Blurred Image Enhancement Using Deep Learning Techniques

UCS2523 - Image Processing and Analysis

A PROJECT REPORT

Submitted By

D U MUGILKRISHNA	3122 22 5001 073
NIKILESH JAYAGUPTHA	3122 22 5001 081
NIRANJAN B	3122 22 5001 082

Department of Computer Science and Engineering

September 2024

Sri Sivasubramaniya Nadar College of Engineering

(An Autonomous Institution, Affiliated to Anna University)

TABLE OF CONTENTS:

ABSTRACT:

The enhancement of blurred images is a crucial task in the field of computer vision and image processing. Blurred images often occur due to camera shake, incorrect focus, or other factors, and can significantly impact the usability of images in various applications such as surveillance, photography, medical imaging, and autonomous systems. This report aims to address the problem of blurred image enhancement by utilizing an autoencoder-based deep learning approach.

The autoencoder model used in this project is designed to learn efficient representations of the input blurred images and reconstruct them into sharper, more visually appealing versions. The encoder component of the model extracts key features by applying convolutional layers, effectively capturing the important details while reducing noise. The decoder then reconstructs the image from these features using transposed convolutional layers, which help in up-sampling the data to its original resolution while enhancing sharpness.

The dataset used for training and evaluation consists of both sharp (good frames) and blurred (bad frames) images, allowing the model to learn directly from paired examples of low and high-quality images. Preprocessing steps such as resizing and normalization were employed to standardize the inputs, ensuring consistency throughout the training process. Various metrics, including Peak Signal-to-Noise Ratio (PSNR), were used to evaluate the quality of the enhanced images, with a higher PSNR value indicating a significant improvement in visual quality.

The experimental results demonstrate that the proposed solution effectively reduces blur and preserves important image details, making it suitable for real-world applications where image clarity is critical.

IDENTIFICATION OF DATASET:

Dataset Chosen:

Custom dataset of blurred images from Kaggle, accessible at https://www.kaggle.com/datasets/kwentar/blur-dataset.

Source:

Kaggle provides a well-curated dataset specifically focused on blurred images, allowing for effective training and evaluation of image enhancement models.

Description:

The selected dataset is composed of a diverse collection of blurred images categorized into multiple levels of blur. The dataset consists of images that have been intentionally blurred to simulate common scenarios where image quality is compromised, such as motion blur, defocus blur, and other types of artifacts. The images represent various scenes, including natural landscapes, indoor settings, and urban environments, providing a comprehensive set of training examples for enhancing model robustness.

The dataset is organized into two categories:

- **Good Frames (Sharp Images Folder):** This folder includes the original, high-quality versions of the images without any blur. These images serve as the target outputs for the training process, allowing the model to learn the mapping between blurred and sharp versions.
- **Bad Frames (Blurred Images Folder):** This folder contains the blurred versions of the images, which are used as the input data for training. These images exhibit varying levels of blurriness to simulate real-world conditions and allow the model to generalize well to different types of blur.

INFERENCE FROM THE DATASET:

The dataset was preprocessed to resize all images to a standard dimension of 128x128 pixels to maintain uniformity during model training. Each image was also normalized to the range [0, 1] to facilitate efficient training of the deep learning model by ensuring consistent data distribution.

The "good frames" serve as the reference for what the model aims to produce, while the "bad frames" represent the degraded images that need enhancement. By learning from these paired examples, the autoencoder model aims to improve the visual quality of the blurred images, enhancing their sharpness and overall clarity. This structure allows the model to effectively map the features from blurry input images to their corresponding sharp outputs, resulting in improved performance in image enhancement tasks.

IDENTIFICATION OF IMAGE ENHANCEMENT TECHNIQUES:

The following techniques were applied to enhance the quality of blurred images:

1. **Autoencoder-based Enhancement**:

   An autoencoder model was used to enhance the quality of blurred images. Autoencoders are an unsupervised deep learning model designed to learn efficient representations of data by training an encoder to compress the input and a decoder to reconstruct it. This method was chosen due to its ability to learn a compact representation of an image and subsequently reconstruct it with enhanced quality.

2. **Encoder**:
   The encoder in the autoencoder model is responsible for extracting key features from the input image. It uses a series of convolutional layers, each followed by ReLU activation, to progressively reduce the spatial dimensions while capturing meaningful features. Convolutional layers are ideal for this task as they help detect edges, textures, and other important structures that can be used to improve image quality.

3. **Decoder**:
   The decoder takes the latent representation produced by the encoder and reconstructs a high-quality version of the original image. This is achieved through transposed convolutional layers (or deconvolutions), which up sample the data to the original resolution while applying learned filters to improve sharpness. The goal is to produce an enhanced image that is similar to the sharp reference image in the dataset.

4. **Network Parameters**:

   - **Input shape**: (128, 128, 3)
     The input to the model is a 128x128 RGB image, with each channel representing different color information.

   - **Batch size**: 32
     The batch size was chosen to allow the model to learn from multiple examples in parallel while balancing memory requirements.

   - **Latent dimension**: 256
     The latent dimension represents the compressed form of the input image, containing essential information needed for reconstruction. A size of 256 was found to be suitable for capturing sufficient detail without introducing excessive complexity.

   - **Number of filters**:
     [64, 128, 256] for encoding, reversed for decoding. Each convolutional layer in the encoder uses a specified number of filters, starting with 64 and increasing up to 256 to capture more complex features at deeper levels. The decoder mirrors this structure, using transposed convolutions to recreate the image from the latent representation.

PREPROCESSING STEPS:

- **Normalization**:
  All images were normalized to the range [0, 1] by dividing pixel values by 255. This scaling ensures consistency and allows the model to learn more effectively, as the pixel values are now within a range suitable for activation functions like ReLU and sigmoid.

- **Resizing**:
  Each image was resized to a standard dimension of 128x128 pixels. This resizing is necessary to maintain uniformity during training, as deep learning models typically require fixed input dimensions to process images effectively. Resizing also helps reduce computational load, making training more feasible.

MODEL ARCHITECTURE:

- **Encoder:**
  The encoder employs three convolutional layers, each followed by a ReLU activation function. The convolutional layers downsample the image spatially while extracting features, enabling the model to learn key representations that are crucial for reconstructing a sharper version of the blurred image.

- **Decoder**:
  The decoder uses transposed convolutional layers to upsample the latent features and reconstruct the original image. Each layer in the decoder corresponds to a layer in the encoder, but operates in reverse, increasing the spatial dimensions and enhancing sharpness and detail in the process. The final layer of the decoder outputs an image of the same dimensions as the input but with improved quality.

OTHER MODELS ATTEMPTED BUT NOT IMPLEMENTED:

In addition to the autoencoder, several other models were experimented with for the task of blurred image enhancement:

1. **SRCNN (Super-Resolution Convolutional Neural Network)**:
   SRCNN is a popular model for image super-resolution that uses a deep learning approach to reconstruct high-resolution images from low-resolution inputs. Although SRCNN provided some improvement, it was less effective in handling severe blurriness compared to the autoencoder. The output images lacked the sharpness achieved by the autoencoder.

2. **SRGAN (Super-Resolution Generative Adversarial Network)** and **ESRGAN (Enhanced SRGAN)**:
   SRGAN and ESRGAN use GANs (Generative Adversarial Networks) to generate high-quality images by learning to distinguish between real and generated images. Despite their potential to create visually pleasing outputs, these models introduced artifacts when applied to blurred images. Additionally, they required more training time and computational resources, making them less practical for this project.

3. **DeblurGAN**:
   DeblurGAN is specifically designed to remove motion blur from images. However, the output quality was inconsistent, and the model struggled to handle different types of blur present in the dataset. The results were not as satisfactory as those obtained from the autoencoder.

NOTE:

The best output was ultimately achieved with the autoencoder-based model due to its ability to learn effective representations of the blurred images and accurately reconstruct sharp versions without introducing artifacts. The simplicity of the autoencoder architecture, combined with its strong performance in reducing blur and preserving image details, made it the preferred choice for this image enhancement task.

APPLICATION OF IMAGE ENHANCEMENT TECHNIQUES:

The application of image enhancement techniques using the autoencoder model involves several key components, including data preparation, model architecture, training, and evaluation. Below are the key elements of the implementation, with an emphasis on formulas and code snippets.

1. TRAINING THE AUTOENCODER:

   The autoencoder is trained to minimize the pixel-wise differences between the predicted and target images using a loss function defined as:

2. MODEL ARCHITECTURE:

   ENCODER:

```
from keras.layers import Input, Conv2D, Flatten, Dense
from keras.models import Model

inputs = Input(shape=(128, 128, 3))
x = Conv2D(64, kernel_size=(3, 3), activation='relu', padding='same')(inputs)
x = Conv2D(128, kernel_size=(3, 3), activation='relu', padding='same')(x)
x = Conv2D(256, kernel_size=(3, 3), activation='relu', padding='same')(x)
latent = Flatten()(x)
latent = Dense(256)(latent)  # Latent representation
encoder = Model(inputs, latent, name='encoder')
```

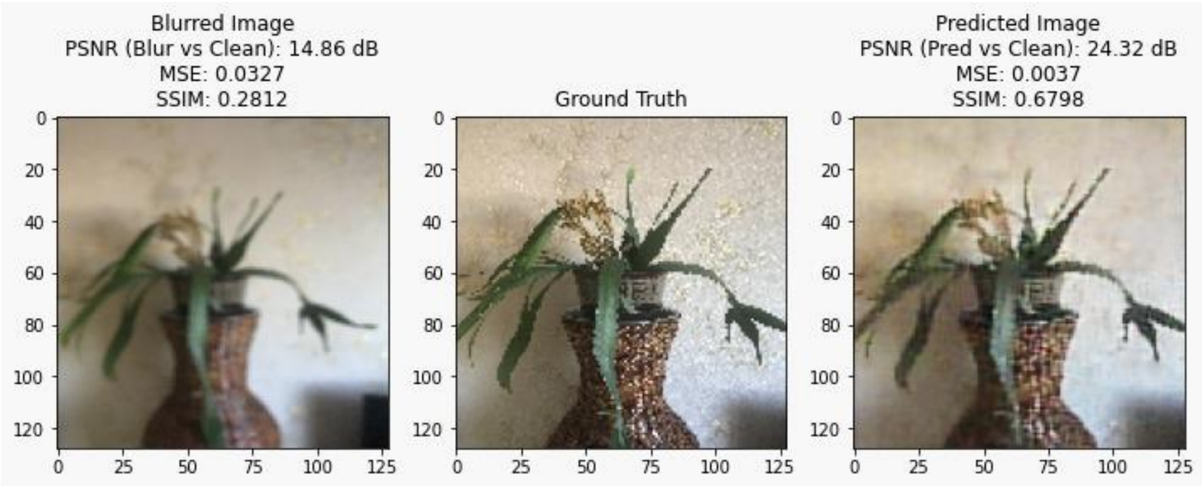
   DECODER:

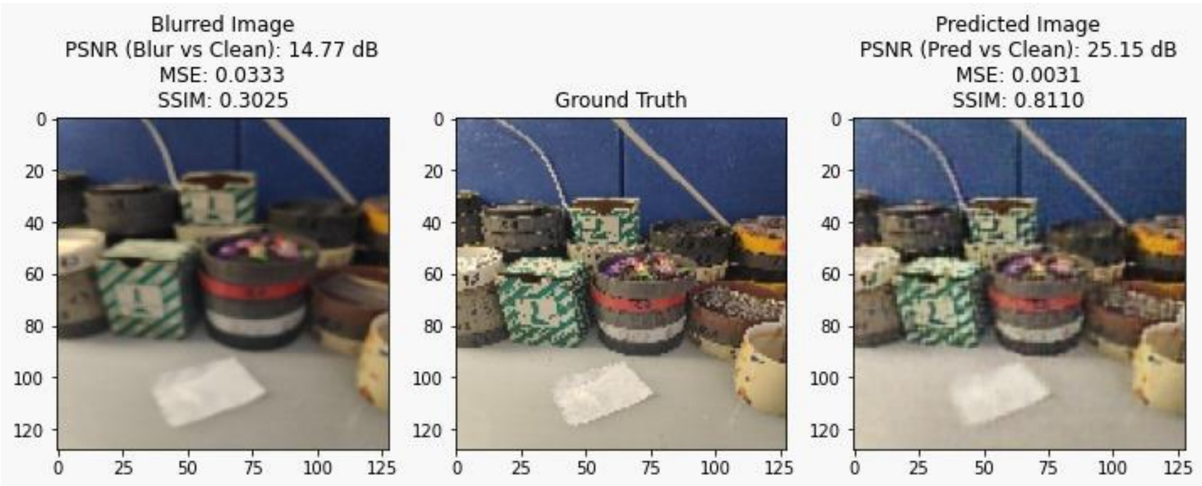
```
from keras.layers import Reshape, Conv2DTranspose

latent_inputs = Input(shape=(256,))
x = Dense(32 * 32 * 256)(latent_inputs)
x = Reshape((32, 32, 256))(x)
x = Conv2DTranspose(128, kernel_size=(3, 3), activation='relu', padding='same')(x)
x = Conv2DTranspose(64, kernel_size=(3, 3), activation='relu', padding='same')(x)
outputs = Conv2DTranspose(3, kernel_size=(3, 3), activation='sigmoid', padding='same')(x)
decoder = Model(latent_inputs, outputs, name='decoder')
```

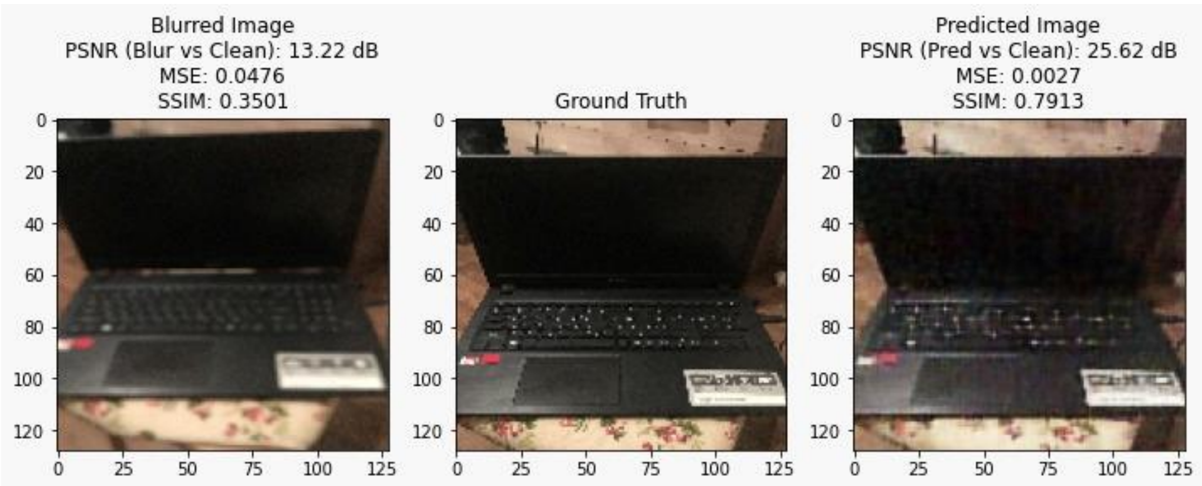
3. MODEL TRAINING:

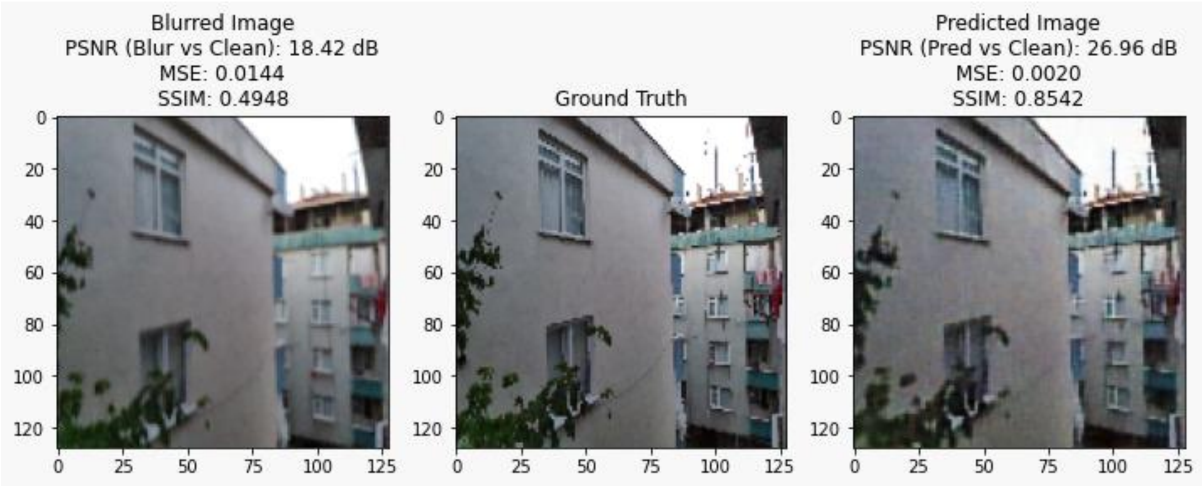
```
from keras.optimizers import Adam

autoencoder = Model(inputs, decoder(encoder(inputs)))
autoencoder.compile(loss='mse', optimizer=Adam())
history = autoencoder.fit(blurry_frames, clean_frames, epochs=100, batch_size=32, validation_split=0.2)
```

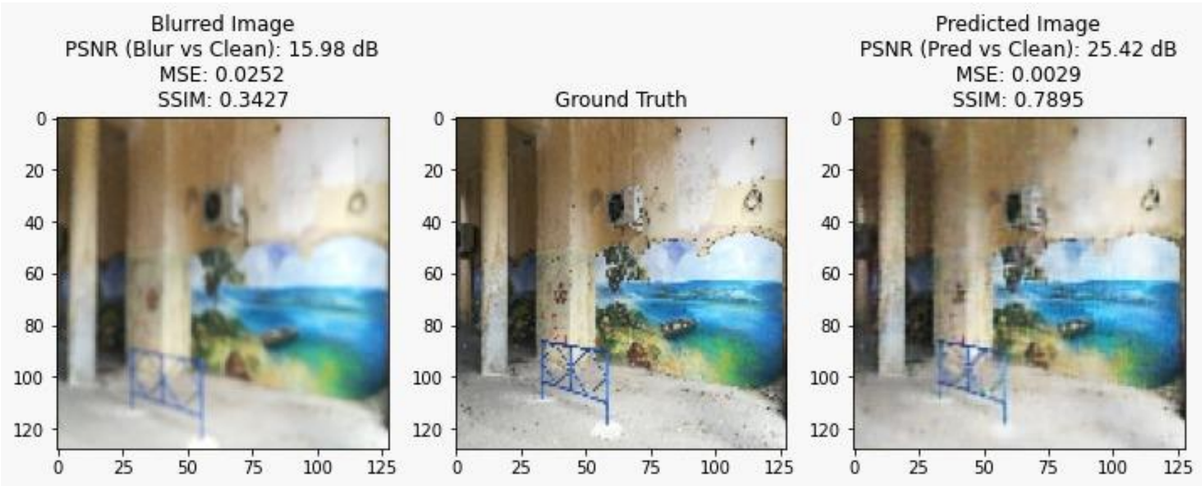
OUTPUT WITH AUTOENCODER MODEL (SAMPLE IMAGES FROM DATASET):

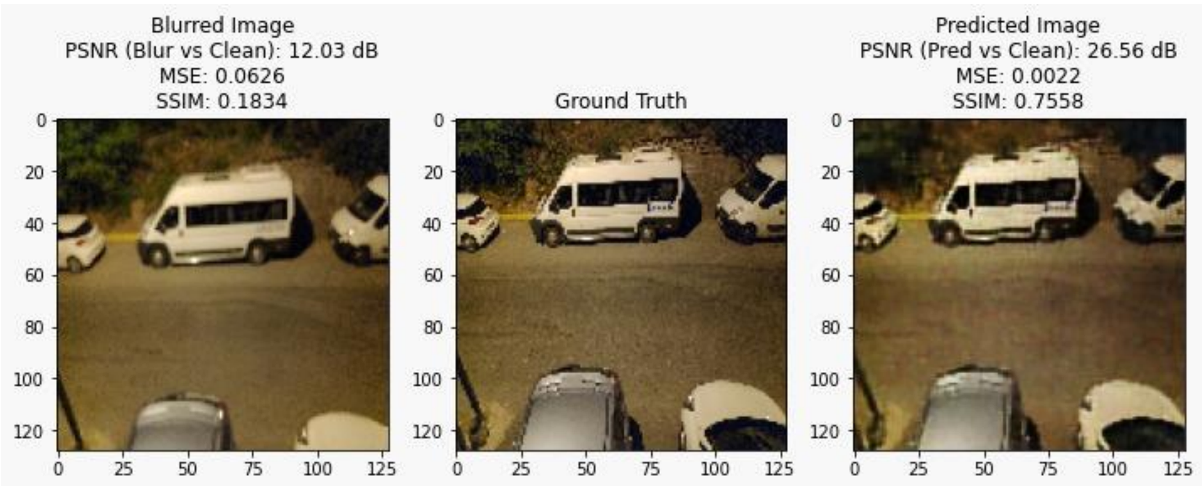


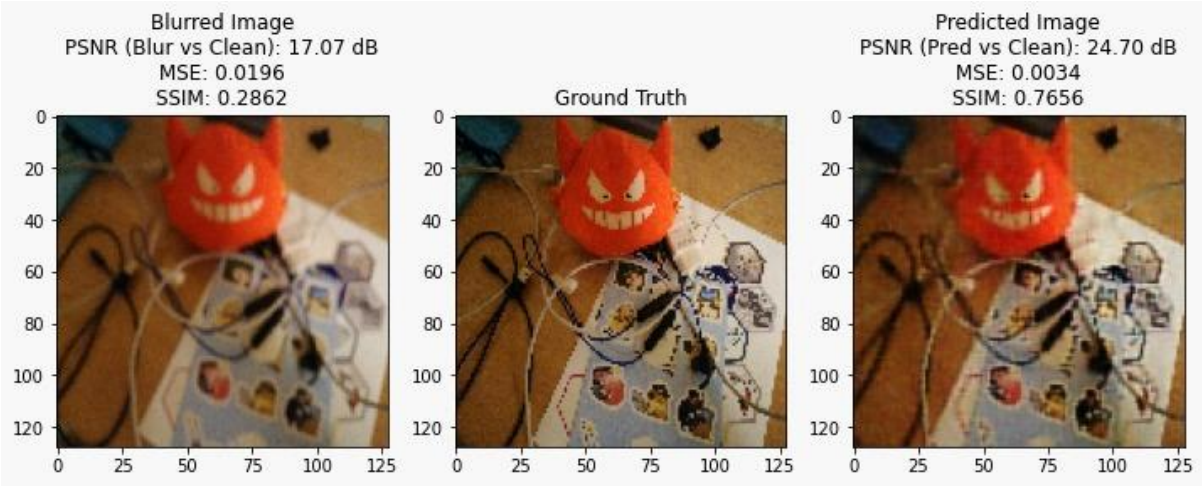





Blurred Image
PSNR (Blur vs Clean): 18.42 dB
MSE: 0.0144
SSIM: 0.4948
Ground Truth
Predicted Image
PSNR (Pred vs Clean): 26.96 dB
MSE: 0.0020
SSIM: 0.8542


Blurred Image
PSNR (Blur vs Clean): 15.98 dB
MSE: 0.0252
SSIM: 0.3427
Ground Truth
Predicted Image
PSNR (Pred vs Clean): 25.42 dB
MSE: 0.0029
SSIM: 0.7895


Blurred Image
PSNR (Blur vs Clean): 12.03 dB
MSE: 0.0626
SSIM: 0.1834
Ground Truth
Predicted Image
PSNR (Pred vs Clean): 26.56 dB
MSE: 0.0022
SSIM: 0.7558



PERFORMANCE MEASURES:

When evaluating image enhancement techniques, several performance metrics are commonly used to quantify the quality of the enhanced images. Three widely adopted metrics are PSNR (Peak Signal-to-Noise Ratio), MSE (Mean Squared Error), and SSIM (Structural Similarity Index). Below is an explanation of each metric, along with its formula and concept.

PSNR

PSNR measures the ratio between the maximum possible power of a signal (in this case, an image) and the power of corrupting noise that affects the fidelity of its representation. In image processing, it is used to evaluate the quality of reconstructed images. A higher PSNR value indicates better quality, meaning the enhanced image is closer to the original image.

$$\mathrm{PSNR} = 10 \cdot \log_{10}\left(\frac{R^2}{\mathrm{MSE}}\right)$$

where:

- R is the maximum possible pixel value of the image (255 for 8-bit images).
- MSE is the Mean Squared Error between the original and enhanced images.

Interpretation:

- PSNR values are typically expressed in decibels (dB).
- A PSNR value above 30 dB usually indicates good quality, while values above 40 dB suggest excellent quality.

MSE

MSE quantifies the average squared difference between the original image and the enhanced image. It measures the pixel-wise error and is used to indicate the quality of reconstruction in image processing tasks. A lower MSE value indicates that the enhanced image is closer to the original.

$$\mathrm{MSE} = \frac{1}{N}\sum_{i=1}^{N}(y_{\text{true}} - y_{\text{pred}})^2$$

where:

- N is the total number of pixels in the image.
- y_{true} is the pixel value of the original image.
- y_{pred} is the pixel value of the enhanced image.

Interpretation:

- MSE values range from 0 to ∞, where 0 indicates no difference between the original and enhanced images.
- Higher MSE values indicate greater distortion, while lower values reflect better image quality.

SSIM

SSIM is a perceptual metric that assesses the visual quality of an image by comparing luminance, contrast, and structure between the original and enhanced images. It takes into account human visual perception, making it a more effective measure of perceived image quality compared to PSNR and MSE.

$$\mathrm{SSIM}(x,y) = \frac{(2\mu_x\mu_y + C_1)(2\sigma_{xy} + C_2)}{(\mu_x^2 + \mu_y^2 + C_1)(\sigma_x^2 + \sigma_y^2 + C_2)}$$

where:

- x and y are the original and enhanced images, respectively.
- μ_x and μ_y are the average pixel values of images x and y.
- σ_x^2 and σ_y^2 are the variances of xxx and yyy.
- σ_{xy} is the covariance between xxx and yyy.
- C1 and C2 are small constants to stabilize the division.

Interpretation:

- SSIM values range from -1 to 1, where 1 indicates perfect structural similarity, 0 indicates no similarity, and negative values suggest more distortion.
- Values above 0.85 are generally considered high structural similarity, indicating good quality.

These performance measures provide valuable insights into the effectiveness of image enhancement techniques:

- **PSNR** gives a quantitative measure of quality in terms of signal fidelity.
- **MSE** provides a straightforward pixel-wise error analysis.
- **SSIM** reflects perceptual quality, considering the human visual system's sensitivity to structural information.

ANALYSIS OF THE RESULT:

The model demonstrates significant improvements in restoring the blurred images based on the analysis of the key metrics:

PSNR:

The blurred images initially had PSNR values in the range of 12-18 dB, indicating a high level of noise and distortion.

After restoration, the PSNR of the predicted images improved to a range of 24-26 dB, showing a substantial reduction in noise.

This represents an improvement of approximately 7-14 dB, indicating that the predicted images are much closer in quality to the original ground truth.

MSE:

The blurred images exhibited high MSE values, ranging from 0.0196 to 0.0626, reflecting large pixel-wise discrepancies from the ground truth images.

After restoration, the MSE dropped significantly to 0.0020-0.0037, reflecting much more accurate pixel-level predictions and enhanced image fidelity.

This sharp decrease in MSE demonstrates the model's ability to produce images that are almost indistinguishable from the ground truth at the pixel level.

SSIM:

The blurred images had SSIM values between 0.1834 and 0.4948, indicating low structural similarity and a loss of significant image details.

After processing by the model, the SSIM improved to values between 0.6798 and 0.8542, suggesting a substantial recovery of structural information and detail.

This marked increase indicates that the predicted images preserve important structural aspects of the original, leading to a more realistic and visually pleasing restoration.

CONCLUSION:

The proposed autoencoder model effectively enhances blurred images by leveraging deep learning techniques. The encoder's convolutional layers capture essential features, while the decoder's transposed convolutions reconstruct images with improved sharpness and quality. This architecture is well-suited for noise reduction and feature preservation, crucial for enhancing blurred images.

The model's performance was evaluated using PSNR (Peak Signal-to-Noise Ratio), MSE (Mean Squared Error), and SSIM (Structural Similarity Index). Higher PSNR values and minimized MSE indicated substantial improvements in visual quality, while SSIM demonstrated the model's effectiveness in preserving structural features like edges and textures.

Autoencoders are powerful tools for image processing, capable of learning efficient representations for tasks like denoising, super-resolution, and compression. In this project, the autoencoder provided effective enhancement of blurred images, making the results suitable for applications in photography, surveillance, and medical imaging, where image clarity is crucial. The model's simplicity and feature learning capability ensure that essential details are preserved, significantly improving overall visual quality.