Low Light Image Enhancement Using CNN: Project Report

Introduction

Project Overview

This project focuses on enhancing images taken in low-light conditions using a Convolutional Neural Network (CNN). Images captured in poor lighting often suffer from noise and lack of detail, which can impair various computer vision tasks. The aim is to develop a model that can effectively denoise and enhance these images, improving their clarity and overall quality.

Project Goals and Scope

Scope: The project aims to create and implement a CNN-based model for enhancing low-light images using a provided dataset.

Objectives:

- Design a CNN architecture specifically for image denoising.
- Train the model on a dataset of paired low-light and high-light images.
- Evaluate the model's performance using metrics such as Peak Signal-to-Noise Ratio (PSNR).
- Deploy the model and demonstrate its ability to enhance low-light images.

System Architecture and Data Flow

System Components

Dataset

- **Source:** The dataset is provided by the VLG Slack platform.
- **Contents:** It contains paired images of low and high light conditions.

Model Architecture

The model uses a convolutional neural network designed to reduce noise and enhance low-light images. The key layers include:

- **Input Layer:** Accepts RGB images at any resolution.
- **Convolutional Layers:** Multiple layers with 32 filters each, using 3x3 kernels and ReLU activation for feature extraction and noise reduction.
- **Skip Connections:** These connections help retain and utilize information from earlier layers.
- **Output Layer:** A convolutional layer with sigmoid activation to produce the final enhanced image.

```
__________________________________________________________________________________________________
 Layer (type)                Output Shape                 Param #   Connected to
==================================================================================================
 img (InputLayer)            [(None, None, None, 3)]      0         []

 conv2d (Conv2D)             (None, None, None, 32)       896       ['img[0][0]']

 conv2d_1 (Conv2D)           (None, None, None, 32)       9248      ['conv2d[0][0]']

 conv2d_2 (Conv2D)           (None, None, None, 32)       9248      ['conv2d_1[0][0]']

 conv2d_3 (Conv2D)           (None, None, None, 32)       9248      ['conv2d_2[0][0]']

 add (Add)                   (None, None, None, 32)       0         ['conv2d_2[0][0]',
                                                                     'conv2d_3[0][0]']

 conv2d_4 (Conv2D)           (None, None, None, 32)       9248      ['add[0][0]']

 add_1 (Add)                 (None, None, None, 32)       0         ['conv2d_1[0][0]',
                                                                     'conv2d_4[0][0]']

 conv2d_5 (Conv2D)           (None, None, None, 32)       9248      ['add_1[0][0]']

 add_2 (Add)                 (None, None, None, 32)       0         ['conv2d[0][0]',
                                                                     'conv2d_5[0][0]']

 conv2d_6 (Conv2D)           (None, None, None, 1)        289       ['add_2[0][0]']

==================================================================================================
Total params: 47425 (185.25 KB)
Trainable params: 47425 (185.25 KB)
Non-trainable params: 0 (0.00 Byte)
__________________________________________________________________________________________________
```

Fig:- Model Summary

Data Flow

1. **Loading Data:** Images are loaded from the dataset and pre-processed.

2. **Training the Model:** The CNN model is trained on pre-processed images.
3. **Model Testing:** The trained model is tested on a separate set of low-light images.
4. **Image Enhancement:** The model processes low-light images to create enhanced versions.

Techniques and Methodologies

CNN Architecture Details

- **Layers:** The model consists of several convolutional layers with ReLU activation, interspersed with skip connections for better gradient flow and feature reuse.
- **Activation Functions:** ReLU is used in hidden layers and sigmoid in the output layer to ensure pixel values are in the range [0,1].
- **Loss Function:** Mean Squared Error (MSE) measures the difference between predicted and ground truth images.
- **Metrics:** PSNR is used to evaluate the quality of denoised images.

Data Pre-Processing

- **Normalization:** Image pixel values are normalized to the range [0,1].
- **Shape Adjustment:** Input images are adjusted to match the model's expected input shape.
- **Channel Handling:** Grayscale images are converted to RGB by replicating one channel.

Training and Evaluation

- **Training Process:** The model is trained for 22 epochs with batch sizes determined by the dataset size, using Google Colab.
- **Evaluation Metrics:** PSNR is calculated to assess the denoising performance. Higher PSNR indicates better image quality.

Implementation

Data Collection and Storage

Sources: The dataset is provided on Slack and includes paired low-light and high-light images.

Storage Method: Images are uploaded to Google Drive and stored in directories within the Google Colab environment.

Schema Design: The dataset is organized into separate folders for low and high light images.

Data Processing

Loading and Preprocessing: Images are loaded using the imageio library and pre-processed to meet the model's input requirements.

Normalization: Pixel values are normalized to the range [0,1].

Model Development

Model Selection: The CNN architecture is chosen for its effectiveness in image processing tasks.

Inspiration: The model's architecture is inspired by a Kaggle notebook that utilized parallel networks and complex CNN architectures.

Model Architecture: The final architecture was determined through experimentation with different CNN layers and skip connections.

Training: The model is trained using the Adam optimizer and MSE loss function, with parameters like epochs and steps per epoch tuned for optimal performance.

Deployment and Monitoring

Tools and Platforms: The model is trained and deployed on Google Colab, with the trained model saved to Google Drive.

Deployment Process: The model is saved in HDF5 format and can be loaded for generating new images.

Monitoring: Model performance is monitored using PSNR values during testing and inference.

Conclusion

Achievements: Successfully developed and trained a CNN model for low-light image enhancement, demonstrating significant improvement in image quality as measured by PSNR.

Challenges: Managing different architectures and ensuring stable training were significant challenges, addressed through extensive experimentation and model tuning.

Results: The model was tested on a portion of dataset provided on VLG slack, yielding significant improvements in image quality.

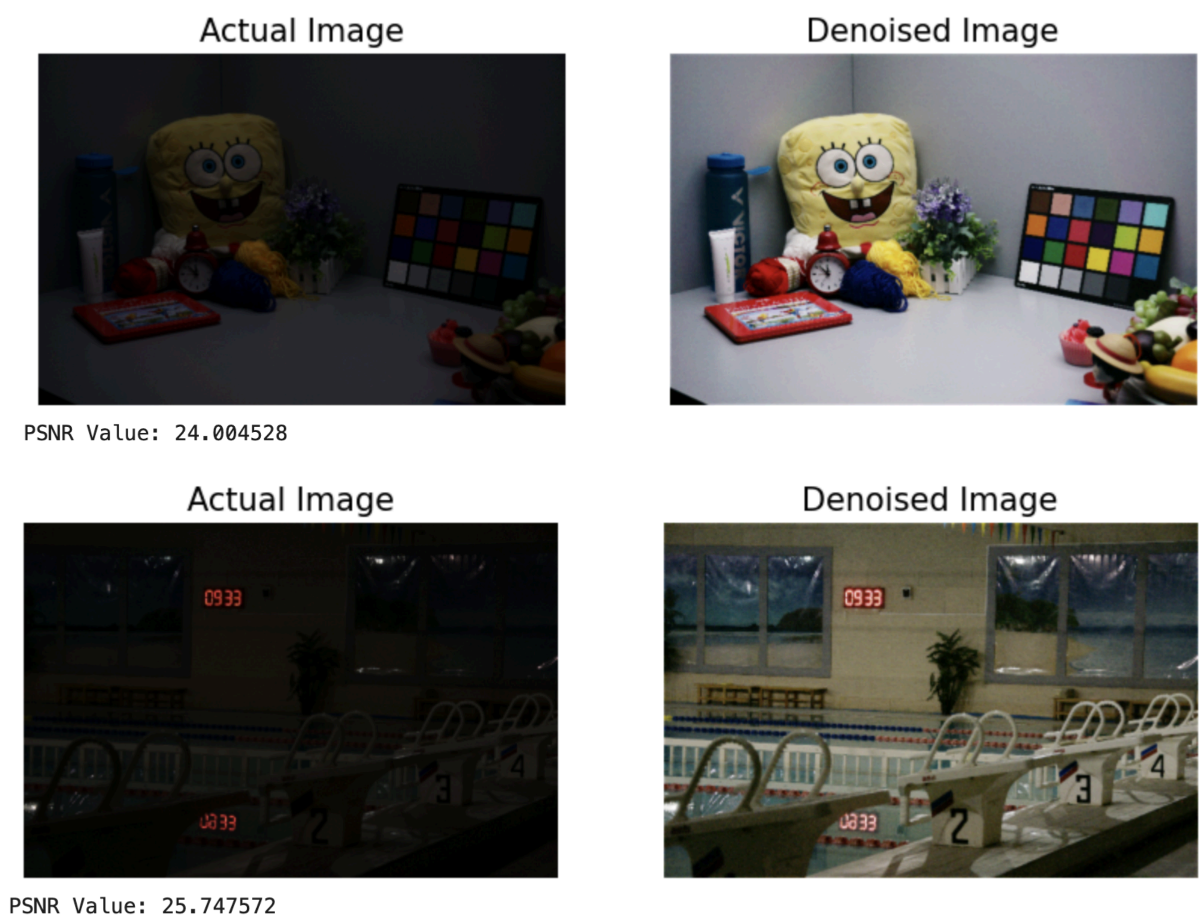

Fig:- Actual Low light image Vs Denoised Image

The model achieved an average PSNR value of 27.06695.

For reference, the model's architecture and further details can be found at [Kaggle Notebook].