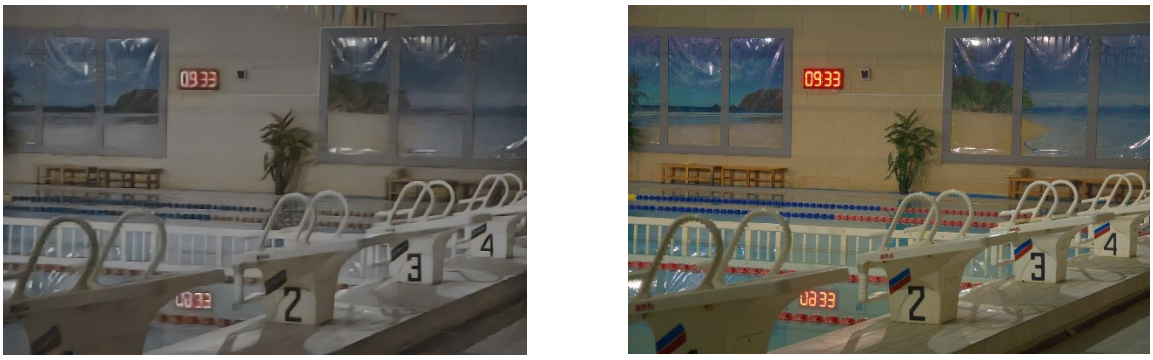
6.2. LOL Train-Eval Split

The authors Yufei et al. [1] have utilized only the 500 captured by Wei et al. [2] for the LOL dataset. 485 image pairs resized to 400x600x3 were kept for training the network, the rest 15 image pairs were kept for model evaluation. To keep our experimentation similar for easy comparison we also used the same data with a similar split up.

6.3. Analyzing Image Results

This section gives an overview of the type of images used for our experimentation and how they look like. Moreover, the output images of the model on these datasets are used for understanding the behavior of these generative models and how well they fair in a diverse environment. We also had a better understanding of how we can improve the model in the future.

6.3.1. LOL Dataset



(a) Generated Image (b) Reference Image

(c) Noisy Low Light Image (d) Paired Low light Image

Figure 6.1: The image (c) x_{l_noisy} has been generated from (d) x_l with the help of the unprocess pipeline and then adding read and shot noise. Here x_l is the image captured with low-exposure time during creation of the LOL dataset. x_l and (b) x_{ref} are image pairs which are captured for the same scene. Image (a) x_h is the image with normal exposure generated from the model θ.

Figure 6.1 & Figure 6.2 displays different type of low light images generated from x_l, but studies in Section "6.1. Analyzing Experimental Data" shows us that model θ trained on x_{l_noisy} - x_{ref} also denoted as "**custom v3_RAW_noisy**" in Table 6.1, is able to preserve more features while creating the x_h as compared to other variants.

(a) No unprocessing, Noisy Image (b) Low Light Gaussian Noise Image

Figure 6.2: Compares the way the low light images were generated from x_l.

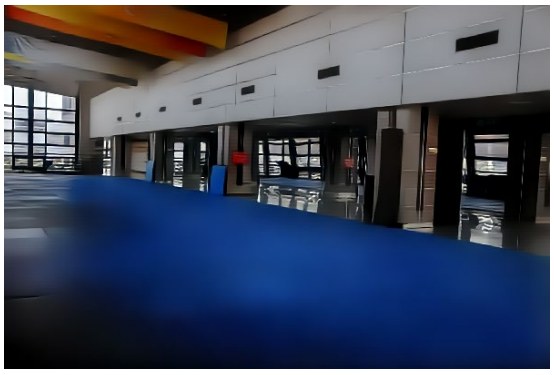
(a) $x_{l\_no\_CCM\_WB\_Gain}$ means that none of the image processing steps like "color correction matrix" (CCM), "White Balance" (WB) and "Digital Gain" (Gain) were reversed. Shot + Read noise was added to the Bayer RGGB version of the images and then these were converted back to their sRGB variants. Image (b) x_{l_AWGN} has also been created from x_l, but here all the unprocessing steps are similar like the x_{l_noisy}, except for the noise model. Here the Shot + Read noise was replaced by "Additive White Gaussian Noise" (AWGN) with $\mu = 0, \sigma = 1.5$.

Though we did compare models using quantitative results from Table 6.5 for better human interpretation we thought of sharing some of the output images. Figure 6.3 compares the output from two of models based unmodified LOL data. Simply from observation we infer model "**custom v3_RAW_noisy"** being able to preserve more details on a kind of data on which it has not been trained on (i.e., no noise data) though it loses some color contrast w.r.t its counterpart. This helps us to conclude that this model has more consistent performance than the other one i.e., "**custom v2_ no_ccm_wb_gain**".

(a)

(b)

(c)



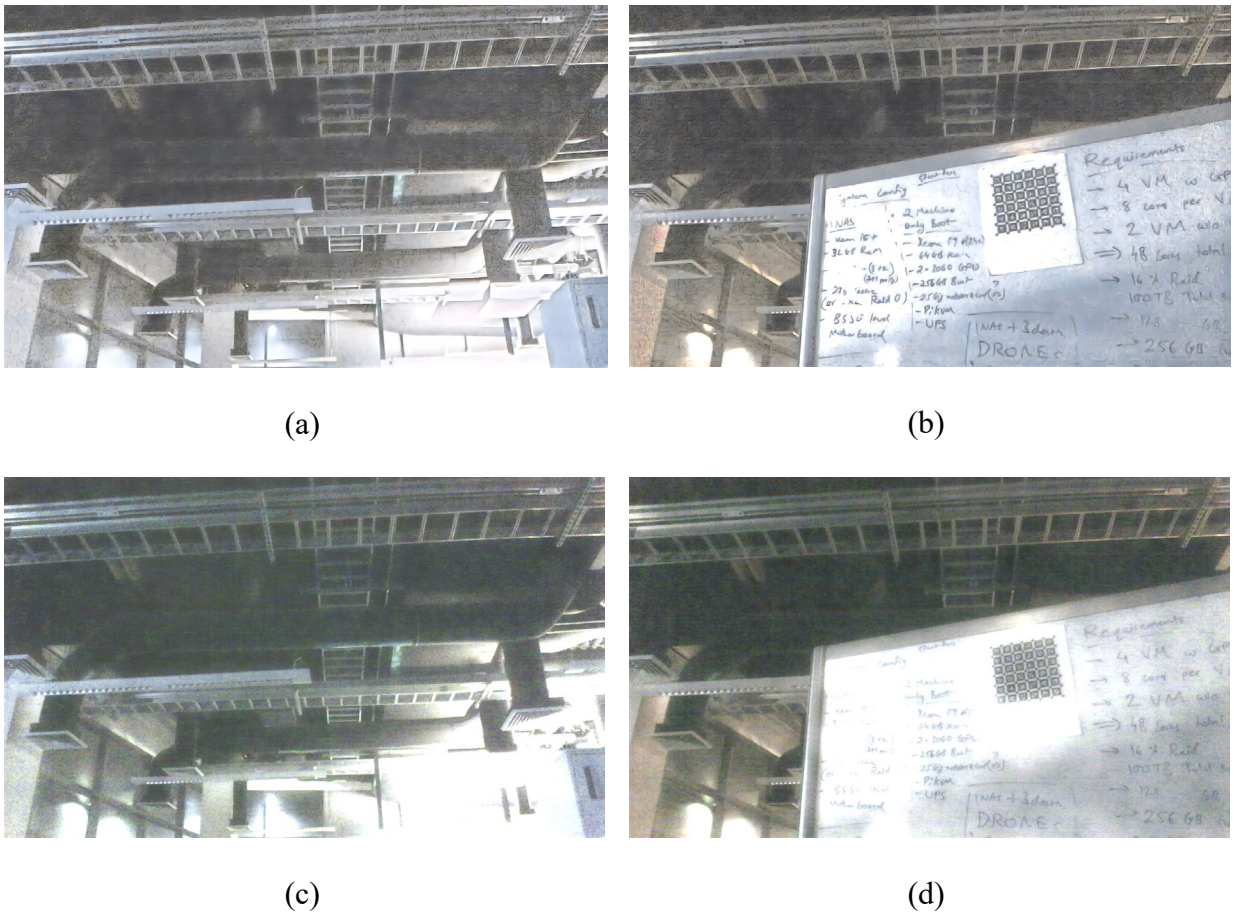
(d)

(e)

(f)

Figure 6.3: Compares the image being generated from model "**custom v2_ no_ccm_wb_gain**" {(b), (d)} and "**custom v3_RAW_noisy**" {(a),(c)} w.r.t the bright images for the same scenes. Here (e) and (f) are the images captured in proper lighting conditions (reference images) and are used to compare the models' output images.

6.3.2 Pepper images



(a) (b)

(c) (d)

Figure 6.4: Image (a) & (b) are generated from the "**custom v3_RAW_noisy**" model and (c) & (d) are the outputs from the "**pretrained**" model.

The models were given low-light images as discussed in section 5.2. DronesLab Images from Robot Pepper. We clearly visualize that the image generated from the model Θ trained on the RAW noisy LOL images was able to better preserve the features by maintaining the exposure in a better way than the one trained by the authors Yufei et al. [1] also denoted as "**pretrained**" in Table 6.1. But also, if we observe image (a) & (b) closely we may notice that these images a filled with visual artifacts more than (c) & (d) and may look noisy in our eyes. Our visual perception does match with the qualitative results presented in Table 6.1. The image generated

from our custom model will definitely be better suited in applications like object detection and to be specific in our image example like **OCR** (optical character recognition). There are also various other domains in which this kind of image enhancement model may be useful.

We have a handful of images from this robot, so we have used this data only for understanding how our modified model performs w.r.t. the pretrained model.

6.3.3 VE-LOL-H dataset

(a) face detection algorithm completely failed

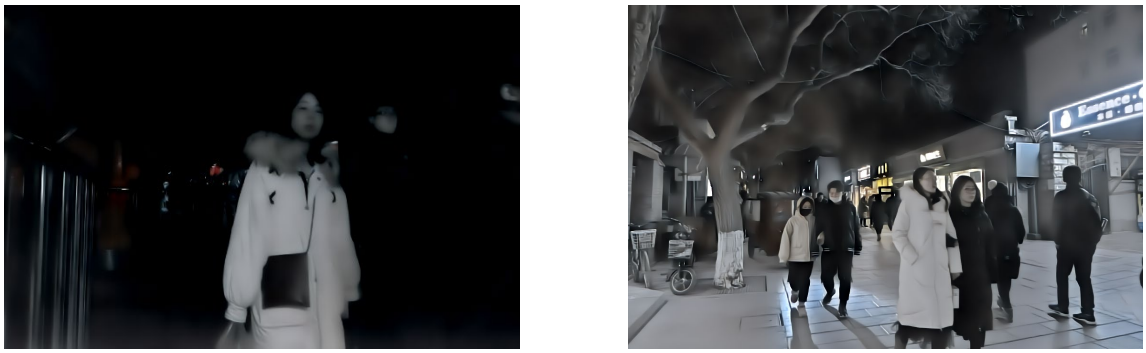
(b) face detection algorithm able to detect at least 1 face

Figure 6.5: Image (a) is from the VE-LOL-H dataset, where it's completely unprocessed whereas the right image (b) is the same image passed via our image enhancement network.

For our experimentations we took the train folder and passed that thorough our LLFLOW network (**custom v3_RAW_noisy**), which was trained on the noisy LOL data, This resulted in images which were brighter but also somewhat smoothened out and when we applied a face detection network (RetinaFace [29]) on these images the total percentage of face being detected for the enhanced images was higher than the original ones. For the enhanced image set it's approximately 11% whereas for the original images it's only 8%, that's a 37.5% increase. This justifies that in extreme conditions an intermediate step is necessary which may take care of

enhancement scenarios and makes it easy for the application-based models to work better than they would normally perform.

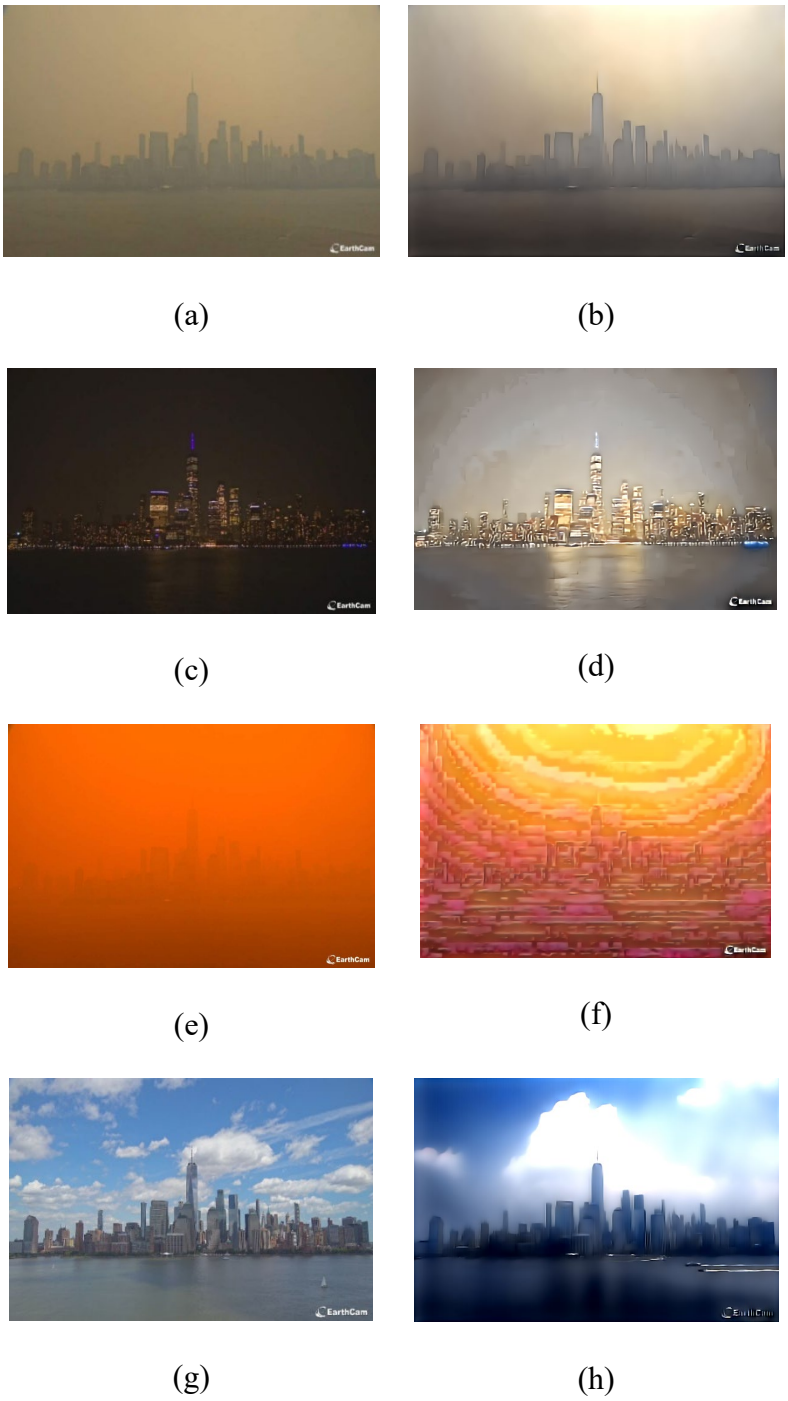
Though our network has resulted in better face detection, facial recognition needs the precise details of a face which our network is losing while enhancing images. This phenomenon is not limited to this dataset, but we have observed something similar on our EarthCam dataset. Our experimentations from the Figure 6.5 make us believe that we need to further investigate and come up with ways of improving the network.



(a) (b)

Figure 6.6: Typical output from our image enhancement network. The faces are generally smoothened while the illumination of these images is being adjusted by the network.

6.3.4 EarthCam dataset



(a) (b)

(c) (d)

(e) (f)

(g) (h)

Figure 6.7: Comparison of the EarthCam image of New York city from June 5th to 7th. Here images on the left side are the unaltered EarthCam images and on the right are the output for those images from the "**custom v3_RAW_noisy**" model.

In this dataset the images have different textures, lighting conditions and objects. Our purpose here is to observe the output characteristics of pictures from the model which we have been using for our experiments. And as it was trained on image pairs which contains low light – bright light images, it performed fairy well in changing the image illumination for the pair (c)-(d) from Figure 6.6. But for this pair it did lose some intricate detail of the buildings (like the World Trade Center and the other one in its surroundings), this is due to the smoothening which happen for denoising these images as its one of the major factors for our trained network as per the Improved Retinex Theory. Also, the model did perform fairly well in illumination the image for the pair (a)-(b), but regarding the details neither it did lose nor were there many as the image is filled with smoke.

For the last 2 pairs the model didn't perform as we would normally hope, but the reason for both the pairs (e)-(f) and (g)-(h) is different. For the (e)-(f) pair it's due to the fact that the NYC skyline (which is the main feature) in image (e) is overshadowed by a thick layer of smoke and that makes it impossible to infer what's the image is about. But for the last pair (g)-(h) is somewhat different. The purpose of a well illuminated image (g) was to check how does the model handles these kinds of scenarios, and we believe its hallucinated with that it's supposed to understand as noise and what's a useful feature. This image is filled up with sharp gradients, and this sudden change in pixel values are most likely considered as noise which needs to be smoothened out.

Chapter 7 Conclusion

Image captured in low-light environments has always been hard to perceive due to the lack of detail and noise being one of the predominant factors in distorting the captured details. Our characterization of different types of noise and adding them synthetically in the LOL dataset proved to be useful as it has greatly improved the performance of the Image Enhancement network. The noise model of the network in section "2.4.1. Overview of LLFlow Architecture" is able to learn the randomness which becomes predominant in low-light conditions. This has helped us in creating a network which performs better while enhancing images in blind scenarios. There are various fields where these kinds of networks may prove to be useful. In autonomous vision based driving systems, having synthetic images with better illumination will certainly help. It will greatly improve the object detection capability as proved in the previous section with person detection being our example.

Our experiments on each different datasets led us in learning something new about these Generative networks. Our results Pepper images suggest that though the custom trained network is able to preserve features better, it also adds unwanted artifacts in some of the generated images. This is due to the fact that completely modeling noise in a blind scenario is always challenging and there's room for improvement. Our experiments on EarthCam led us to conclude that sometimes the model gets hallucinated and messes up, in those conditions the enhanced image gets severely altered from the original image and our expectations don't hold up. From experimentation on the VE-LOL-H dataset we learned that our present model certainly performs better in face

detection tasks as compared to the completely unaltered dark images. But for face recognition it will require further modifications. The face images are blurred in many cases on the dataset which is completely unsuitable.

Improvement in the performance of the model can be achieved both ways. First is to collect massive datasets with image pairs being collected from different image scenarios and lighting conditions. But as we all know there will always be limitations in creating an ideal data set, so we should consider the second solution. This approach involves the model to understand the context by matching blobs from the captured image with its pre-trained features, also they are capable of understanding the type of noise, as the proportion between shot and read noise differ for different scenarios. This leads in the model performing image enhancement which is specific to each and every image. Its already being implemented in some of the smartphone cameras, like how they capture sharp images of the moon in complete pitch-dark conditions. For this scenario the enhancement process is fulfilled by matching the blurred/noisy blob of the moon from the captured image with its pretrained features of moon and then adding those pretrained details to fill the gaps. This process might greatly enhance the theoretical capability of camera systems.

Bibliography

[1] W. Yufei and e. al., "Low-light image enhancement with normalizing flow," in *Proceedings of the AAAI Conference on Artificial Intelligence.*, 2022.

[2] C. Wei and e. al., "Deep retinex decomposition for low-light enhancement," in *arXiv preprint arXiv:1808.04560 (2018)*.

[3] Chen, Chen and e. al., "Learning to see in the dark.," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2018.

[4] Schwartz, Eli, R. Giryes, Bronstein and A. M., "Deepisp: Toward learning an end-to-end image processing pipeline," in *IEEE Transactions on Image Processing (2018): 912-923*.

[5] A. Foi, M. Trimeche, V. Katkovnik and K. Egiazarian, "Practical Poissonian-Gaussian Noise Modeling and Fitting for Single-Image Raw-Data," in *IEEE Transactions on Image Processing, vol. 17, no. 10, pp. 1737-1754,*, Oct. 2008.

[6] Anaya, Josue and A. Barbu, "Renoir–a dataset for real low-light image noise reduction.," in *Journal of Visual Communication and Image Representation 51 (2018): 144-154.*, 2018.

[7] F. Linwei and e. al., "Brief review of image denoising techniques," in *Visual Computing for Industry, Biomedicine, and Art 2.1 (2019): 1-12.*, 2019.

[8] K. Dabov, A. Foi, V. Katkovnik and K. Egiazarian, "Image denoising by sparse 3-D transform-domain collaborative filtering.," in *IEEE Trans Image Process 16(8): 2080–2095.*, 2007.

[9] S. Gu, Q. Xie, D. Meng, W. Zuo, X. Feng and L. Zhang, "Weighted nuclear norm minimization and its applications to low level vision.," in *International journal of computer vision 121 (2017): 183-208.*, 2017.

[10] T. Brooks and e. al., "Unprocessing images for learned raw denoising.," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. 2019*.

[11] Y. Wang, Y. Cao, Z.-J. Zha, J. Zhang, Z. Xiong, W. Zhang and F. Wu, "Progressive retinex: Mutually reinforced illumination-noise perception network for low-light image enhancement.," in *Proceedings of the 27th ACM international conference on multimedia. 2019.*.

[12] Kobyzev, Ivan, S. J. Prince and M. A. Brubaker., "Normalizing flows: An introduction and review of current methods," in *IEEE transactions on pattern analysis and machine intelligence 43.11 (2020): 3964-3979.*, 2020.

[13] Dang-Nguyen, Duc-Tien and e. al., "Raise: A raw images dataset for digital image forensics.," in *Proceedings of the 6th ACM multimedia systems conference. 2015.*.

[14] I. Goodfellow and e. al, "Generative adversarial networks," in *Communications of the ACM 63.11 (2020): 139-144.*.

[15] Z. Richard, P. Isola, A. A. Efros, E. Shechtman and O. Wang, "The unreasonable effectiveness of deep features as a perceptual metric.," in *Proceedings of the IEEE conference on computer vision and pattern recognition. 2018.*.

[16] X. Wang, K. Yu, S. Wu, J. Gu, Y. Liu, C. Dong, C. C. Loy, Y. Qiao2 and X. Tang, "Esrgan: Enhanced super-resolution generative adversarial networks.," in *Proceedings of the European conference on computer vision (ECCV) workshops. 2018.*.

[17] S. W. Hasinoff, "Photon, Poisson Noise.," in *Computer Vision, A Reference Guide 4 (2014): 16.*, 2014.

[18] Plotz, Tobias and S. Roth, "Benchmarking denoising algorithms with real photographs," in *Proceedings of the IEEE conference on computer vision and pattern recognition. 2017*.

[19] A. Gijsenij, T. Gevers and J. v. d. Weijer, "Computational color constancy: Survey and experiments," in *TIP*, 2011..

[20] J. T. Barron and Y.-T. Tsai, "Fast fourier color constancy," in *CVPR*, 2017.

[21] R. Davis and F.Walters, "Sensitometry of photographic emulsions and a survey of the characteristics of plates and films of American manufacture.," in *Govt. Print. Off., 1922.*.

[22] P. E. Debevec and J. Malik, "Recovering high dynamic range radiance maps from photographs.," in *SIGGRAPH*, 1997.

[23] K. Ma, K. Zeng, Wang and Zhou, "Perceptual quality assessment for multiexposure image fusion.," in *IEEE Transactions on Image Processing, 24(11):3345, 2015*.

[24] W. Shuhang, Z. Jin, Hu, H. Miao and L. Bo, "Naturalness preserved enhancement algorithm for non-uniform illumination images," in *IEEE Transactions on Image Processing, 22(9):3538–48, 2013*.

[25] G. Xiaojie, L. Yu and L. Haibin, "Lime: Low-light image enhancement via illumination map estimation.," in *IEEE Transactions on Image Processing, 26(2):982–993, 2017*.

[26] C. Lee, C. Lee and C. S. Kim, "Contrast enhancement based on layered Contrast enhancement based on layered," in *IEEE International Conference on Image Processing, pages 965–968, 2013*.

[27] K. Dabov, A. Foi and K. Egiazarian, "Image denoising with blockmatching and 3d filtering," in *Proceedings of SPIE - The International Society for Optical Engineering, 6064:354–365, 2006*.

[28] J. Liu, D. Xu, W. Yang, M. Fan and H. Huang, "Benchmarking Low-Light Image Enhancement and Beyond," in *International Journal of Computer Vision (2021)*, 2020.

[29] J. Deng, J. Guo, E. Ververas, I. Kotsia and Z. S, "Retinaface: Single-shot multi-level face localisation in the wild.," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition (pp. 5203-5212).*, 2020.

[30] "Wikipedia," [Online]. Available: https://en.wikipedia.org/wiki/Bayer_filter.

[31] Abdelrahman, S. Lin and M. S. Brown, "A high-quality denoising dataset for smartphone cameras.," Proceedings of the IEEE conference on computer vision and pattern recognition. 2018..

[32] A. Abdelhamed, M. A. Brubaker and M. S. Brown, "Noise flow: Noise modeling with conditional normalizing flows.," in *Proceedings of the IEEE/CVF International Conference on Computer Vision. 2019.*.

[33] Ledig, Christian; Lucas Theis; Ferenc Huszár; Jose Caballero; Andrew Cunningham; Alejandro Acosta; Andrew Aitken; et al., "Photo-realistic single image super-resolution using a generative adversarial network.".

[34] Y. Jiang, X. Gong, D. Liu, Y. Cheng, C. Fang, X. Shen, J. Yang, P. Zhou and Z. Wang, "Enlightengan: Deep light enhancement without paired supervision.," in *IEEE transactions on image processing 30 (2021): 2340-2349*.

[35] D. P. Kingma and P. Dhariwal, "Glow: Generative flow with invertible 1x1 convolutions.," in *Advances in neural information processing systems 31 (2018)*.

[36] Kirichenko, Polina, P. Izmailov and A. G. Wilson., "Why normalizing flows fail to detect out-of-distribution data," in *Advances in neural information processing systems 33 (2020): 20578-20589.*.

[37] C. Jianrui, S. Gu and L. Zhang, "Learning a deep single image contrast enhancer from multi-exposure images.," in *IEEE Transactions on Image Processing 27.4 (2018): 2049-2062.*.

[38] C. X, Z. Q, L. M, Y. G and H. C, "No-reference color image quality assessment: From entropy to perceptual quality.," in *EURASIP Journal on Image and Video Processing 2019.1 (2019): 1-14*.

[39] C. Cheng, S. Zhang, J. Xing, Z. Lei, S. Z. Li and X. Zou., "Selective refinement network for high performance face detection.," in *Proceedings of the AAAI conference on artificial intelligence. Vol. 33. No. 01. 2019.*.

[40] X. Fu, Y. Sun, M. LiWang, Y. Huang, X.-P. Zhang and X. Ding., "A novel retinex based approach for image enhancement with illumination adjustment.," In 2014 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), pp. 1190-1194. IEEE, 2014..

[41] M. Fan, W. Wang, W. Yang and J. Liu., "Integrating semantic segmentation and retinex model for low-light image enhancement.," in In Proceedings of the 28th ACM international conference on multimedia, pp. 2317-2325. 2020..

[42] Wang, Zhou, et al. "Image quality assessment: from error visibility to structural similarity." *IEEE transactions on image processing* 13.4 (2004): 600-612.