PROJECT REPORT

LOW LIGHT IMAGE ENHANCEMENT

Submitted By: Hardik Agarwal

EE 3rd Year, 22115060

# INTRODUCTION:

In the broader field of computer vision, low-light image enhancement focuses on improving the visibility and quality of images captured in poor lighting conditions. Traditional methods often rely on reference images or manual adjustments, which can be impractical. The Zero-DCE (Zero-Reference Deep Curve Estimation) framework offers an innovative solution by enhancing low-light images without requiring reference images or complex pre-processing steps. This approach is particularly practical because it does not need paired low-light and normal-light images, thereby expanding the applicability of low-light enhancement techniques.

Zero-DCE is lightweight and suitable for devices with limited computational resources, such as smartphones. It operates by estimating pixel-wise and high-order tonal curves from the input image, allowing for fine control over the enhancement process. The framework iteratively estimates these tonal curves, effectively handling diverse lighting conditions within a single frame. By eliminating the need for reference images, Zero-DCE provides a practical and versatile solution for low-light image enhancement.

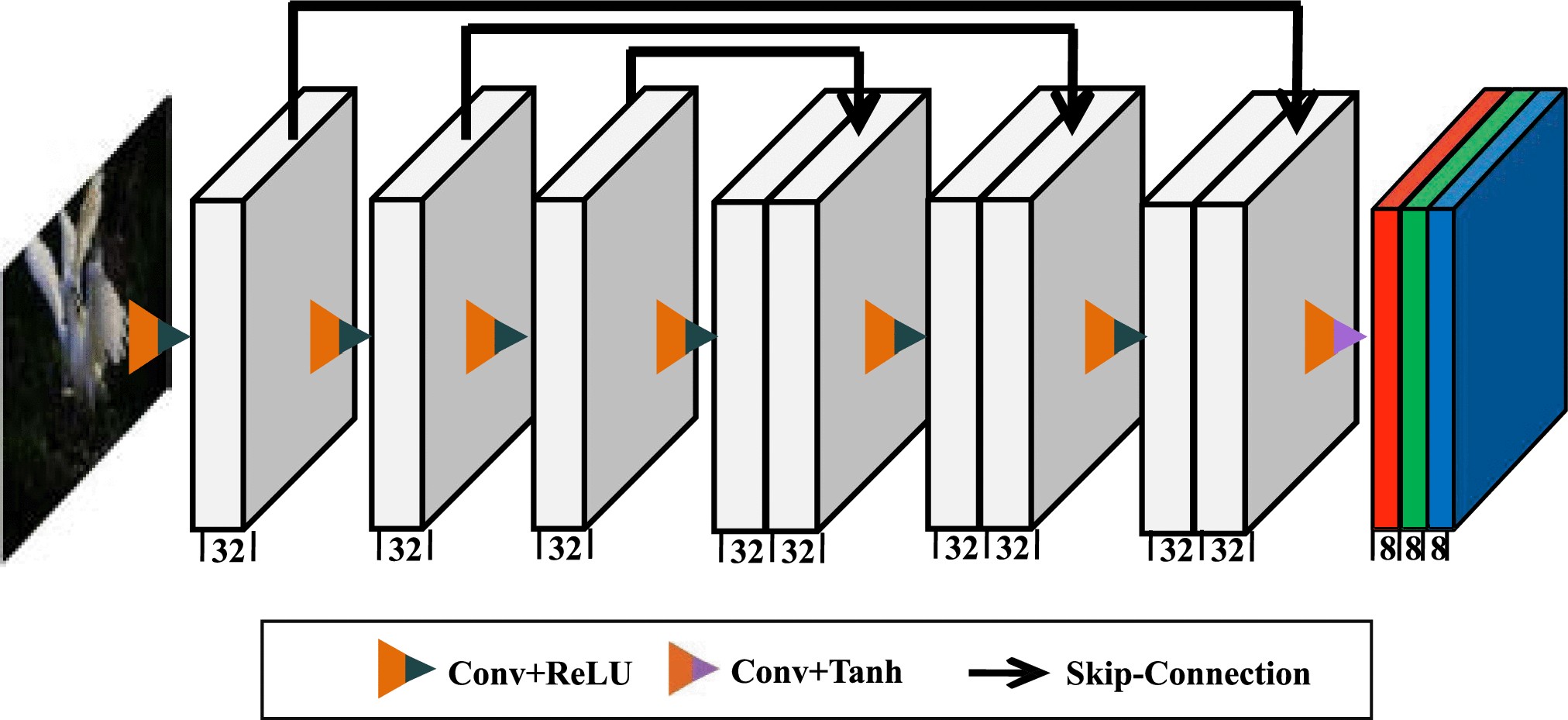
**PSNR:** 28.07 dB

# ARCHITECTURE :

The DCE network is built with multiple convolutional layers followed by concatenation layers to combine features from different stages capturing both fine-grained and high-level features, The use of 'relu' activations in the intermediate layers helps in learning complex patterns.

This is how it **works:**

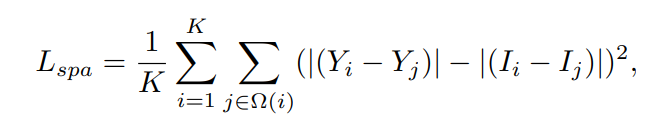
1. **Input Layer:** Takes a dark photo that you want to improve.
2. **Feature Extraction via convolutional layers:** Looks at photo on a pixel scale to understand what’s in it, like edges and textures.
3. **Curve Estimation layers:** Tonal curves are mathematical functions that describe how to adjust the brightness and contrast of each pixel.
4. **Integration Layer**: The outputs from the curve estimation modules are integrated to create a comprehensive tonal curve map for the entire image. This map guides subsequent processing steps to adjust pixel intensities accordingly.
5. **Pixel wise adjustment:** The estimated curves are applied to each pixel in the image. This means every pixel's brightness and contrast are adjusted based on its specific needs.
6. **Output Layer:** It is made up of enhanced brightness and improved contrast preserving the integrity of the original content.



The DCE-Net, a deep convolutional neural network, is designed to enhance low-light images by learning to map them to their optimal curve parameter maps. The architecture consists of seven convolutional layers, utilizing symmetrical skip concatenation. Each of the first six convolutional layers includes 32 filters with a 3x3 kernel size and a stride of 1, followed by ReLU activation functions. The final convolutional layer contains a variable number of filters, specifically iteration x 3. For instance, setting the iteration count to 8 results in 24 curve parameter maps, as each iteration generates three curve parameter maps corresponding to the RGB channels.

DCE-Net is unique in that it does not require input/output image pairs during training. Instead, it utilizes a set of meticulously designed non-reference loss functions that implicitly assess enhancement quality, guiding the network's training. The proposed zero-reference loss functions are differentiable, allowing for the evaluation of the enhanced image's quality. These functions inherently measure the enhancement effectiveness and drive the network's learning process.

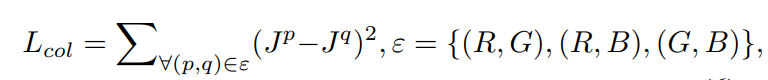
**Spatial Consistency Loss:** By ensuring consistency of the spatial feature maps produced over consecutive training epochs, maintaining per-class running-average heatmaps for each training image. We show that this spatial consistency loss further improves the enhanced image through preserving the difference of neighboring regions between the input image and its enhanced version.



We denote Y and I as the average intensity value of the local region in the enhanced version and input image, respectively

**Color Constancy Loss:** This function aims to ensure that the enhanced image maintains a consistent color balance. This function calculates the loss based on the mean color values of the red, green, and blue channels of the image, penalizing deviations from an ideal color balance. Then the function calculates the squared differences between the mean value of each pair of channels. Function returns square root of sum of differences of each pair of means.

The use of square root ensures that the loss function is differentiable and provides a gradient that can be used for optimization.



* where J p denotes the average intensity value of p channel in the enhanced image, (p,q) represents a pair of channels.

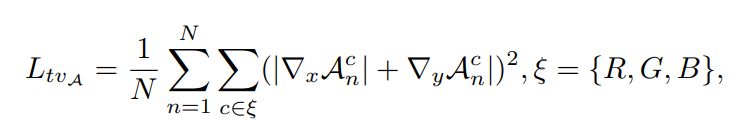
**Exposure Loss:** This function is designed to ensure that the enhanced image has an appropriate overall brightness level. This function calculates the loss based on how the mean brightness of overlapping patches of the image deviate from specific target value. First we reduce number of channels to 1 by taking average of R, G, B channels which represents average brightness value of that pixel. Then function apply Average pooling over non overlapping patches which computes

A mathematical equation with numbers and symbols

Description automatically generated with medium confidence

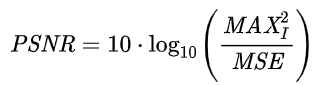
* We set E to 0.6 in our experiments although we do not find much performance difference by setting E within [0.4, 0.7].
* Y is the average intensity value of a local region in the enhanced image.

**Illumination Smoothness Loss:** This function is essential in improving images using deep learning. It focuses on making sure that changes in brightness across the image are smooth and gradual, rather than sudden and jarring. It looks at batches of images and calculates how much each pixel's brightness differs from its neighbors vertically and horizontally. By averaging these differences across all pixels and images in the batch, the function helps create enhanced images that look more natural and pleasing to the eye. This is particularly useful in tasks where maintaining a consistent look throughout the image is important for overall image quality.



* where N is the number of iterations, ∇x and ∇y represent the horizontal and vertical gradient operations, respectively.

**PSNR Value**: PSNR (Peak Signal-to-Noise Ratio) is a metric used to measure the quality of an image or a signal reconstruction compared to its original version**.**

****

**DATASET PREPARATION:**

The dataset on which the training is done is the LoL dataset which contains 485 images for training and 15 for testing. Each image pair in LoL consists of a low-light input image and its corresponding well-exposed reference image. We have split the images available for training into train and validation sets with the train dataset containing 390 images and the validation dataset containing 95 images. The Adam optimizer is used for the optimization of the training process and the learning rate being 2 × 10-4 . Model is train in batch of size 16. The model was trained for 50 epochs.

**RESULTS:**

The model is evaluated mainly on the metrics PSNR (peak signal-to-noise ratio) and we have also calculated MSE (mean squared error).

|  |  |  |  |
| --- | --- | --- | --- |
| **Metrics** | **Train Dataset** | **Test Dataset** | **Validation Dataset** |
| **MSE** | **102.20** | **101.83** | **106.28** |
| **PSNR** | **28.06** | **28.07** | **27.88** |

Example of image enhancement

A close-up of a pool

Description automatically generated





The implementation of my project can be found on my GitHub repository:

[**https://github.com/krishagrwl/Image\_Denoising\_Zero\_DCE\_NET/tree/main**](https://github.com/krishagrwl/Image_Denoising_Zero_DCE_NET/tree/main)

* **zero\_dce\_model.py**: This python file contains architecture of model
* **losses.py:** This python file contains all the loss function which is to be used in model
* **train.ipynb:** This file runs the training script and saves a model.h5 file in current directory
* **main.ipynb**: This file runs the inference on image which are present in test/low/ of current directory and save results on /test/predicted.

**FINDINGS:**

* Zero-Reference Deep Curve Estimation (Zero-DCE) enhances low-light images by generating high-order tonal curves specifically for each image.
* This method basically improves color constancy,control average brightness of the image and for image smoothness (which helps in keeping image looks natural), it reduce pixel intensity differences with all its adjacent pixels
* Effective for various low-light scenarios without the need for paired training data, making it versatile and efficient

**METHODS TO IMPROVE:**

* Incorporate a diverse range of low-light conditions in the training dataset. This diversity will help the model generalize better to various scenarios.
* further refine the visual quality of the enhanced images, applying additional contrast

enhancement techniques after the initial enhancement.

**RESOURCES:**

* **Zero DCE Paper:** [**https://openaccess.thecvf.com/content\_CVPR\_2020/papers/Guo\_Zero-Reference\_Deep\_Curve\_Estimation\_for\_Low-Light\_Image\_Enhancement\_CVPR\_2020\_paper.pdf**](https://openaccess.thecvf.com/content_CVPR_2020/papers/Guo_Zero-Reference_Deep_Curve_Estimation_for_Low-Light_Image_Enhancement_CVPR_2020_paper.pdf)