-IMAGE DENOISING-

LOW LIGHT IMAGE ENHANCEMENT

Introduction-

This project tackles the challenge of enhancing low-light images. It leverages a deep learning technique called Zero-Reference Deep Curve Estimation (Zero-DCE). This project focuses on implementing the Zero-DCE model and evaluating its effectiveness in enhancing low-light images.

Core idea of Zero-DCE-

The core idea of Zero-DCE is to train a deep neural network, specifically a lightweight network called DCE-Net, to estimate an image- specific curve. This curve essentially dictates how the brightness of each pixel in the low-light image should be adjusted. The network considers factors like pixel value range, monotonicity (brightness should always increase or decrease smoothly) and differentiability (the curve should have no abrupt jumps) to ensure the adjustments are natural-looking.

Advantages-

No need for reference images: Make the enhancement process more convenient and applicable to a wider range of scenarios where capturing a well-lit version of the scene might not be possible.

Image-specific adjustments: The curve is tailored to the unique characteristics of each low-light image, leading to potentially more natural – looking enhacements compared to generic techniques.

About Project-

This project delves into the implementation of the Zero-DCE model using Keras, a popular deep learning library for building and training neural networks. The project is structured as a Kaggle notebook, allowing for interactive exploration and visualization of the process.

The notebook can be broken down into several key sections, each playing a crucial role in achieving low-light image enhancemnet through Zero- DCE:

Importing Libraries-

First we need to import all necessary libraraies like os, glob, numpy, matplotlib, tensorflow, keras etc.

TensorFlow/Keras: This is the heart of this project. TensorFlow provides the computational backend for deep learning, while keras allows to define the architecture of the Zero-DCE model (DCE-Net) using layers and funtions.

Numpy: This fundamental library offers powerful tools for numerical computations.

Matplotlib: It allows to create various plots and charts to understand the model's behavior and analyze results.

Loading Data and Dataset-

About Data- The dataset is composed of 500 low-light and normal- light image pairs and is divided into 485 training pairs and 15 testing pairs. The low light images contain noise produced during the photo capture process. Most of the images are indoor scenes.

Preprocessing of Data- This step ensures the model receives data in a format it can understand and efficiently process. The common preprocessing techniques are resizing images, normalizing pixel values, data type conversion etc.

Dataset Split: Training Set is the largest portion used to train the Zero-DCE model. The model learns form patterns and relationships within these images to perform low-light enhancement.

Validation Set: This set helps monitor the model's performance during training.

Test Set: After training and validation, this final set is used to evaluate the model's performance on completely new low-light images.

I used 64 as the Batch Size.

```
Train Dataset: <_BatchDataset element_spec=TensorSpec(shape=(64, 256, 256, 3), dtype=tf.float32, name=None)>
Validation Dataset: <_BatchDataset element_spec=TensorSpec(shape=(64, 256, 256, 3), dtype=tf.float32, name=None)>
```

Building ZeroDCE Network-

The core of project lies in constructing the Zero- DCE model using Convolutional layers. Convolutional layers are the building blocks of Convolutional Neural Networks (CNNs) like Zero-DCE. They learn filters (weights) that are applied to small regions of the input images of extract features. As the network progresses through deeper layers, the filters becomed more intricate, allowing them to capture higher-level features that combine and build upon the simpler features learned earlier. These highter- level features might represent specific shapes, objects or illumination patterns relevent to low- light image enhancement.

Convolutional layers themeselves perform linear operations. However, real- world problems often involve non- linear relationships. Activation functions are introduced after each convolutional layer to add non-lineraity to the network.

I used common activation functions in Zero-DCE like ReLU (Rectified Linear Unit) and Tanh (Hyperbolic tangent)/ These functions introduce non- linearity while maintaining computational efficiency.

Loss Functions-

Exposure loss- Exposure loss plays a vital role in the Zero-DCE model by guiding the network towards producing enhanced images with appropriate brightness levels. Here's a closer look at how it works:

Mean Squared Difference: pooled_mean (This represents the average intensity of the enhanced image after applying average pooling.) and **target_mean (**This is a predefined value that signifies the desired average intensity for a well-exposed image.)

Average Pooling for Overall Mean: Average pooling is a technique used to reduce the dimensionality of the feature maps generated by the convolutional layers.

Illumination smoothness loss- The illumination smoothness loss plays a crucial role in the Zero-DCE model by ensuring the adjustments made to image illumination occur smoothly across neighboring pixels. This helps to prevent artifacts and unnatural lighting effects in the enhanced image. In simpler terms, the illumination smoothness loss acts as a "smoothness filter" for the Zero-DCE model. It discourages the model from making drastic changes in brightness from one pixel to the next, promoting a more natural and visually pleasing enhancement.

Color constancy loss- The color constancy loss acts as a color balance check for the Zero-DCE model. It helps to prevent the model from introducing unrealistic color casts or imbalances in the enhanced image, leading to a more natural and faithful representation of the original scene's colors.

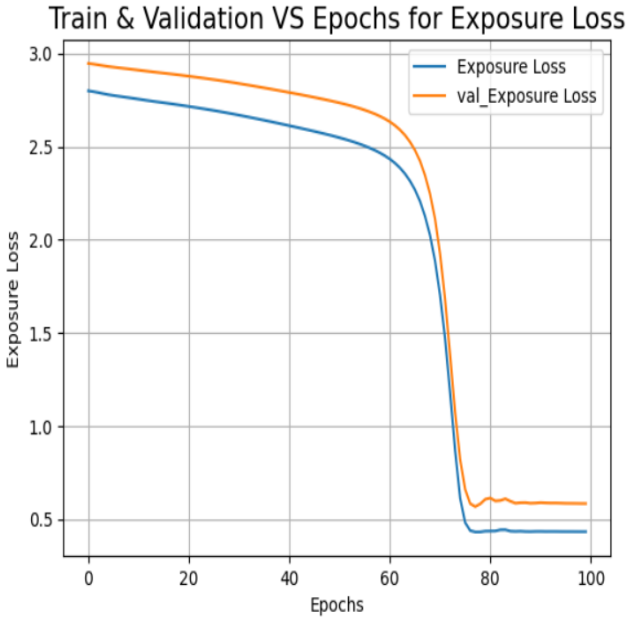
Spatial consistency loss- The spatial consistency loss acts as a "detail preservation" check for the Zero-DCE model. It ensures that while enhancing the image's brightness, the model doesn't alter the fundamental relationships between neighboring pixels, which are crucial for conveying scene details and visual coherence.

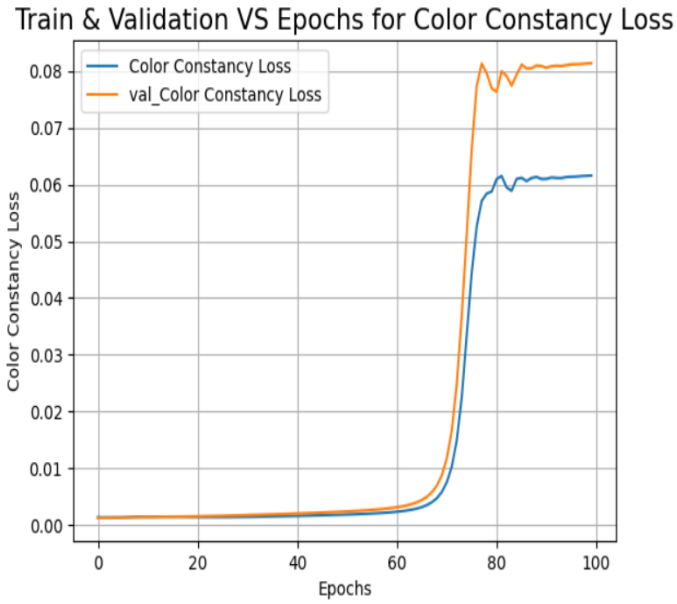
Training-

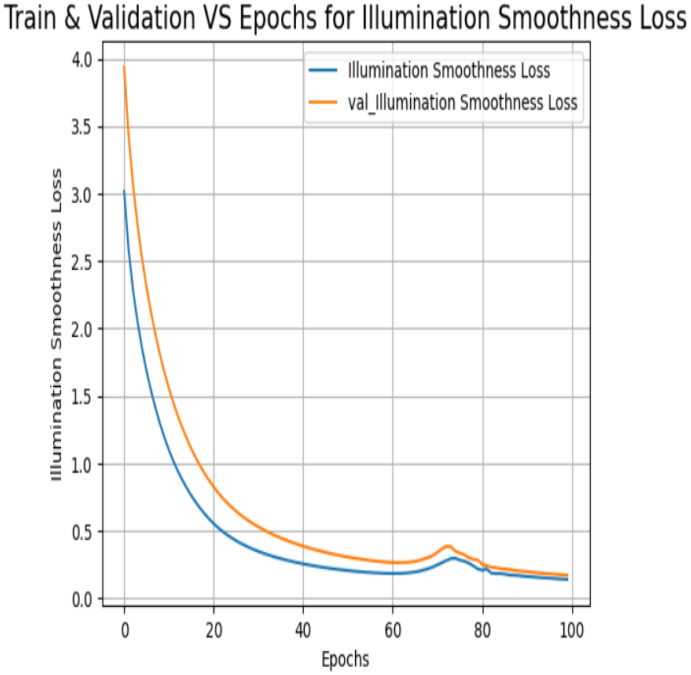
In this section includes defining the number of training epochs (iterations over the entire dataset). Setting the batch size which determine how many images are processed at once during training. Compile the model and fit the train and validation dataset in model.

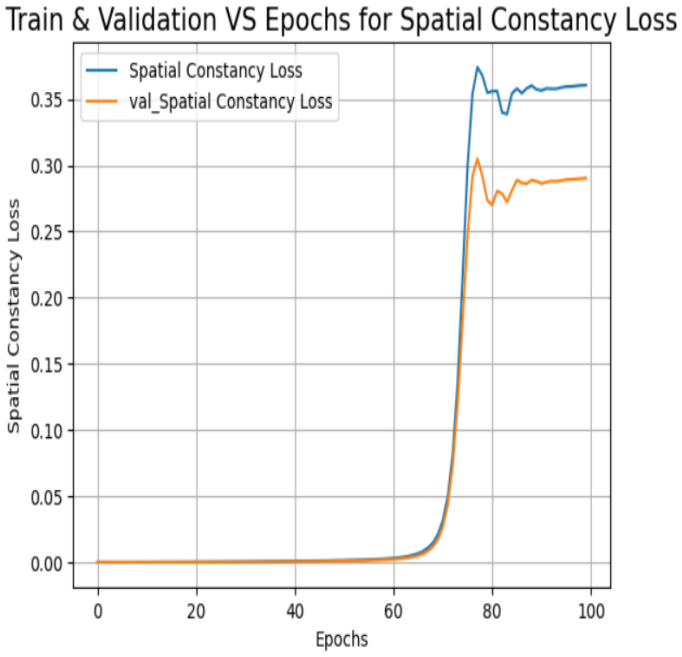
```
DCE_model = Zero_DCE()
DCE_model.compile(learning_rate=1e-4)
history = DCE_model.fit(train_dataset, validation_data=val_dataset, epochs=100)
```

Visualization of Loss Function -









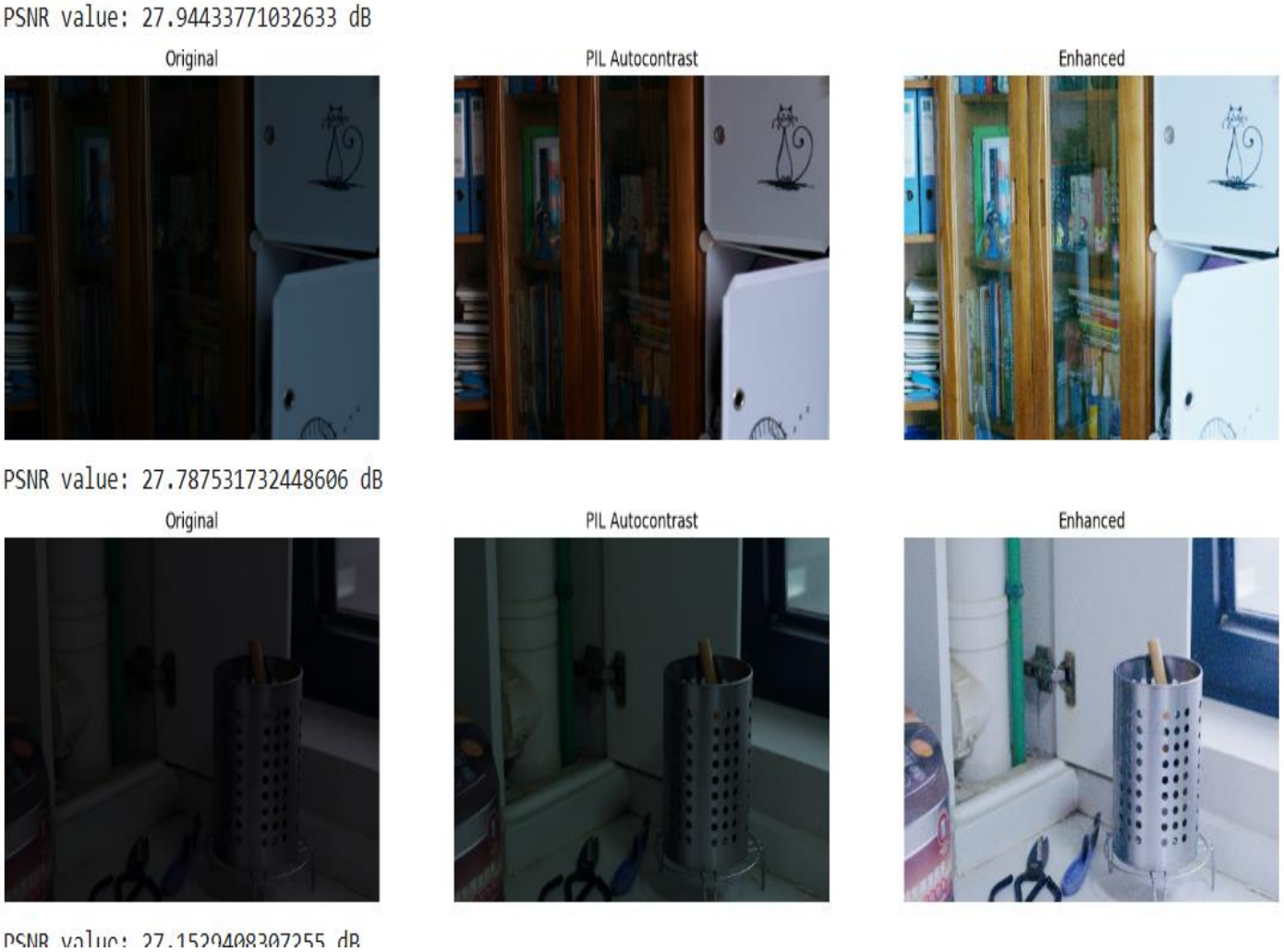
Evaluation (PSNR Score)-

PSNR (Peak Signal- to- Noise Ration) as metric to quantify the improvement in image quality. PSNR measures the ration between the maximum possible signal power and the power of noise introduced during the enhancements process. Higher PSNR values generally indicate better image quality.

I tested the model on the test data (15 images) and displaying the original low-light images, corresponding predicted (enhanced) images and high light images. And I calculated the PSNR value for all this test images.

I got 28.24 dB average PSNR value for my this model.

The final result sample:-



Summary-

Unlike traditional methods, Zero-DCE estimates enhancement curves to improve image illumination without requiring reference images. This iterative process leads to significant improvements in image quality, as demonstrated by an average PSNR of 28.24 dB in our evaluation.

Reference-

This project is based on the research paper.

“ Zero-Reference Deep Curve Estimation for Low-Light Image Enhancement”

You can access this paper form here.

Rajshree Prajapati

22124032

BSBE