DENOISING IMAGE USING U-NET

\

I implemented U-Net model for low light image enhancement. This U-Net model has many features such as-

1. U-Net Architecture

- **Symmetric Encoder-Decoder Structure**: The U-Net consists of a contracting path to capture context and a symmetric expanding path for precise localization.
- **Skip Connections**: Direct connections between corresponding layers in the encoder and decoder paths facilitate the recovery of spatial information lost during down sampling.

2. Residual Blocks

- **Enhanced Feature Learning**: Integrating residual blocks allows the model to learn identity mappings, which help in training deeper networks more effectively.
- **Gradient Flow Improvement**: Residual connections mitigate the vanishing gradient problem, promoting better gradient flow during backpropagation.

3. Image Denoising

- **Noise Reduction**: The model is specifically trained to remove noise from images, enhancing their visual quality.
- **Preservation of Details**: The architecture ensures that fine details are preserved while denoising, maintaining the integrity of the original image.

4. Configurable Hyperparameters

- **Adjustable Filters**: The number of filters in each convolutional layer can be customized, allowing flexibility in model complexity.
- **Dropout Regularization**: Dropout layers can be included to prevent overfitting, enhancing the model's generalizability.

5. Training Configuration

- **Loss Function**: Utilizes Mean Absolute Error (MAE) to optimize the model, which is effective for regression tasks like image restoration.
- **Optimizer**: Employs the Adam optimizer with configurable learning rate and gradient clipping to ensure stable and efficient training.

6. Evaluation Metrics

- **PSNR (Peak Signal-to-Noise Ratio)**: Used to quantify the performance of the denoised images, providing a measure of reconstruction quality.

7. TensorFlow and Keras Integration

- **Efficient Implementation**: The model leverages TensorFlow and Keras for efficient deep learning operations, making it scalable and adaptable to different hardware configurations.
- **Ease of Use**: Keras' high-level API simplifies the process of model building, training, and evaluation.

The Code and the structure has the following-

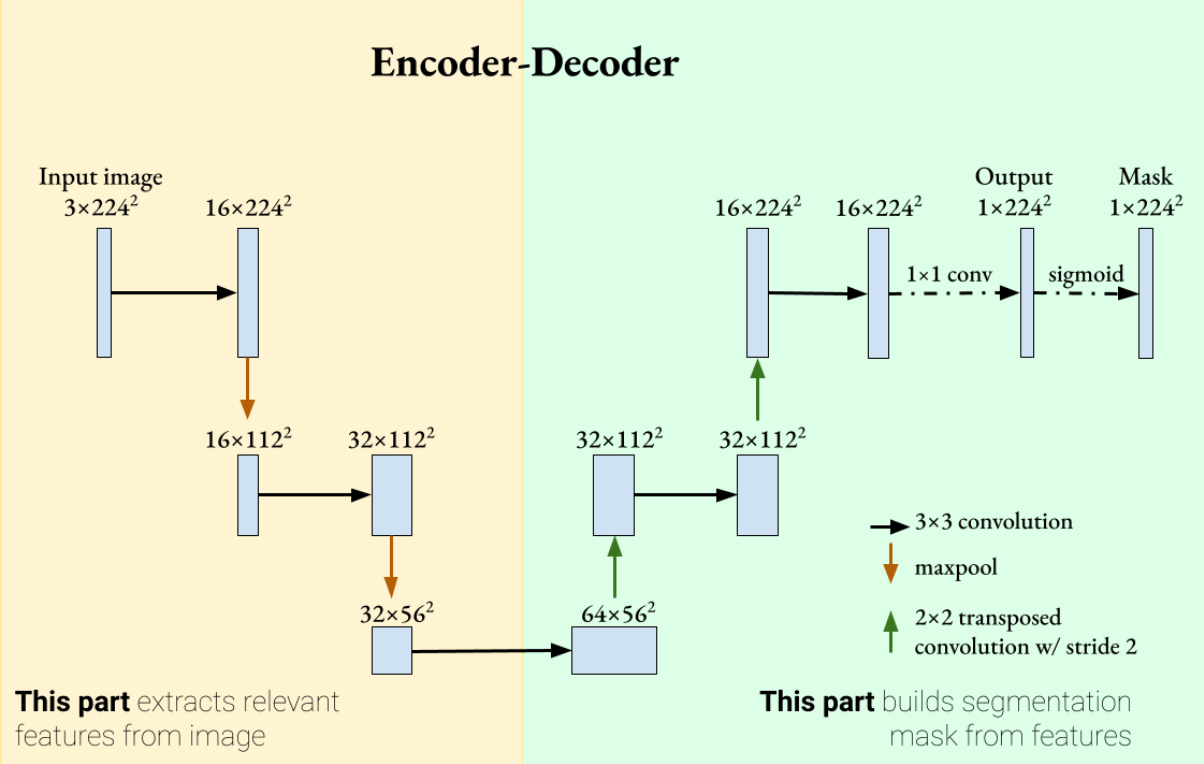
1) Loading and Preprocessing Image
2) Applying Unet Model
3) Defining Model
4) Training Model
5) Calculating PSNR value
6) Plotting Graph Results
7) Normalizing Pixel Values

U-Net Model Architecture

Below is a visual representation of the U-Net model architecture, highlighting its encoder-decoder structure and skip connections:

Description of U-Net Architecture

1. **Encoder (Contracting Path)**:
    - Series of convolutional layers with increasing filters.
    - Max pooling layers for downsampling.
    - Skip connections to corresponding layers in the decoder.
2. **Decoder (Expanding Path)**:
    - Series of upsampling layers (using transposed convolutions).
    - Convolutional layers for feature learning.
    - Skip connections to incorporate high-resolution features from the encoder.
3. **Residual Blocks**:
    - Implemented within the convolutional layers.
    - Enhance feature extraction and gradient flow during training.



Using The UNET MODEL

The unet model was used and the data was trained on images imported on drive and loss was calculated instead of using a mean squared error a Mean Absolute error was calculated

Which would give a better result and the PSNR value was calculated using this

The formula for PSNR is:

$$\text{PSNR} = 20 \cdot \log_{10} \left(\frac{\text{MAX}_I}{\sqrt{\text{MSE}}} \right)$$

where:

- MAX_I is the maximum possible pixel value of the image. For an 8-bit image, this value is 255.
- MSE stands for Mean Squared Error between the original image I and the denoised image K.

The model was trained for 50 epochs with a batch size of 50, resulting in a PSNR value of 19.3976, calculated using the standard formula. To enhance this PSNR value, we utilized PyTorch, Python 3.x, TensorFlow 2.x, NumPy, Matplotlib, and scikit-image. A U-Net model with Residual LAB blocks was then implemented to further improve the PSNR. Despite these

efforts, the PSNR value achieved was still below the target threshold of 23, indicating a need for further optimization and refinement of the model architecture and training parameters. For the second case the input function had an activation function which was Relu and the output had an activation function which was Sigmoid. The data was trained for 75 epochs and 4 batch size.

TRAINING:

1) The U-Net model was trained utilizing the Charbonnier Loss function, which is a robust loss function less sensitive to outliers compared to traditional L2 loss. The Adam optimizer was employed for optimization, with a learning rate set to 1e-4. This configuration was chosen to enhance the convergence rate and stability of the training process.

2) The performance of the model was evaluated using the Peak Signal-to-Noise Ratio (PSNR) metric. PSNR is a widely used metric for assessing the quality of reconstructed images in comparison to their original versions. It quantifies the ratio between the maximum possible value of a signal and the power of the noise that distorts its representation. A higher PSNR value indicates better reconstruction quality, implying that the model effectively reduces noise and preserves the integrity of the original image. The PSNR is particularly useful in this context as it provides an objective measure of the improvement in image quality achieved by the denoising process. By employing these advanced techniques and metrics, the training aims to achieve significant enhancements in image restoration performance.