Image Denoising Project

Introduction

Image denoising is used in image processing and computer vision, focused on enhancing image quality by eliminating noise. Noise can be introduced during image acquisition due to factors such as poor lighting conditions, sensor inaccuracies, or high ISO settings. In this project, I developed a model that transforms noisy images into denoised versions using a straightforward autoencoder architecture. This autoencoder compresses the noisy input images into a latent space and then reconstructs them as denoised images. I trained my model on a cifar-10 training dataset and assessed its performance(PSNR). I also applied my model on dental Xray images with Median filter which shows huge improvement in PSNR due to many normalisations.

Dataset

I have made 2 iypnb files one trained on cifar-10 and other Dental Xrayimages.
My dataset is divided into two parts: training and testing.

- **Training Dataset :**This consists of pairs of noisy (low-quality) images and their corresponding denoised (high-quality) images.
- **Testing Dataset:**This contains noisy images and their corresponding ground truth denoised images.

The noisy images in both datasets are referred to as "low" images, while the clean, denoised images are referred to as "high" or "ground" images in the training and testing datasets, respectively.The noisy images are also generated from original or clean images only.

Preprocessing

Before feeding the images into the model, we applied several preprocessing steps:

1. **Resizing:** We resized all images to a consistent size of 32*32 pixels.
2. **Grayscale Conversion:** Each image(RGB) was converted to grayscale mode ('L').This helps in reducing complexity
3. **Rescaling:** The pixel values of the images were rescaled to the range of 0 to 255.
4. **Reshaping**: Finally, the images were reshaped to the dimensions (32, 32, 1), suitable for input into the model.

Model Architecture

The core of our image denoising system is an autoencoder composed of two main parts: the encoder and the decoder.

Encoder

The encoder compresses the input image into a lower-dimensional latent space. It consists of Convulation layers(Their role is to extract the important/meaning feautures) and pooling layers(to reduce the spatial dimensions,so basically for dimensionality reduction):

First Convolutional Layer:

1. **Filters:** 32 filters
2. **Kernel Size:** (3, 3)
3. **Activation:** ReLU
4. **Padding:** 'same'
5. **Description:** The first convolutional layer convolves the input images with 32 filters, each of size 3x3 pixels. This operation captures local patterns and features from the input.

First Max-Pooling Layer:

1. **Pool Size:** (2, 2)
2. **Padding:** 'same'
3. **Description:** Following the first convolutional layer, a max-pooling operation with a pool size of 2x2 pixels is applied. This layer downsamples the spatial dimensions of the feature maps by retaining the maximum value within each pooling window, reducing computational complexity and preserving important features.

Second Convolutional Layer:

1. **Filters:** 32 filters
2. **Kernel Size:** (3, 3)
3. **Activation:** ReLU
4. **Padding:** 'same'
5. **Description:** The second convolutional layer further processes the feature maps obtained from the first max-pooling layer. It applies 32 additional filters of size 3x3 pixels to capture more complex patterns and features.

Second Max-Pooling Layer:

1. **Pool Size:** (2, 2)
2. **Padding:** 'same'
3. **Description:** After the second convolutional layer, another max-pooling operation with a pool size of 2x2 pixels is performed. This operation continues to reduce the spatial dimensions of the feature maps while maintaining important spatial relationships and features.

.

Decoder

The decoder in the denoising autoencoder is responsible for reconstructing denoised images from the compressed latent space representation obtained from the encoder. It consists of convolutional layers followed by upsampling operations to increase the spatial dimensions

First Convolutional Layer:

1. **Filters:** 32 filters

2. **Kernel Size:** (3, 3)
3. **Activation:** ReLU
4. **Padding:** 'same'
5. **Description:** The first convolutional layer takes the compressed latent representation as input. It applies 32 filters of size 3x3 pixels to extract features and begin the process of reconstructing the denoised image.

First Upsampling Layer:

1. **Size:** (2, 2)
2. **Description:** Following the first convolutional layer, an upsampling operation with a size of 2x2 pixels is applied. Upsampling increases the spatial dimensions of the feature maps, allowing the decoder to progressively reconstruct the denoised image.

Second Convolutional Layer:

1. **Filters:** 32 filters
2. **Kernel Size:** (3, 3)
3. **Activation:** ReLU
4. **Padding:** 'same'
5. **Description:** The second convolutional layer continues to process the feature maps obtained from the first upsampling layer. It applies 32 additional filters of size 3x3 pixels to capture more detailed features and refine the reconstruction.

Second Upsampling Layer:

1. **Size:** (2, 2)
2. **Description:** After the second convolutional layer, another upsampling operation with a size of 2x2 pixels is performed. This operation further increases the spatial dimensions of the feature maps, facilitating the reconstruction of the denoised image.

Final Convolutional Layer:

1. **Filters:** 3 filters (corresponding to RGB channels)
2. **Kernel Size:** (3, 3)
3. **Activation:** Sigmoid
4. **Padding:** 'same'
5. **Description:** The final convolutional layer converts the processed feature maps into the output denoised image. It applies 3 filters of size 3x3 pixels, corresponding to the RGB channels of the image, and uses the sigmoid activation function to ensure pixel values are between 0 and 1.

Training

The model was compiled using the Adam optimization algorithm and binary cross-entropy loss function. We split the training data into training and validation sets and employed early stopping to prevent overfitting.

Testing

To test our performance of our model, we used the trained model to predict the denoised images from the noisy images in the test dataset. The performance was measured using the Peak Signal-to-Noise Ratio (PSNR), a widely used metric in image processing to assess the quality of reconstructed images.

The avg PSNR value obtained for my test dataset was 18.75dB.

$$
\begin{aligned}
PSNR &= 10 \cdot \log_{10} \left(\frac{MAX_I^2}{MSE} \right) \\
&= 20 \cdot \log_{10} \left(\frac{MAX_I}{\sqrt{MSE}} \right) \\
&= 20 \cdot \log_{10}(MAX_I) - 10 \cdot \log_{10}(MSE).
\end{aligned}
$$

here MSE is mean squared error ans MAX=255 the maximum pixel value of our image

Model Performance

The autoencoder model was able to learn and identify patterns upto a extent, as indicated by the PSNR value. However, a PSNR of ~19 suggests that there is significant room for improvement. Typically, higher PSNR values (30 and above) are desirable for high-quality denoising. The moderate PSNR achieved can be attributed to several factors, including the complexity of the noise, the simplicity of the autoencoder architecture, and the computational inability.

Challenges and Limitations

1. Preservation of important information along with removing noise can sometimes lead to tradeoff between noise reduction and preservation of image feautures.
2. Computational Complexity and limited resources
3. Generalization: They may not adopt and generalise immediately to unseen noise
4. Overfitting : As 3rd point itself says many times models are overfit as of to not compromise performance.

Future Work

To enhance the model's performance, several improvements and extensions can be considered:

1. **Advanced Architectures**: Implementing more sophisticated models like U-Net, residual networks, or GANs (Generative Adversarial Networks) specifically designed for image denoising.
2. **Data Augmentation**: Applying various data augmentation techniques to increase the diversity and size of the training dataset, which could help the model generalize better.
3. **Hyperparameter Tuning**: Conducting extensive hyperparameter tuning, including experimenting with different optimizers, learning rates, batch sizes, and loss functions.
4. **Multi-Channel Images**: Extending the model to handle multi-channel (RGB) images instead of grayscale, which could provide more information for the denoising process.

Conclusion

In this project, I developed a simple autoencoder-based model for image denoising. The model was able to reduce noise to some extent, achieving a average PSNR value of 19 on the test dataset. While

this performance is moderate, it highlights the potential of autoencoder-based approaches for image denoising. Future work will focus on improving the model's architecture, optimizing the training process, and exploring advanced techniques to achieve higher quality denoised images.