Pix2Pix based Image Enhancement

Atishay Jain, Chaitanya Animesh, Madhumitha Karri,
Purva Kothari, Satvik Gupta, Shreyas Anantha Ramaprasad, Shivaank Agarwal
Department of Computer Science, UCSD
La Jolla 92037

Abstract

In recent times, the demand for high-quality cameras has risen quite sharply. High-quality cameras are ubiquitous in smartphones all across the globe. However, the quality of images captured using smartphones are still far from the quality of images captured by expensive DSLR cameras. In this work, we present a deep learning based approach for enhancing images captured by smartphones in a way that they appear as though they have been captured by DSLR cameras. We a use a popular conditional GAN deep learning architecture - Pix2Pix - for our task. We decompose the task of image enhancement into three parts - Directly mapping a phone image to that of DLSR, Deblurring and conversion of Low dynamic range images to high dynamic range. We experiment with the DPED [6] and the LDR-HDR-pair Dataset [8] by performing various changes to the original Pix2Pix architecture, using different loss functions and quantify our results for a range of metrics. We are able to achieve an average test SSIM of 0.8600, an improvement of more than 15% approximately over the SSIM score of blurred input images. For the case of LDR-HDR conversion, we consistently improve on all the three metrics- 23.94 average PSNR over LDR's 19.10, 0.902 average SSIM over LDR's 0.864 and 0.9947 UQI over LDR's 0.9741.

1 Introduction

Given the era where the mobile phones have become ubiquitous and revolutionized the art of photography, the issue of the low-quality of these pictures is prevalent and cannot be unseen. The main reason for the low-quality of images, being the size limitations of lens of this compact device. Many photography tools exist for the enhancement of these images but they focus on adjusting the image as a whole, i.e. the global parameters of the image like the brightness and contrast. This implies that major photo enhancement of images on the mobile phones is still restricted to manual image correction using specialized retouching software. Fortunately, this problem can be addressed using deep learning. We treat this problem as an image translation task and use the Pix2Pix architecture which is based on the Conditional Generative Adversarial nets to solve it.

Image translation is defined as the task of translating one possible representation of an image to another representation given sufficient information regarding the translation that has to be made,i.e, sufficient training data. The task of translation of image fundamentally involves predicting of pixels from pixels. To achieve our image translation task of enhancing the quality of phone images and Deblurring using the Pix2Pix architecture, we use the DPED Dataset. For the translation of Low Dynamic Range to High Dynamic Range, we use the LDR and HDR pair Dataset.

The architecture we use for the implementation of this project is Pix2Pix [7]. It is an architecture based on the conditional GAN where a target image is generated, conditioned over a given input image. The generator is trained using the perceptual, color , content and texture losses and the discriminator is trained using the Wasserstein loss. The job of the generator is to generate images that are very close in representation to the ones in training data and the job of the discriminator is to

distinguish if the image fed to it is from the training data or a fake one. GAN will obtain equilibrium when the discriminator is no longer able to distinguish between the image generated by the generator and the image in the training data. Therefore, it is essential to train the generator and discriminator simultaneously to obtain the results.

In section 2, we discuss the related works on Image de-blurring, conversion to High Dynamic range (HDR) for the enhancement of the images. In section 3, we talk about our methodology and in section 4, we provide results accumulated from our experiments, and in section 5 we discuss our results on the various experiments conducted by us for the enhancement of the mobile images. In section 6, we present some future works along with some conclusions drawn from our project.

2 Related Work

Image deblurring is a classic low-level computer vision problem in which the objective is to recover a sharp and high-quality image from a given blurred image. Blurring can occur in an image due to various reasons such as - poor camera quality, camera shake, out of focus objects, motion in captured objects etc. Several non-deep learning models already exist as a solution to this problem which either assume that the deblurring kernel is known [14] or it is unknown and do blind deblurring [2] on the image. Recent advancements in deep learning promise even better solutions to this problem. [11] uses a CNN based model for removing non-uniform motion-blur. In this work, we use a GAN based model for deblurring.

A High dynamic range (HDR) image is usually obtained by taking images of the same scene using three different exposures by varying the amount of light that goes through the lens [10]. These images are then combined into one by the image sensor. There have been many works which use a sequence of images to generate an HDR image. In [5] the authors propose a weighting function to combine images at different exposures. [12] generates high dynamic range (HDR) images from multi-exposed low dynamic range (LDR) stereo images. The vast majority of cameras in the market only capture a limited dynamic range of a scene by computing the disparity map between the stereo images. In this work we propose generating an HDR image from a single LDR image. Previously there have been various attempts at using neural networks to generate an HDR image from a single image. In [4] the authors use an autoencoder architecture to translate an LDR image into the corresponding HDR image, and in [9] three specialized CNNs are learned to replicate the HDR image formation pipeline in the camera. In this work we experiment on using GANs to generate an HDR image from a single LDR image. We experiment on the [8] which consists of 528 LDR-HDR image pairs.

3 Methodology

3.1 Pix2Pix Architecture

Pix2Pix is an architecture used for image to image translation that is based on conditional GANs. The generator is based on the U-net architecture. Unet consists of an encoder and decoder, where the encoder is a contracting part to capture context and the decoder is a symmetric expanding path to enable precise localization of pixels. Figure 2 shows the Pix2Pix generator architecture and Table 1 shows the details for each layer. The output of the generator are images that are of same dimensions as the input images.

For the discriminator, a PatchGAN is used. It models high-frequency structures by restricting its attention to the structure in local image patches. It penalizes structure at the scale of patches.Figure 1 shows the PatchGAN discriminator architecture and Table 2 shows the details for each layer. We pass the generated/target images along with the input image to the PatchGAN and the output corresponds to whether patches in the images are real or fake.

In Section 3.2 we describe the task of converting an phone image to dslr, in Section 3.3 we describe the process of deblurring an image and in section 3.4 we describe the task of converting and LDR image to HDR.

Layer	Layer Type	Input Channels	Output Channels	Kernel	Stride	Activation	Dropout	Batch Norm
1	Conv2d	3	64	4 X 4	2	Leaky	False	False
2	Conv2d	64	64X2	4 X 4	2	Leaky	False	True
3	Conv2d	64X2	64X4	4 X 4	2	Leaky	False	True
4	Conv2d	64X4	64X8	4 X 4	2	Leaky	False	True
5	Conv2d	64X8	64X8	4 X 4	2	Leaky	False	True
6	Conv2d	64X8	64X8	4 X 4	2	Leaky	False	True
7	Conv2d	64X8	64X8	4 X 4	2	Leaky	False	True
8	Conv2d	64X8	64X8	4 X 4	2	ReLu	False	False
9	UpConv2d	64X8	64X8	4 X 4	2	ReLu	True	True
10	UpConv2d	64X16	64X8	4 X 4	2	ReLu	True	True
11	UpConv2d	64X16	64X8	4 X 4	2	ReLu	True	True
12	UpConv2d	64X16	64X8	4 X 4	2	ReLu	False	True
13	UpConv2d	64X16	64X4	4 X 4	2	ReLu	False	True
14	UpConv2d	64X8	64X2	4 X 4	2	ReLu	False	True
15	UpConv2d	64X4	64	4 X 4	2	ReLu	False	True
16	UpConv2d	64X2	3	4 X 4	2	Tanh	False	False

Table 1: Generator Architecture Details

Layer	Layer Type	Input Channels	Output Channels	Kernel	Stride	Batch Norm	Activation
1	Conv2d	6	64	4 X 4	2	False	Leaky ReLU
2	Conv2d	64	128	4 X 4	2	True	Leaky ReLU
3	Conv2d	128	256	4 X 4	2	True	Leaky ReLU
4	Conv2d	256	512	4 X 4	1	True	Leaky ReLU
5	Conv2d	512	1	4 X 4	1	True	None

Table 2: Discriminator Architecture Details

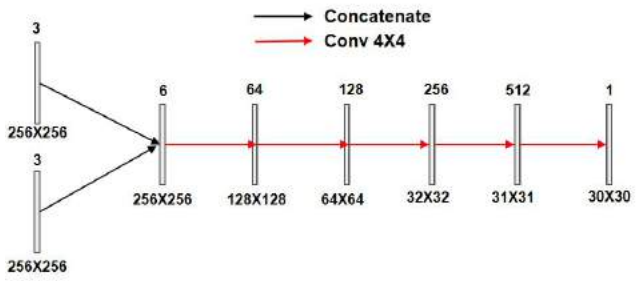


Figure 1: Pix2pix Discriminator architecture

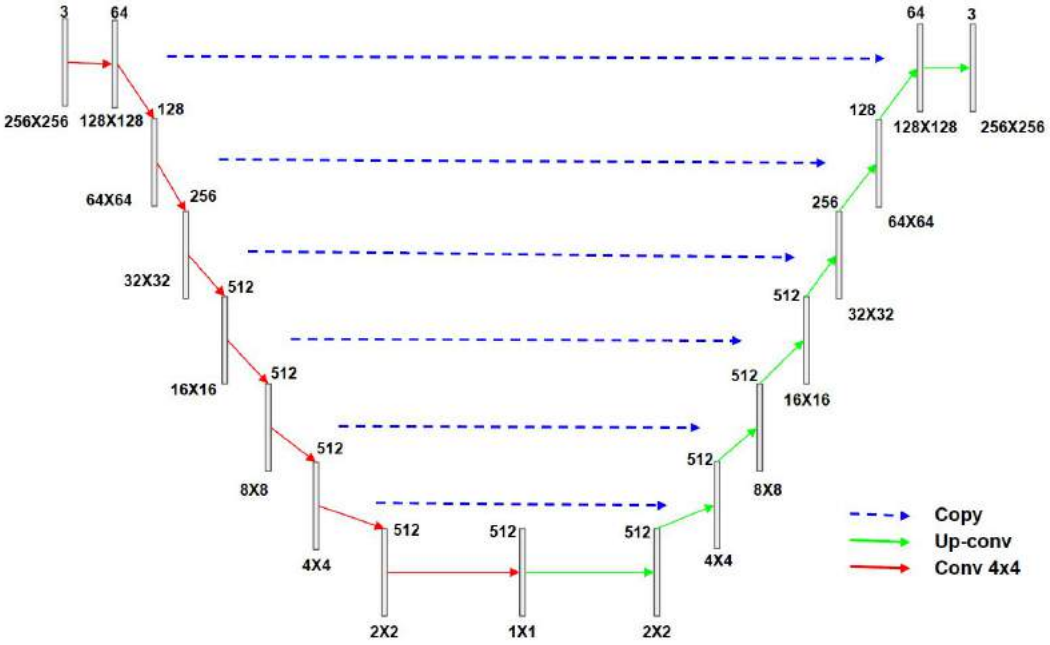


Figure 2: Pix2pix Generator architecture

3.2 Phone to DSLR Image-to-Image translation

In spite of the rapid improvement in the quality of smartphone cameras, their physical limitations - small sensor size, compact lenses and the lack of specific hardware, - impede them to achieve the quality results of DSLR cameras. We attempt to translate phone images to DSLR quality like images by making use of the Pix2Pix architecture. Figure 3 shows the stark difference between the quality of phone images and DSLR images.

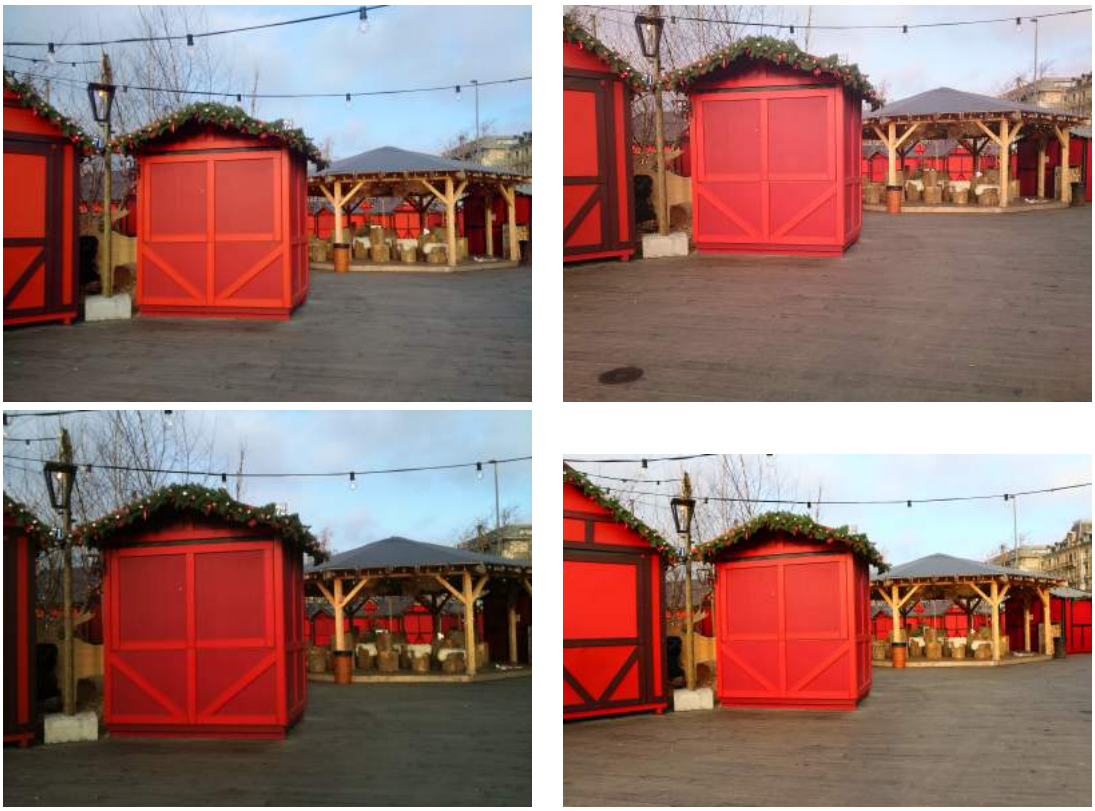


Figure 3: Photos captured by a blackberry phone (top left), sony phone (top right), iphone (bottom left) and DSLR (bottom right) of the same scene

3.2.1 Dataset and Preprocessing

We take the DPED dataset [6] and used the patch data that they have collected. These are 100X100 patches that correspond to the phone and DSLR images after aligning and rescaling. These patch images are z-score normalized with standard deviation 0.25 and mean 0.5 and then used further. Currently, we are training with 4994 pairs of phone and DSLR images, a validation dataset of 500 pairs and a test set as well of 250 pairs.

3.2.2 Architecture

We used a slightly modified version of the Pix2pix architecture described in the previous section. The generator used is the same as 2 without Layers 6,7,10,11. The discriminator is the same as described in 2. Also, unlike Pix2pix, we used Instance Norm layers in the discriminator instead of batch norm.

3.2.3 Training Details

Table 3 shows the hyperparameters used for training for the problem of phone to DSLR image translation. We trained the network on the patch data by resizing it to 64X64 images. The training was done for 600 epochs on 4994 training examples with a batch size of 32. We used 500 images for the validation set and 250 images for the test set. We experimented with a variety of loss functions for both the generator and discriminator

- For the discriminator, we experimented with Binary Cross Entropy Loss and Wasserstein Loss

- For the generator, we experimented with different permutations of L1 loss, perceptual loss, color loss, texture and content loss. These losses will be explained an the upcoming sections.

The model that gave us the lowest generator loss was saved and used for the test set.

	Optimizer	**Learning Rate**	**Batch Size**	$\beta 1$	$\beta 2$	**Image Size**
Generator	Adam	2e-4	32	0.5	0.99	64X64X3
Discriminator	Adam	2e-4	32	0.5	0.99	64X64X3

Table 3: Hyperparameter Settings for Phone to DSLR image translation

3.3 Deblurring

In this section we discuss how we perform deblurring on an image using Pix2Pix based Generative Adversarial model.

3.3.1 Dataset and Preprocessing

Figure 4: Left-Blurred Image and Right-Corresponding DSLR image (intended target)

In order to create our dataset, we take the DPED dataset [6] and filter out the DSLR images from it. Since the phone images present in this dataset are not exactly aligned with the DSLR images, we perform Gaussian blurring with $\sigma = 1$ on the DSLR images of this dataset in order to create our dataset of blurred images. The dataset images are z-score normalized with standard deviation 0.25 and mean 0.5 and then used further. Currently, we have made a training dataset with 2800 pairs of blurred and dslr images, a validation dataset of 50 pairs and a test set as well of 50 pairs. Fig. 4

3.3.2 Architecture

Table 1, Table 2 show the generator and discriminator architecture details of the Pix2Pix based conditional GAN model respectively that we use for deblurring of images. We use the conditional GAN loss for training the discriminator. For the training of generator, in addition to the conditional GAN loss we also use L1 loss and the perceptual loss as described in Section 3.4.

3.3.3 Training Details

Table 4 shows the hyperparameters used for training for the problem of deblurring. To enhance the training process, we apply data augmentation on-the-fly during training. We first randomly choose whether to augment the current input data. If the random choice is to augment the data, we apply

different augmentations such as random contrast, brightness and saturation. We observe that these augmentations impact the performance of the model positively. We experimented with tweaking several hyper-parameters including the weights given to various loss functions. G_loss was a sum of the generator's component of the conditional GAN loss, L1 loss and the perceptual loss with weights as follows:

$$G_{loss} = G_{cGAN\ Loss} + 100 * L1_{loss} + 1 * P_{loss}$$

We observed that our outputs had a bit less contrast compared to the target images and hence, at the end, as part of our post-processing step, we perform contrast enhancement as follows:

$$I(x,y) = \begin{cases} \frac{G(x)-G_{min}}{G_{max}-G_{min}}, & \text{if } G_{min} \leq G(x) \leq G_{max} \\ 0, & G_{min} \geq G(x) \\ 255, & G_{max} \leq G(x) \end{cases}$$

where $I(x, y)$ denotes our final output and $G(x)$ denotes the generator's output. G_{max} and G_{min} are set to 240 and 10 based on empirical results.

	Optimizer	**Learning Rate**	**Batch Size**	$\beta 1$	$\beta 2$	**Image Size**	**Epochs**
Generator	Adam	2e-4	8	0.5	0.99	256X256X3	400
Discriminator	Adam	2e-4	8	0.5	0.99	256X256X3	400

Table 4: Hyperparameter Settings for Deblurring

3.4 LDR To HDR Image-to-Image Translation

The prime goal of digital imaging techniques is to reproduce the realistic appearance of a scene. Low Dynamic Range (LDR) cameras are incapable of representing the wide dynamic range of the real-world scene. The captured images turn out to be either too dark (underexposed) or too bright (overexposed). This also makes synthesizing high dynamic range (HDR) images from multiple low-dynamic range (LDR) exposures is challenging problem. We formulate this problem as an image-to-image (I2I) translation task. To this end, we present a conditional GAN based framework trained in an end-to-end fashion over multi-exposed LDR images and correponding HDR images as described in upcoming sections to synthesize HDR images for sampled LDR images.

3.4.1 Dataset and Preprocessing

The LDR and HDR pair Dataset [8] dataset consists of LDR (Low dynamic range) and the corresponding HDR (High dynamic range) pairs of various everyday scenes. It consists of an overexposed (+2) , underexposed (-2) and normally exposed (+0) LDR images and the corresponding HDR image obtained by combining the above images. All the 4 images of a particular scene are taken from the same position which ensures that there is an alignment between the images. Sample LDR-HDR images are shown in Fig 5. The dataset consists of 176 scene images with each scene having 3 LDR and 1 HDR image.

Figure 5: LDR images for two scenes at -2,0,+2 exposure levels along with the corresponding HDR image

HDR images have a different range of pixel values than the corresponding LDR images. LDR images are stored using 8-bit floating point values and HDR using 10-bit floating point values. Each LDR and HDR image is normalized independently and converted to Pytorch tensors. We also resize the images to 256x256 for faster processing. Each LDR and the corresponding HDR image is treated as a separate pair. Hence we obtain a total of 3x176 = 528 input output pairs. The pairs corresponding to 20 scenes (60 pairs) are treated as test images and the rest are split into 80:20 ratio as train and validation images. We thus ensure that the test image scenes are independent of the train or validation scenes.

3.4.2 Architecture

The architecture used to tackle the image-to-image translation problem of LDR to HDR image conversion is the pix2pix based conditional GAN architecture for paired images as described in section 3.1. By this architecture, we are able to create a LDR to HDR image generator without explicitly feeding information about the conversion process, leaving it to the GAN structure to learn based on the training images. We made a few tweaks to the original architecture to aid the generator model to learn and successfully generate HDR images for given LDR images. For this task, we changed the output activation layer to ReLU to cater to the HDR image tensors, since they are greater than 1. We also switched the Batch Normalization layer used in U-net generator and PatchGAN discriminator with a Instance Normalization for normalizing each element of batch independently.

3.4.3 Training Details

For training our GAN model, we feed LDR images to the generator (U-net in our case) which generates a fake HDR image. This fake HDR image along with the corresponding real HDR image is then fed to PatchGAN discrminator to distinguish between fake image and real HDR image. The loss of the discriminator is propagated back to the generator which then learns to produce better quality images to fool the discriminator. We experiment with the Binary Cross Entropy(BCE) and the Wasserstein loss functions for the discriminator model. For the generator, we use loss functions such as L1 loss, Colour loss, Perceptual loss and Style loss in combination with Wasserstein or BCE loss as objective functions, described in more detail in section 3.5. After each epoch of training process as described above, we also validate the accuracy of our generator and discriminators on a

validation set of images to visualise the improvement in performance of our model. We also vary several hyper-parameters other than loss functions like number of epochs, learning rates and batch sizes per epoch to extract the best performance out of our model.

3.5 Losses

In this subsection, we discuss about the various losses that we used for training the generator and discriminator. The generator tries to generate images as close as possible to the target image. Since our task involves enhancing the image quality, we use losses such as perceptual loss, color loss, texture loss, content loss to encourage the generator to produce high quality images. For the discriminator, we started off by using a Binary Cross Entropy loss. Then, we switched to the Wasserstein loss to make the training more stable. The rest of this section provides details regarding the losses that we used.

3.5.1 Perceptual Loss

For achieving a high resolution on the generated image, the easiest way is to calculate the pixel-wise loss, i.e., the Mean Squared Error between the generated image and the DSLR captured image. However, the ability of Mean squared error loss to capture the perceptual differences is limited due to its pixel-wise calculation of the error. To overcome this limitation, we use the VGG perceptual loss for the generator to capture all the perceptual intricacies such as the Content loss and the Adversarial loss.

$$l^{SR} = \underbrace{\underbrace{l^{SR}_{\mathrm{X}}}_{\text{content loss}} + \underbrace{10^{-3} l^{SR}_{Gen}}_{\text{adversarial loss}}}_{\text{perceptual loss (for VGG based content losses)}}$$

$$l^{SR}_{MSE} = \frac{1}{r^2 W H} \sum_{x=1}^{rW} \sum_{y=1}^{rH} \left(I^{HR}_{x,y} - G_{\theta_G}\left(I^{LR}\right)_{x,y} \right)^2$$

$$l^{SR}_{VGG/i.j} = \frac{1}{W_{i,j} H_{i,j}} \sum_{x=1}^{W_{i,j}} \sum_{y=1}^{H_{i,j}} \left(\phi_{i,j}\left(I^{HR}\right)_{x,y} - \phi_{i,j}\left(G_{\theta_G}\left(I^{LR}\right)\right)_{x,y} \right)^2$$

where $W_(i,j)$ and $H_(i,j)$ represent the dimensions of the respective feature maps within the VGG network.

3.5.2 Color Loss

To further accentuate the capturing of all perceptual similarities, we go on to add the colour loss to our base perceptual loss stated above. For calculating the colour loss, gaussian blur is applied to the generated image and the DSLR captured image and the Euclidean distance is computed between the obtained distances.

$$\mathcal{L}_{\text{color}}(X, Y) = \|X_b - Y_b\|_2^2,$$

where X_b and Y_b are the blurred images of X and Y, resp.:

$$X_b(i,j) = \sum_{k,l} X(i+k, j+l) \cdot G(k,l)$$

and the 2D Gaussian blur operator is given by

$$G(k,l) = A \exp\left(-\frac{(k-\mu_x)^2}{2\sigma_x} - \frac{(l-\mu_y)^2}{2\sigma_y} \right)$$

where A is a constant that factors the expression and $\mu_{x,y} = 0$ and $\sigma_{x,y} = 3$

3.5.3 Texture Loss

The texture loss helps us in transforming the generated image to the texture of the DSLR captured image. We use the texture consistency loss between the generated image and the DSLR image to obtain the contrast in colours by training the genrator to imitate the style of the DSLR captured image

$$\mathcal{L}_{\text{texture}} = -\sum_{i} \log D\left(F_{\mathbf{W}}\left(I_s\right), I_t\right)$$

where $F_{\mathbf{W}}$ and D denote the generator and discriminator networks, respectively.

3.5.4 Content Loss

The content consistency loss is calculated between the generated image and the DSLR captuted image to enforce the model to maintain the content feature of the input image after being translated. This loss encourages them to have similar feature representation that comprises various aspects of their content and perceptual quality

$$\mathcal{L}_{\text{content}} = \frac{1}{C_j H_j W_j} \left\|\psi_j\left(F_{\mathbf{W}}\left(I_s\right)\right) - \psi_j\left(I_t\right)\right\|,$$

where C_j, H_j and W_j denotes the number, height and width of the feature maps, and $F_{\mathbf{W}}\left(I_s\right)$ the enhanced image.

3.5.5 Wasserstein Loss

The Conditional GAN model that we have used(UNET+PatchGAN) is difficult to train because it involves training a generator and discriminator simultaneously in a two player zero-sum game. It is difficult to maintain an equilibrium between the generator and discriminator. Instead of distinguishing between real and fake patches from the image, the loss at discriminator is modified to effectively measure how close is the model distribution to the real distribution. To do this, the Wasserstein distance [1] is used which is an approximation of the Earth Mover (EM) distance. Using this loss, training the GAN can be done without the need to balance the generator and discriminator. This also reduces the mode collapse which exists in the GAN. The Wasserstein distance is given by

$$W\left(P_r, P_\theta\right) = \sup_{\|f\|_L \leq 1} \left[E_{x \sim P_r}[f(x)] - E_{x \sim P_\theta}[f(x)]\right]$$

where f refers to the discriminator(critic) function, P_θ is the generator probability distribution, P_r is the real probability distribution. Note that the f should be 1-Lipschitz continuous and this is enforced by weight clipping. After after each gradient update, we clip w range to [0.01, 0.01]. This constraint can also be satisfied by making use of gradient penalty.

3.6 Evaluation Metrics

The goal of the project is to enhance the quality of images captured. The quality of the image is generally subjective, but we use objective methods to evaluate the image quality. We studied several GAN evaluation metrics like average log-likelihood using Parzen window estimation, Maximum Mean Discrepancy etc [3]. Most of the metrics were related to evaluating the probability distribution of the class of the image being generated. In our project, the focus is on improving the quality of images. So, the evaluation metric should be reflective of that. We measure how similar is the quality of the generated image to that of the target reference image. The three evaluation metrics we use are Peak Signal-to-Noise Ratio (PSNR), Structural Similarity Index Method (SSIM) and Universal Image Quality Index (UQI).

3.6.1 Peak Signal-to-Noise Ratio (PSNR)

Peak signal-to-noise ratio (PSNR) is the ratio between the maximum possible value power of a signal and the power of distorting noise that affects the quality of its representation. The PSNR is generally expressed in a decibel scale. The higher the PSNR, the closer it is to the DSLR image (in terms of quality).

For two monochrome images I and K, $PSNR$ is given by

$$\begin{aligned} \mathrm{PSNR}(I,K) &= 10\log_{10}\left(\frac{MAX_I^2}{MSE}\right) \\ &= 20\log_{10}\left(MAX_I\right) - 10\log_{10}\left(MSE_{I,K}\right) \end{aligned} \tag{1}$$

where

$$MSE_{I,K} = \frac{1}{mn}\sum_{i=0}^{m-1}\sum_{i=0}^{n-1}(I(m,n) - K(m,n))^2$$

Our dataset has 3 images with channels - R,G and B. So, we calculate the PSNR for each channel separately and average it.

$$PSNR(I,K) = \frac{PSNR(I_r,K_r) + PSNR(I_g,K_g) + PSNR(I_b,K_b)}{3}$$

3.6.2 Structural Similarity Index Method (SSIM)

SSIM measures the structural similarity between a pair of images. By comparing the similarity between images generated with the corresponding DSLR image, we obtain a quantitative estimate about the quality of the generated image.

SSIM compares the luminance (I), contrast(C) and Structure(S) of two images x and y by comparing corresponding pixels and their neighborhoods.

$$I(x,y) = \frac{2\mu_x\mu_y + C_1}{\mu_x^2 + \mu_y^2 + C_1} \quad C(x,y) = \frac{2\sigma_x\sigma_y + C_2}{\sigma_x^2 + \sigma_y^2 + C_2} \quad S(x,y) = \frac{\sigma_{xy} + C_3}{\sigma_x\sigma_y + C_3}$$

where $\mu_x, \mu_y, \sigma_x, and \sigma_y$ denote mean and standard deviations of pixel value in a window centered at either x or y. σ_{xy} denotes the correlation coefficient between corresponding pixels in the window. C1, C2, and C3 are constants for numerical stability. SSIM is given by a combination of the three values

$$SSIM(x,y) = I(x,y)^\alpha C(x,y)^\beta S(x,y)^\gamma$$

We use $\alpha = \beta = \gamma = 1$ in our implementation.

SSIM assumes a fixed viewing distance.

Universal Image Quality Index (UQI) UQI [13]is a full reference method of image fidelity as it takes into account the error between a distorted image and a reference image and tries to quantify an error incorporating a set of known properties of human visual system. UQI takes into account three factors for modelling any image distortion. The three factors are: 1.loss of correlation 2. luminance distortion and 3. contrast distortion. UQI provides meaningful comparison across different types of image distortion.

Let **x** and **y** be original and test images respectively(**x**= $\{x_i \mid i{=}1, 2, \cdots, N\}$ and **y**= $\{y_i \mid i{=}1, 2, \cdots, N\}$). UQI is given by the formula:

$$Q = \frac{4\sigma_{xy}\bar{x}\bar{y}}{\left(\sigma_x^2 + \sigma_y^2\right)\left[(\bar{x})^2 + (\bar{y})^2\right]}$$

where

$$\bar{x} = \frac{1}{N}\sum_{i=1}^{N} x_i, \quad \bar{y} = \frac{1}{N}\sum_{i=1}^{N} y_i$$

$$\sigma_x^2 = \frac{1}{N-1}\sum_{i=1}^{N}\left(x_i - \bar{x}\right)^2, \quad \sigma_y^2 = \frac{1}{N-1}\sum_{i=1}^{N}\left(y_i - \bar{y}\right)^2$$

$$\sigma_{xy} = \frac{1}{N-1}\sum_{i=1}^{N}\left(x_i - \bar{x}\right)\left(y_i - \bar{y}\right)$$

The dynamic range of Q is[-1,1]. The best value of 1 is achieved when the test image and original image are equal and the worst value of -1 is achieved when test image is twice the mean of original image subtracted by the original image.

This equations can be re-written as follows to get a better understanding of the three factors mentioned above.

$$Q = \frac{\sigma_{xy}}{\sigma_x \sigma_y} \cdot \frac{2\bar{x}\bar{y}}{(\bar{x})^2 + (\bar{y})^2} \cdot \frac{2\sigma_x \sigma_y}{\sigma_x^2 + \sigma_y^2}$$

The first part of the equation is the correlation coefficient between **x** and **y** with dynamic range [-1,1]. It measures the degree of correlation between x and y. There could be relative distortions between x and y even if they are linearly related. These are evaluated by the second and third parts of the equation. The second part tells how close the mean luminance is between x and y. It is 1(best value), if and only is $\bar{x}$=$\bar{y}$. $\sigma_x and \sigma_y$ are the estimate of contrast of x and y. Therefore the third part tells how similar the contrasts of the image are. Its dynamic range is [0,1].

The index is calculated by a sliding window approach and the final UQI is given by:

$$Q = \frac{1}{M} \sum_{j=1}^{M} Q_j$$

where Q_j is the UQI of each window

4 Results

This section discusses the results obtained for the three tasks. We have provided graphs for the discriminator and generator losses for our best models in each task. Then we show some images where the model performed well and some cases where the model performed poorly. Finally, we provide the SSIM, PSNR and UQI scores for the best model in each of our tasks.

4.1 Phone to DSLR Image Translation

After trying the different combinations of loss functions for the generator and discriminator, here are our findings:

- For the discriminator, using Wasserstein loss was better than Binary cross entropy(BCE) loss because it was better at achieving better equilibrium between generator and discriminator. We saw that BCE loss caused the discriminator to become too good very quickly.
- For the generator, the following combination of losses worked well

$$\text{Gen\_loss} = 2*(3*\text{Perceptual Loss} + 0.15*\text{Color Loss} + 17*\text{Texture Content Loss}) + \text{Disc Fake Loss}$$

Figure 6 shows the loss plots for the generator and discriminator when using the above loss. As you can see, the loss plot for the discriminator oscillates and increases, whereas the generator loss goes down.

Figure 7 shows the images after converting from the phone image to the DSLR image. These are some good examples wherein the the generator was able to brighten some of the dark colors in the original phone image.

Figure 8 shows some failure cases. Our model was unable to reduce the sun flare in one picture, whereas in the other it is unable to distinguish between fine differences in the shades of red

Table 5 shows the SSIM, PSNR, UQI scores of our generated images when compared with the DSLR images, and also when the original Sony phone images are compared to the DSLR images. As you can see, our model did not improve the phone images. The reason for this will be discussed in Section 5.

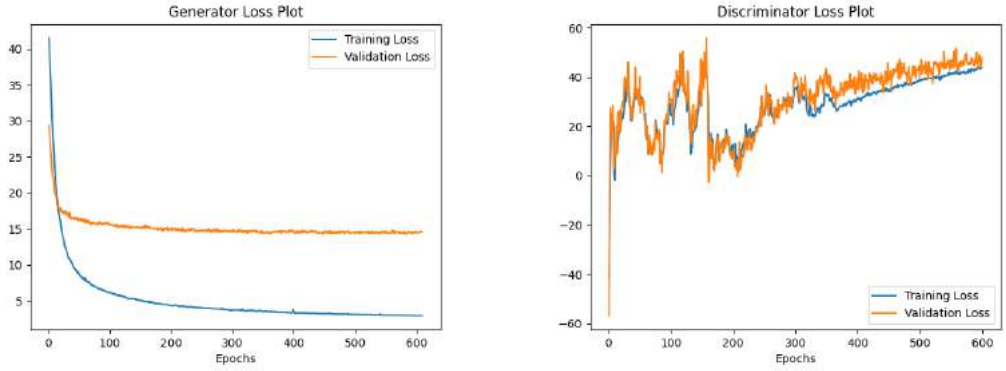


Figure 6: Generator Loss plot (on left) and the discriminator loss plot (on right) obtained for Phone to DSLR image translation model

Figure 7: Phone images (on left), their corresponding deblurred versions (in middle) obtained from our GAN model and their DSLR ground truth images (on right)

Image	PSNR	SSIM	UQI
Generated	19.5084	0.8206	0.8807
Sony Image	**22.9185**	**0.8753**	**0.9343**

Table 5: Evaluation Metric Scores for HDR image conversion

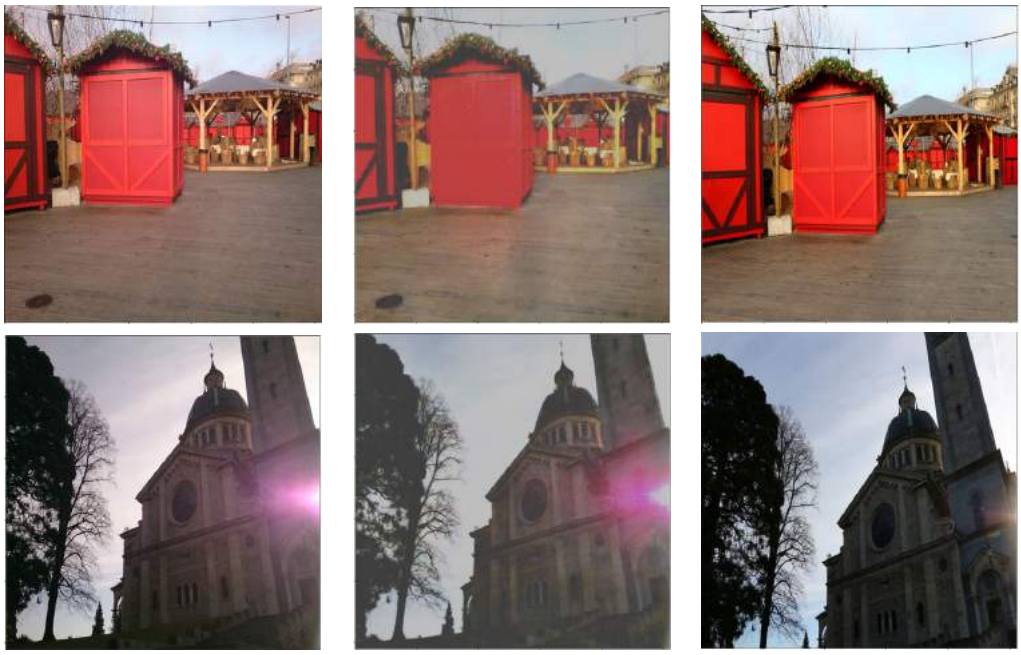


Figure 8: Failure Cases : Phone images (on left), their corresponding deblurred versions (in middle) obtained from our GAN model and their DSLR ground truth images (on right)

4.2 Deblurring Results

Fig. 9 show the results of our model's deblurring applied on the test images for four different scenes. Fig. 10 shows the loss plots observed. Here, we have only plotted the values for the first 100 epochs for better visualization. Fig. 11 provide few examples where our model fail. Table 6 shows the metric scores for our model.

Image	PSNR	SSIM	UQI
Blurred	22.68	0.7028	0.9754
Generated	20.32	0.8600	0.9257

Table 6: Evaluation Metric Scores for Deblurring

4.3 HDR Results

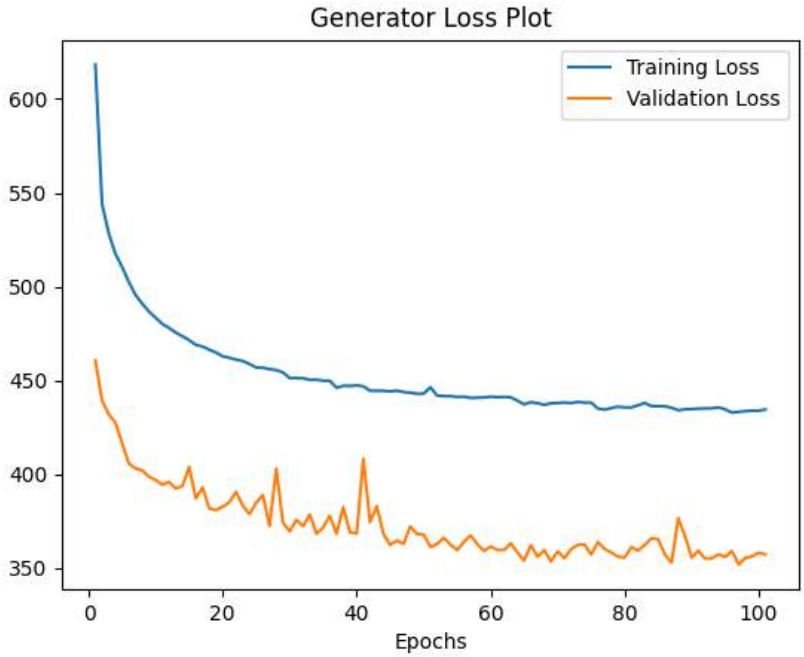
We performed several experiments while training our GAN model as described in section 3.4 to produce improvements on baseline model with BCE loss function so that the generator can generate an HDR image from given LDR image efficiently. We trained our models with different combinations of loss functions as described in section 3.5 to arrive at the best objective function for both generator and discriminator in our model, We also parameterized the combination of loss functions and perform experiments on relevant corresponding parameters. Also, we varied several hyperparameters like number of epochs to train our model, learning rates to update the weights in our model, batch size use while training and try to achieve our aim of LDR to HDR image conversion. We finally chose a model, with BCE and L1 loss trained for 400 epochs that gave us the best results in terms of training and validation losses. We saved our best model and used it to generate HDR images from given LDR images from test dataset. Figure 12 shows the preliminary results obtained during training on our validation dataset. Figure 14 shows some test cases where our model produced significantly poor quality results. Figure 13 shows some test cases where our model produced good quality results. We have also provided metric scores obtained from our best model on HDR results in table 7. We have provided the training and validation loss graphs obtained during training for our

Figure 9: Blurred images (on left), their corresponding deblurred versions (in middle) obtained from our GAN model and their DSLR ground truth images (on right)

best model over 400 epochs in Figure 15. More discussion on our results will follow in section 5 below.

Image	**PSNR**	**SSIM**	**UQI**
LDR	19.10	0.864	0.9741
Generated	**23.94**	**0.902**	**0.9947**

Table 7: Evaluation Metric Scores for HDR image conversion



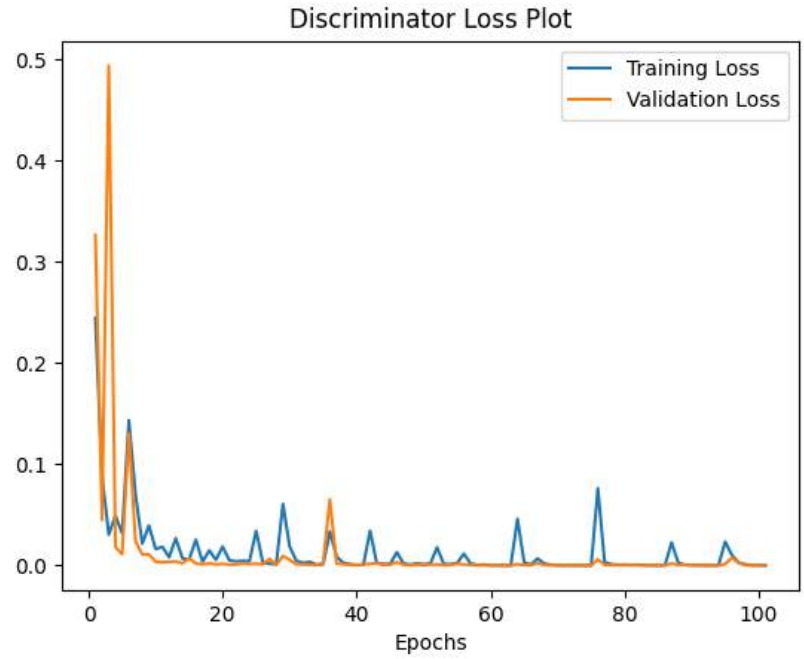


Figure 10: Loss plots for Generator and Discriminator for the case of deblurring (showing plots for first 100 epochs (out of 400) only for better visualization)

5 Discussion

This section explains our results that we presented earlier

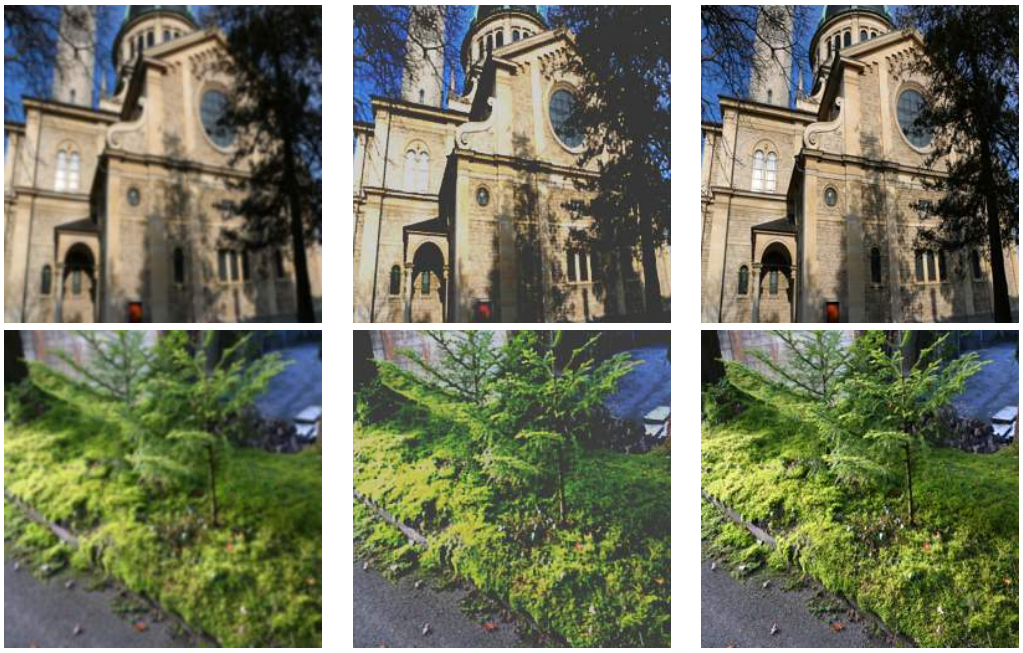


Figure 11: Failure examples for the case of deblurring-Blurred images (on left), their corresponding deblurred versions (in middle) obtained from our GAN model and their DSLR ground truth images (on right). Notice the extra shine leading to colour deviation in the first image and loss of details in the light-green part in the second image

5.1 Phone to DSLR Image Translation

In the results section, we started that using Wasserstein loss for the discriminator was better than using Binary Cross Entropy Loss. The reason why this works well is because BCE limits the values between 0 and 1 whereas Wasserstein doesn't have any such fixed range. So, what we observed with BCE was that the discriminator was getting better and the generator was not able to catch up. Since BCE was limits the values, the gradient received by the generator from the discriminator started vanishing. So, after one point, the discriminator was practically useless to the generator. So, making use of Wasserstein loss made the GAN training more stable. The generator started getting sufficient feedback from the discriminator.

For the generator, we first used L1 loss and our observation was that it was not able to capture the things which are needed to improve an image quality - which is color, perception, texture and content. In order to better capture this, we used separate loss functions to enhance each one of them, and them combined them to form the generator loss. This significantly improved our SSIM from around 0.75 with L1 loss to 0.82.

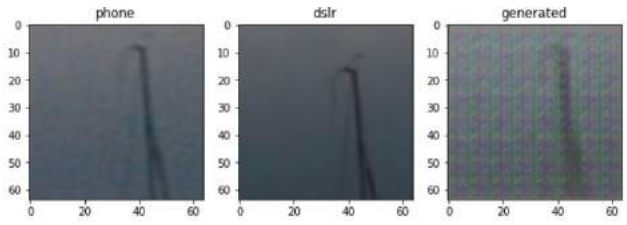


Figure 16: Unaligned Input Image (left) and Target image (middle) leading to poor generated image(right)

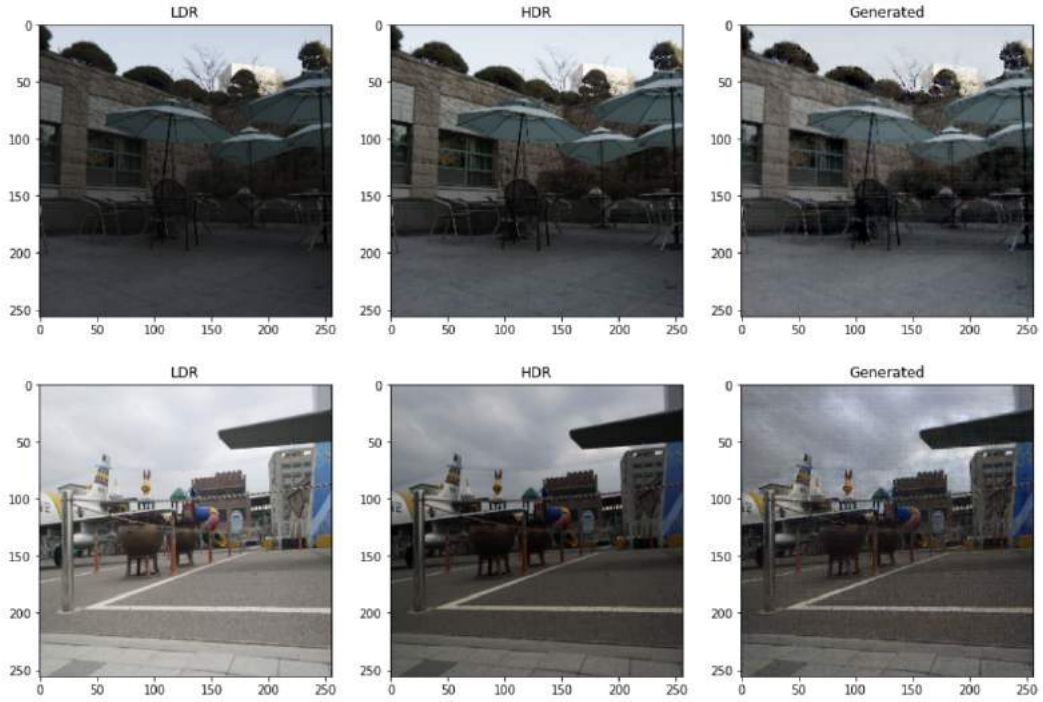


Figure 12: Preliminary validation results showcasing the real LDR, HDR and generated images obtained from our GAN model

As seen in table 5, our model does not improve the image quality. The main reason behind this is the lack of alignment of the input and target images. The patch dataset is supposed to be completely aligned, but we observed that there were many instances where they were not properly aligned. Fig 16 shows one such case. This happens because we used the U-net architecture for the generator and it has skip connections. This means that it is sensitive to misalignment. Since some images are aligned, and some aren't this causes lowers the sharpness and color of the generated images - this becomes blatant in small structures such as leaves and tree branches. In case the patch data was perfectly aligned, the model would have probably fared better.

5.2 Deblurring

Table 6 indicates that with respect to the target test images, the PSNR and UQI scores of the deblurred test images generated by our model are lower than the corresponding blurred input images. However, SSIM score for the deblurred images generated by our model is significantly higher than the corresponding blurred input images. This observation is intuitive and follows the idea that the PSNR and UQI metrics are not as applicable to the task of deblurring as they are to the other tasks. The expression of PSNR metric suggests that PSNR inherently depends on mean square loss, which is computed on a per-pixel basis rather than operating on a local context. MSE for blurred images is lower than the generated images since it tends to take a conservative approach by favoring smoothed image pixel values. Alternatively, as the name of PSNR suggests, since the noise is removed in the blurred images, the denominator of the argument of the log function, MSE, is lower and hence the value of PSNR is higher. SSIM on the other hand, takes the structure around pixels into account while computing the score rather than just operating on a per-pixel basis. As we can observe in the blurred input images, the details around a pixel are not clear at all and hence the different components of the SSIM metric for blurred images are lower, resulting in a lower SSIM score. As far as UQI is concerned, it is a special case of SSIM in which the numerical stability constant terms in the expression of SSIM are set to 0. This makes UQI an unreliable metric as the scores are susceptible to blowing up if the terms in the denominator tend to be very close to zero. In fact, if few

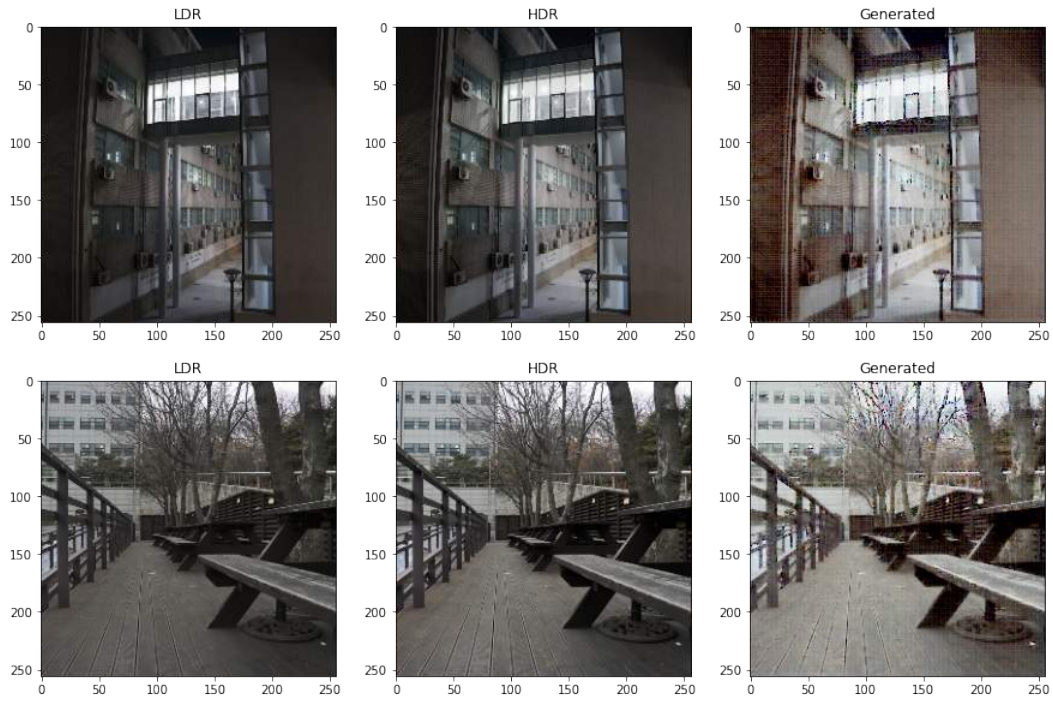


Figure 13: Good test results showcasing the real LDR, HDR and generated images obtained from our GAN model

select images from test set have the terms in the denominator close to zero, then the average value of UQI for the entire test set can shoot up significantly. Hence, the scores are not truly depictive of the improvement in the quality of the images. This is the reason why SSIM comes out as the most appropriate metric for the task of deblurring. Note that without the contrast enhancement step also, we get much better SSIM scores. We add contrast enhancement as a post-processing step only to match the visual quality of our generated images with that of the target DSLR images.

From our training experiments, we observed that the training of the generator and discriminator was much more stable when we used conditional GAN loss as mentioned in [7]. We tried out Wasserstein GAN as well as Gradient Penalty on top of it. We observed that the training was a bit unstable in the case of Wasserstein GAN [1] surprisingly. This was partly because we were not able to use the gradient penalty term in the discriminator loss as suggested since the computational power became a bottleneck for us and WGAN without the penalty term didn't help in improving the performance. Not only does using gradient penalty in the discriminator loss require greater GPU memory, but the training time also shoots up significantly. Fig. 10 shows the trend of the loss curves for a portion of the training. From the figures, it is evident that the generator loss curves show a decreasing trend for both the training and validation curves, which is what we expect. However, the discriminator loss curves also show a decreasing trend which is counter-intuitive since the loss curves should ideally increase in case the training of the generator went well. Even though the trend of the discriminator loss curves is that of gradual decrease, we can still see that there are spikes in the discriminator loss curves as the the training progresses. This indicates that the generator is able to fool the discriminator at regular intervals and still the generator keeps improving as time progresses, which is what we want. But the spikes dampen as time progresses which means the generator learning decreases with time and there comes a point when the generator stops learning altogether.

Fig. 9 shows the input images in the test set which gave good results. It is evident that the details which were missing in the blurred input images are now captured relatively better in the images generated by our model. These images look more like the ground truth images which are the original

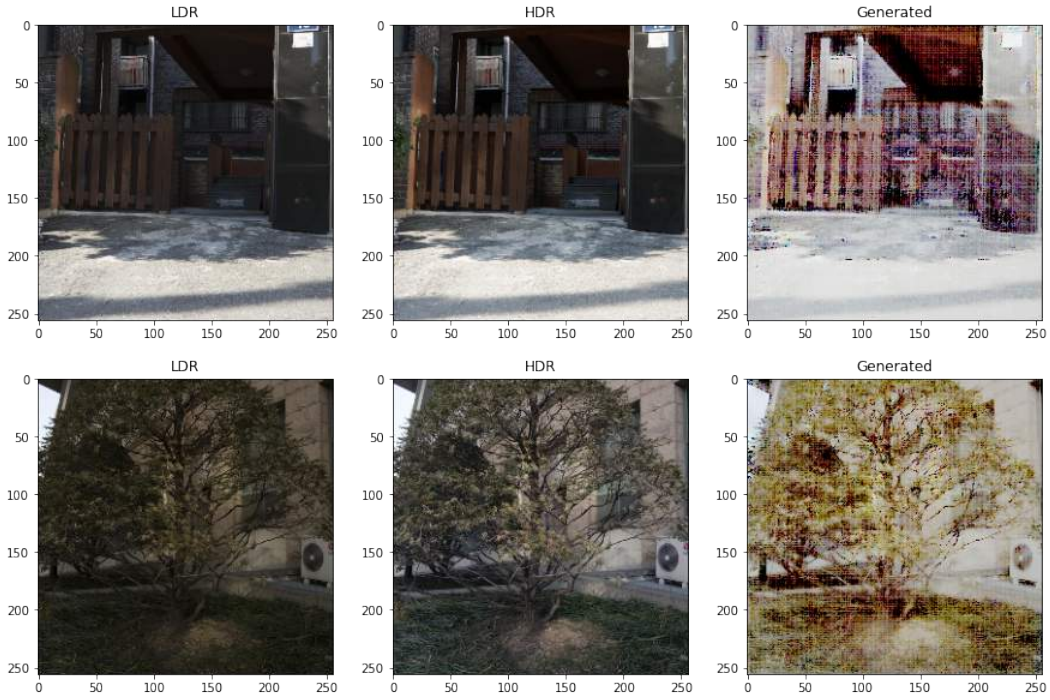


Figure 14: Poor test results showcasing the LDR (left), HDR(right) and generated images(middle) obtained from our GAN model

images before blur operation was applied. Fig. 11 shows sample images for which the generated images are relatively different from the ground truth images compared to the blurred images. In the example on the top, we can observe that the color of the building in the ground truth image (right image) matches with the blurred input image (left image) better than the generated image (middle image). In the next example, we can see that the generated image loses details (top left portion) which are visible in the ground truth image. While the input blurred image also does not have clear details, we do not quite have the intended result for the generated images. In most of the images, we also observe that the sky seems to have a dull appearance and in some case different shade compared to the sky region in the blurred input image and the ground truth image. We know that this problem can be solved by tweaking the loss function appropriately. We plan to take care of these issues in our future work.

5.3 LDR to HDR conversion

For the problem of converting low dynamic range images to high dynamic range images, we performed several experiments are described in section 4. And from those experiments, we obtained some interesting results. As we can infer from Table 6, our best model for LDR to HDR conversion gave better scores for all three metrics i.e. SSIM, PSNR and UQI. We can conclude that the model used for conversion of images to HDR produced better results than the similarity between original LDR and HDR images given by the SSIM score, which also signifies that our generator was able to learn to generate HDR images similar to original HDR images, which was the aim of our experiments.

While performing experiments with several loss functions, we observed that conditional GAN loss [7] gave better results instead of Wasserstein GAN loss as mentioned in [1], since for training with wasserstein loss we didn't utilize the gradient penalty because we had limited GPU power to train our models. We also used perceptual loss which was giving promising results for early epochs and losses were decreasing but eventually L1 loss helped the generator and discriminator to learn better and reduce losses over several epochs and hence lead to good results. From figure 12, we can see

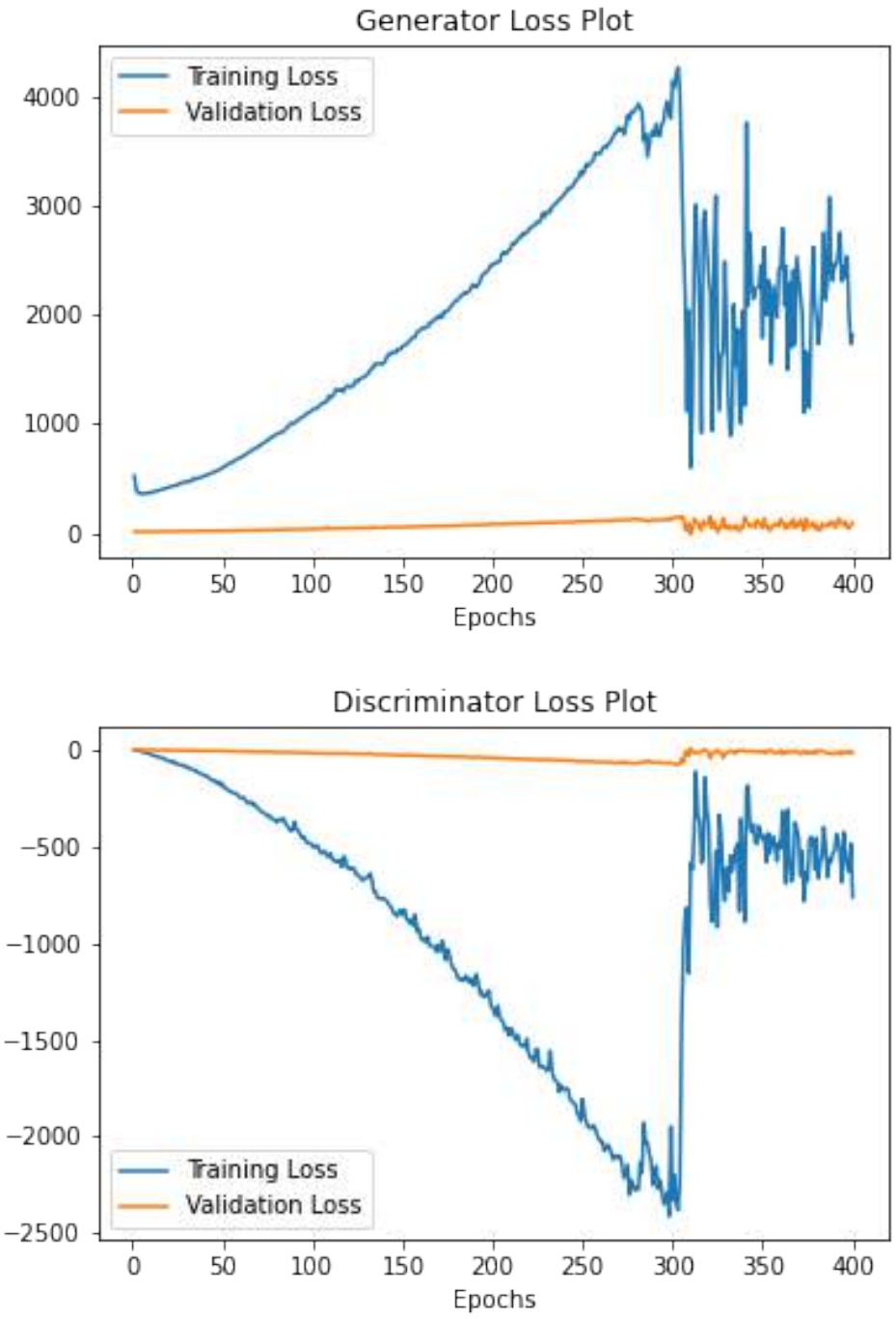


Figure 15: Loss plots for generator and discriminator for the case of LDR to HDR conversion trained for 400 epochs.

that during the period of training our best model with L1 loss provides generated images that learns the fine details of HDR images like clouds which cannot be deciphered from LDR images. So, we know that at this point, our generator is learning tthe task of conversion to HDR and is on the right track. After training is finished, we test our model on test set and observe some promising results

as shown in figure 13. But this is not always the case as for some test cases, our generator overfits the features and produces images worse that current LDR images. For example in figure 14, we can see that our generator model has given more weight to the redness of an image which was not an important feature of HDR image, but is overshadowing the generated image due to generator overfitting that feature of the image.

From figure 15, we can discover that for our best model, until 300 epochs the generator is still trying to fool the discriminator but is not successful and the discriminator loss is decreasing epoch by epoch as generator is not able to produce good hdr images. Then, after 300 epochs, we see that the generator has now learned to fool the discriminator is producing images that fools the discriminator, hence the discrimiator losses are increasing significantly from previous epochs and decreasing loss for generator vice versa. At this point, we know that our generator is capable to produce HDR images from given LDR images that can fool the discriminator. So, we can test this model on our test set which provided results as shown in figures 14 and 13.

We can decrease the probability of bad images produced from our trained models by using appropriate combination of loss functions trained over a lot of epochs that will help the generator learn the right features for conversion process. We plan to do that as a part of our future work.

6 Conclusion and Future Works

In this work, we have explored multiple approaches for the task of Image Enhancement. Based on our encouraging results, we feel that GANs can be a good post-processing tool to enhance photos taken from mobile phones or other low quality cameras. We acknowledge the impediments faced during our experiments in all verticals - data, model, loss, and evaluation - and look to overcome the challenges in the future. To achieve superior results for direct translation from phone to DSLR image, we propose to create an aligned dataset which could be used to train our model. We expect significant improvement in the metrics for phone to dslr translation once this task is achieved. As part of our future goals, we intend to explore various architectures such as StackGANs, Progressive GANs, and CycleGANs. We feel that reconstructing images at lower scales and going higher up progressively will positively impact the results. Due to limited compute/memory, we were not able to leverage the use of WGAN loss as much as we would have liked. We plan to train our models on powerful GPUs with sufficient memory so that gradient penalty can be incorporated into the loss function expression for the discriminator. We feel there is significant scope in improving the evaluation of our models. We plan to work on visualizing what our models are actually learning to understand the internal workings better.

7 Individual Contributions

We worked through all the problems together and discussed issues about how to implement and fine tune the models. Our individual contributions are as mentioned below:

7.1 Atishay

Worked on the deblurring task. Implemented the Pix2Pix model and performed several experiments to discover the best model with the best hyperparameters for our task. Created the entire dataset for the task of deblurring along with the implementation of data augmentation and pre-processing steps. Tried out the Gradient Penalty method also. Wrote corresponding parts in reports.

7.2 Chaitanya

Worked on the task of deblurring. I implemented the baseline Pix2Pix model and conducted many experiments to find the best hyperparameters and the best model. I implemented Wasserstein GAN and tried experimenting with it as well. I also implemented the contrast-enhancement post-processing part. I wrote the corresponding parts in report along with the related works on deblurring.

7.3 Purva

Worked on LDR to HDR image conversion. Impelmented pix2pix model and performed several experiments varying loss functions and other hyperparameters to discover the best model for this problem of LDR to HDR conversion and wrote corresponding parts in report.

7.4 Shivaank

Worked on LDR to HDR image conversion. Impelmented pix2pix model and performed several experiments varying loss functions and other hyperparameters to discover the best model for this problem of LDR to HDR conversion and wrote corresponding parts in report.

7.5 Satvik

Worked on the problem of deblurring. Implemented the Pix2Pix model and performed many runs to discover the best model for this problem. Tried out the various loss functions and analysed in detail which loss functions were suitable for the case of deblurring problem. Wrote the corresponding parts in report.

7.6 Shreyas

Worked on Phone image to DSLR image conversion and wrote the corresponding parts in the report. Worked on the baseline code for the Pix2pix architecture and performed several experiments with Wasserstein loss function and some generator loss function and other hyperparameters.

7.7 Madhumitha

Worked on Phone image to DSLR image conversion task and wrote the corresponding parts in the report. Worked on the code for generator loss functions and performed several experiments with the generator loss functions. Worked on the code for the evaluation metric UQI.

References

[1] Martin Arjovsky, Soumith Chintala, and Léon Bottou. "Wasserstein Generative Adversarial Networks". In: *Proceedings of the 34th International Conference on Machine Learning - Volume 70*. ICML'17. Sydney, NSW, Australia: JMLR.org, 2017, pp. 214–223.

[2] Yuval Bahat, Netalee Efrat, and Michal Irani. "Non-Uniform Blind Deblurring by Reblurring". In: *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*. Oct. 2017.

[3] Ali Borji. "Pros and cons of gan evaluation measures". In: *Computer Vision and Image Understanding* 179 (2019), pp. 41–65.

[4] Gabriel Eilertsen et al. "HDR image reconstruction from a single exposure using deep CNNs". In: *ACM transactions on graphics (TOG)* 36.6 (2017), pp. 1–15.

[5] Miguel Granados et al. "Optimal HDR reconstruction with linear digital cameras". In: *2010 IEEE computer society conference on computer vision and pattern recognition*. IEEE. 2010, pp. 215–222.

[6] Andrey Ignatov et al. "Dslr-quality photos on mobile devices with deep convolutional networks". In: *Proceedings of the IEEE International Conference on Computer Vision*. 2017, pp. 3277–3285.

[7] Phillip Isola et al. *Image-to-Image Translation with Conditional Adversarial Networks*. 2018. arXiv: `1611.07004 [cs.CV]`.

[8] Hanbyol Jang et al. "Dynamic range expansion using cumulative histogram learning for high dynamic range image generation". In: *IEEE Access* 8 (2020), pp. 38554–38567.

[9] Yu-Lun Liu et al. "Single-Image HDR Reconstruction by Learning to Reverse the Camera Pipeline". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. June 2020.

[10] John J McCann and Alessandro Rizzi. *The art and science of HDR imaging*. Vol. 26. John Wiley & Sons, 2011.

[11] Jian Sun et al. "Learning a Convolutional Neural Network for Non-Uniform Motion Blur Removal". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. June 2015.

[12] Ning Sun, Hassan Mansour, and Rabab Ward. "HDR image construction from multi-exposed stereo LDR images". In: *2010 IEEE International Conference on Image Processing*. IEEE. 2010, pp. 2973–2976.

[13] Zhou Wang and A.C. Bovik. "A universal image quality index". In: *IEEE Signal Processing Letters* 9.3 (2002), pp. 81–84. DOI: 10.1109/97.995823.

[14] Li Xu, Xin Tao, and Jiaya Jia. "Inverse Kernels for Fast Spatial Deconvolution". In: *Computer Vision – ECCV 2014*. Ed. by David Fleet et al. Cham: Springer International Publishing, 2014, pp. 33–48. ISBN: 978-3-319-10602-1.