Denoising Project report

1. Introduction

1.1 Project overview

The goal of this project is to improve low-light images using a Convolutional Neural Network (CNN). Poor lighting conditions often result in images with poor visibility, noise, and reduced detail, which can hinder various computer vision tasks. This project aims to develop a model that can effectively denoise and enhance such images, improving their quality and usability.

1.2 Scope and Objectives

- **Scope**: This project focuses on the design and implementation of a CNN-based model for low-light image enhancement using the provided dataset.
- **Goals**:
  - Develop a CNN architecture adapted for image denoising.
  - Train the model using a dataset of paired low and high light images.
  - Evaluate model performance using appropriate metrics such as Peak Signal-to-Noise Ratio (PSNR).
  - Deploy the model and demonstrate its enhancement capabilities on invisible low-light images.

2. Architecture

2.1 System architecture

The system architecture includes the following components:

2.1.1 Data File

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

2.1.2 Model architecture

The model is a convolutional neural network designed to remove noise and enhance low-light images. The key layers and their roles are as follows:

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

2.2 Data Flow

A data flow diagram shows the movement of data through the system:

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 images in low light and creates enhanced versions.

3. Implemented techniques

3.1 CNN architecture

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

3.2 Data pre-processing

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

3.3 Training and Evaluation of Models

- **Training process**: The model is trained for 22 epochs with a batch size determined by the dataset size. Each epoch processes a subset of the training data. This 22 epochs were decided as per experimentations on google colab.
- **Evaluation Metrics**: PSNR is calculated to evaluate the denoising performance of the model. Higher PSNR means better image quality.

4. Overall piping

4.1 Data collection

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

4.2 Data Storage

- **Storage Method**: Images were uploaded on drive and stored in directories on the local file system of the Google Colab environment.
- **Schema Design**: The dataset is organized into separate folders for low and high light images.

4.3 Data processing

- **Loading and preprocessing**: Images are loaded using the `imageio` library and preprocessed to meet the model's input requirements.
- **Normalization**: Image pixel values are reduced to the range [0,1].

4.4 Model development

- **Model Selection**: The CNN architecture is chosen for its efficiency in image processing tasks.
- **Inspiration**: Model's architecture was inspired from Kaggle notebook which used parallel networks and complex architectures with CNN, Link: https://www.kaggle.com/code/basu369victor/low-light-image-enhancement-with-cnn
- **Model Architecture:** After training on different architecture, using different CNN Layers, utilising different skip connections, final architecture was decided. Overall Architecture is given on last page of this report.
- **Training**: The model is trained on the provided dataset using the Adam optimizer and the MSE loss function. Training parameters such as epochs and steps per epoch are tuned for optimal performance.

4.5 Deployment and Loading

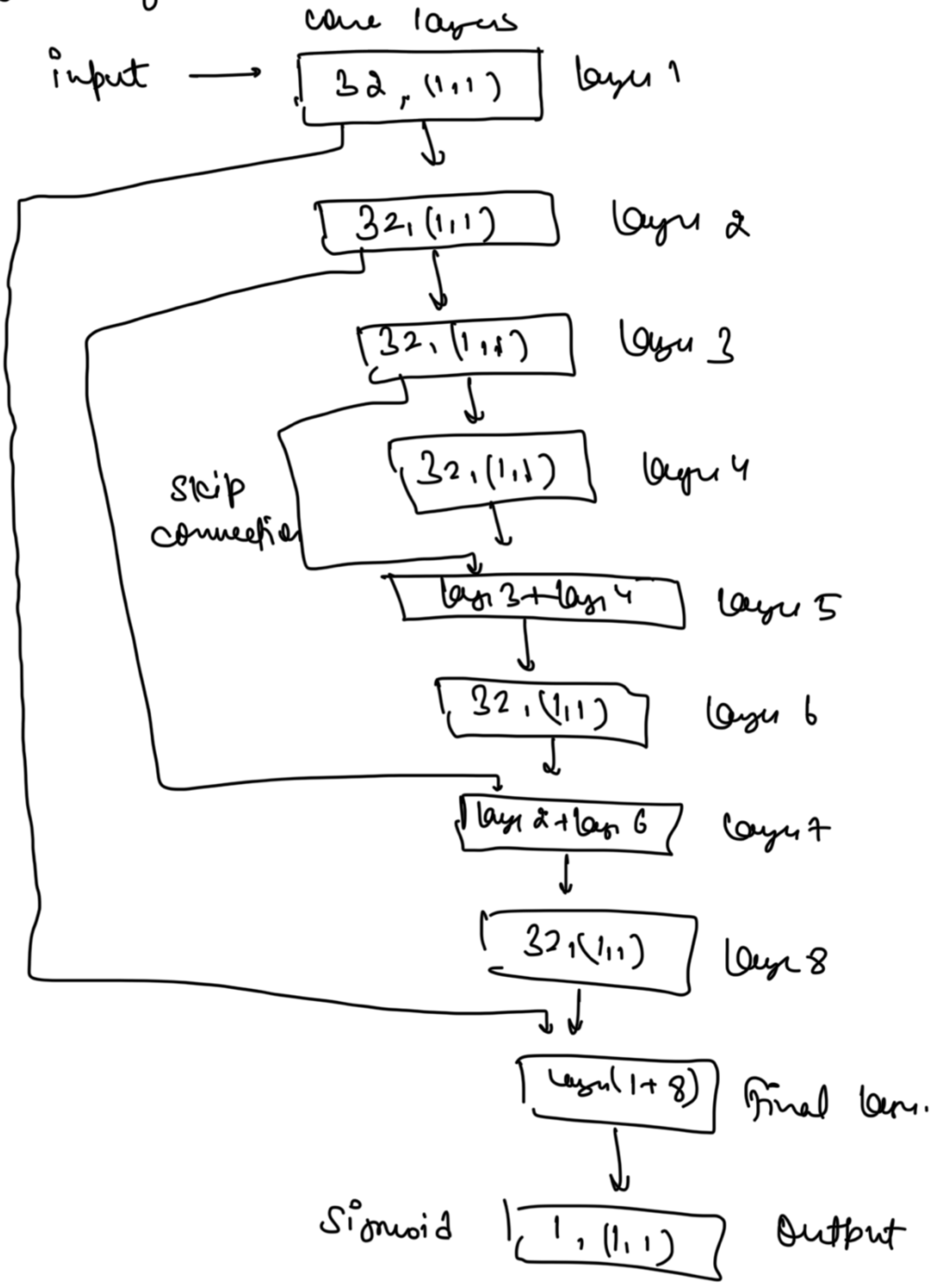
- **Tools and Platforms**: The model is trained and deployed using Google Colab. The trained model is saved to Google Drive and then downloaded.
- **Deployment Process**: The model is saved in HDF5 format and can be loaded to derive new images.
- **Monitoring**: Model performance is monitored using PSNR values during testing and inference.

5. Conclusion

5.1 Summary

- **Achievements**: Successfully developed and trained a CNN model for low-light image enhancement. The model shows a significant improvement in image quality as measured by PSNR.
- **Challenges**: Handling different architectures and ensuring stable training were significant challenges that were solved by multiple experimenting and model tuning.
- **Results**: After training model, it was tested on self-captured photos on phone camera and it provided significant results.

Denoising architecture.


conv layers
input
32, (1,1)
layer 1
32, (1,1)
layer 2
32, (1,1)
layer 3
32, (1,1)
layer 4
skip connection
layer 3 + layer 4
layer 5
32, (1,1)
layer 6
layer 2 + layer 6
layer 7
32, (1,1)
layer 8
layer (1+8)
Final layer.
Sigmoid
1, (1,1)
Output