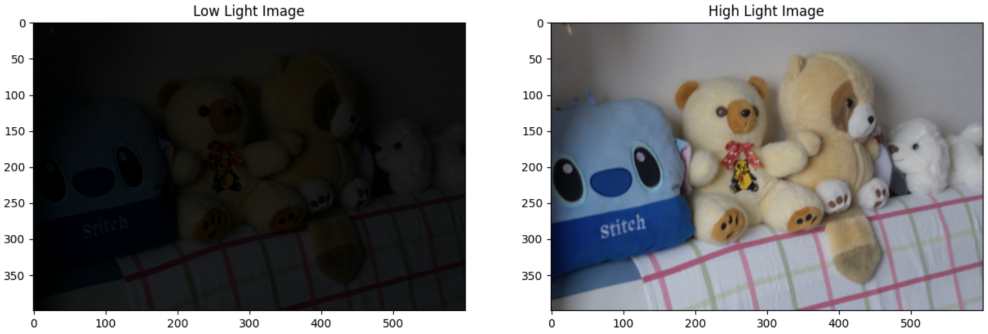
LOW-LIGHT IMAGE ENHANCEMENT

PRATHAM SINGLA
MECHANICAL ENGINEERING,FRESHER
INDIAN INSTITUTE OF TECHNOLOGY (IIT),ROORKEE

Abstract. In this project, our goal was to enhance low-light images by reducing noise and converting them into high-light images. We employed several strategies to improve the Peak Signal-to-Noise Ratio (PSNR) scores, which measure the quality of the processed images. One significant approach was refining the neural network architecture with batch normalization, which helped stabilize and accelerate the training process, leading to better results. Additionally, we implemented data preprocessing techniques that utilized histograms of the original low-light images. By analyzing these histograms, we generated new input images that more effectively captured the necessary details for enhancement. These methods collectively resulted in the most substantial improvements in PSNR scores, indicating higher-quality output images.

Dataset

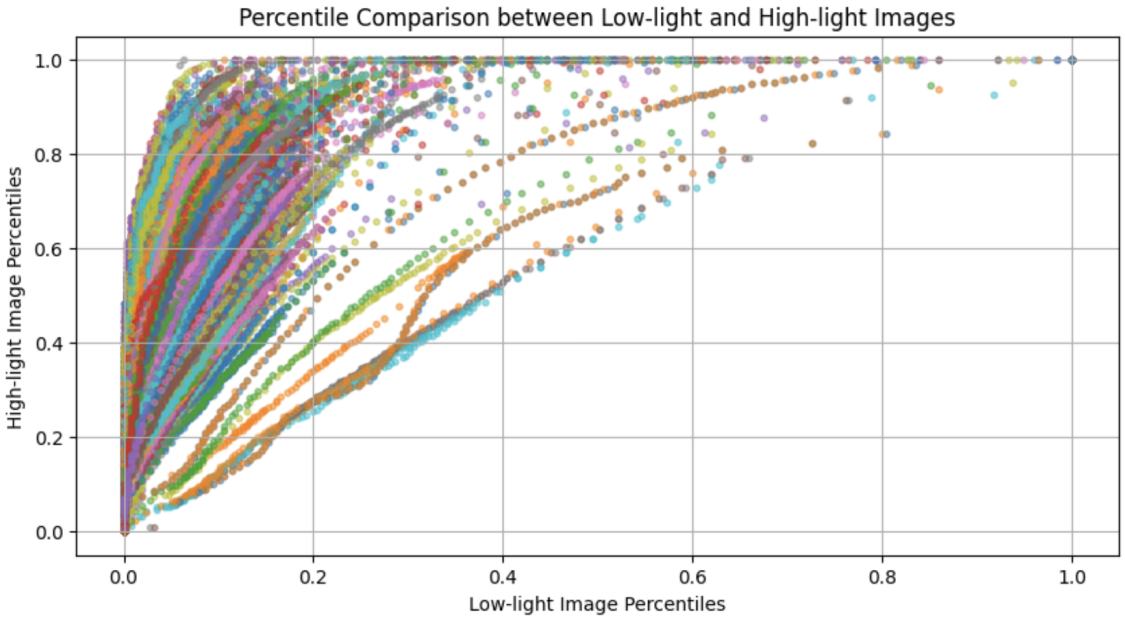
The dataset contains a total of 485 sets of paired images taken in both low-light and high-light conditions, organized into specific directories. Each set consists of two images, one captured in low-light conditions and the other in high-light conditions. An example of such a paired image is illustrated below:



1.1 Exploratory Data Analysis and Preprocessing

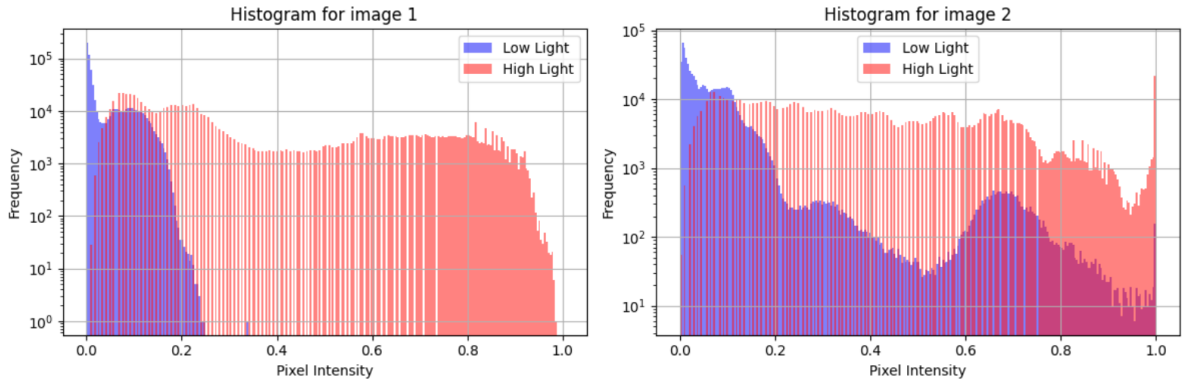
We begin by plotting a graph comparing the percentiles of low-light and high-light images for all 485 images. We then observe the correlations between these percentiles. A correlation of ≥ 0.6 is deemed significant, indicating a strong relationship between the percentiles.

Date: June 20, 2024.
VLG IITR

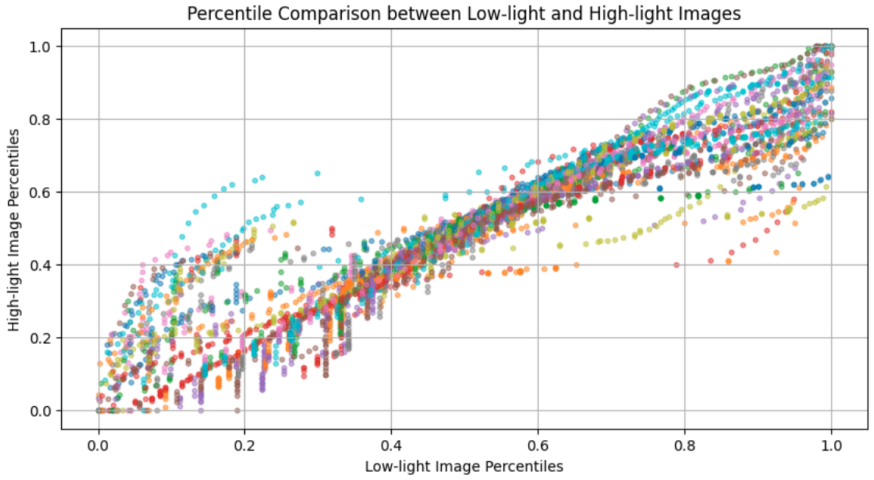



```
Number of pairs with correlation < 0.3: 0
Number of pairs with correlation between 0.3 and 0.6: 40
Number of pairs with correlation > 0.6: 445
```

Following are some more Histogram plots

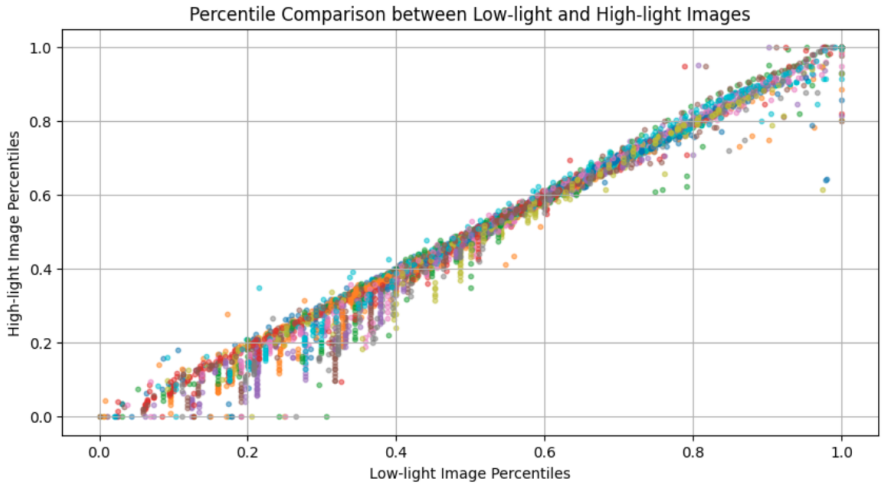


After reviewing our exploratory data analysis (EDA), we proceed to apply a quantile regression model for histogram mapping. We train an XGBoost regressor using the histogram data of low-light images as input. This model predicts the pixel values of high-light images at specific percentiles. To determine the most effective percentile, we average the correlation coefficients for the initial 50 images across different percentiles. Utilizing early stopping with a patience of 2 on n having a stride of 10, we ascertain that a percentile of 2.85 or $n = 35$ yields optimal results.




```
best: 5
```

FIGURE 1. setting $n = 5$




```
stawwp
best: 55
```

FIGURE 2. setting $n = 55$

A new predicted input looks as shown in Figure 4. The pictorial imporvement can be visibly seen as in Figure 5.

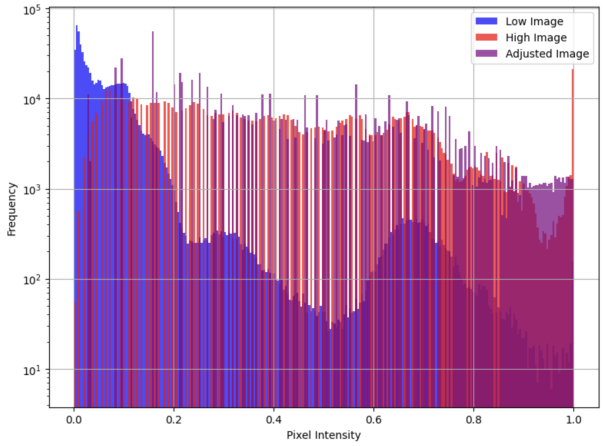


FIGURE 3. Blue is the original low light image, purple is newly learnt representation of low light image and red is the high light image histogram

FIGURE 4. Original input and newly learnt input are juxtaposed

Model and Architecture

2.1 Preprocessing Layer

The Preprocessing Layer prepares the input data for further processing by applying initial convolutional operations. It consists of a single convolutional layer with a 3x3 kernel size and padding of 1, followed by a ReLU activation function. This layer extracts features from the input image.

2.2 Special Convolutional Module

The Special Convolutional Module is structured into two stages. In the first stage, data undergo processing via two separate pathways. One path involves a 1x1 convolutional layer,

while the other employs two 3x3 convolutional layers. These pathways are combined to form the input for the second stage. This initial stage resembles the inception module, with the distinction that the results are not concatenated but rather added together. Moving to the second stage, data are processed in two ways. The first involves two 3x3 convolutional layers, while the second employs a shortcut connection, bypassing the input data directly. This shortcut connection resembles the approach used in residual learning, .

2.3 Output Layer

The Output Layer produces the final output of the model. It consists of a single convolutional layer with a 3x3 kernel size and padding of 1. The convolutional operation generates the final feature representation, which is then passed through a sigmoid activation function

2.4 Main Model

It starts by passing input through an initial layer . It then applies a series of special convolutional modules in a loop, incorporating skip connections to add the original input to the output of each module. After processing all modules, it passes the output through a final layer , applies sigmoid activation , and returns the final output.

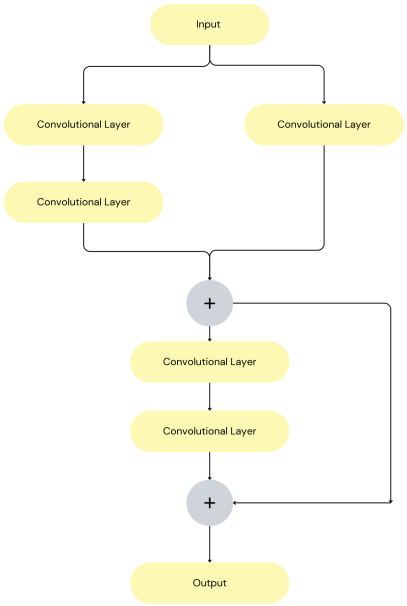


FIGURE 5. Special Convolutional Module

2.5 Losses and training

We have used a combined loss that minimise MSE as well as MAE scores. The training uses the AdamW optimizer with a learning rate scheduler (ReduceLROnPlateau) to adjust the learning rate based on validation loss, and mixed precision training is implemented using torch.cuda.amp.GradScaler for improved performance and memory efficiency. Data is split into training and validation sets, and DataLoader is used to handle batching.Early stopping is

implemented to halt training if validation loss does not improve for a set number of epochs. It is found that 5 number of epochs are sufficient to obtain a PSNR score of 24.57.

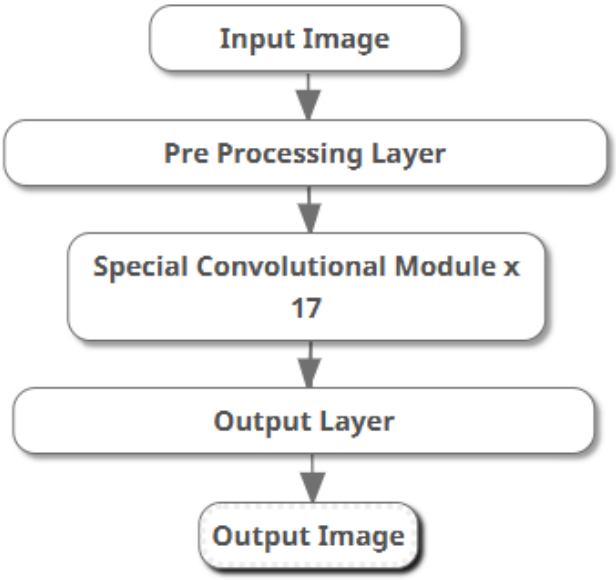


FIGURE 6. Main Architecture