Pranshi Singh
22323034
3rd Year
Mathematics and Computing

Enhancing Image Quality with RIDNet Model

18th june 2024

OVERVIEW

This project aims to enhance the quality of noisy images using the Residual Image Denoising Network (RIDNet). The RIDNet model is trained to denoise images by learning the underlying clean image patterns from noisy inputs. The model uses advanced convolutional neural network (CNN) techniques, including the Enhanced Attention Module (EAM), to achieve high performance in image denoising tasks. ** Idhar libraries kaunsi kaunsi use ki hai vo daalo**

DATASET PREPARATION

The dataset consists of paired high-quality (clean) and low-quality (noisy) images. The images were stored in a zip file and extracted to Google Drive for processing. The following steps were performed to prepare the dataset:

1. **Extraction:** The zip file containing the dataset was extracted to a specified directory in Google Drive.

```python
import zipfile
import os
zip_file_path = '/content/drive/My Drive/Train.zip'
extract_dir = '/content/extracted_dataset'
os.makedirs(extract_dir, exist_ok=True)

with zipfile.ZipFile(zip_file_path, 'r') as zip_ref:
    zip_ref.extractall(extract_dir)

print("extraction done!!!")
```

2. **Loading Images:** Images were loaded from the 'high' (clean) and 'low' (noisy) directories. Images were resized to 512 x 512 pixels to ensure uniformity.

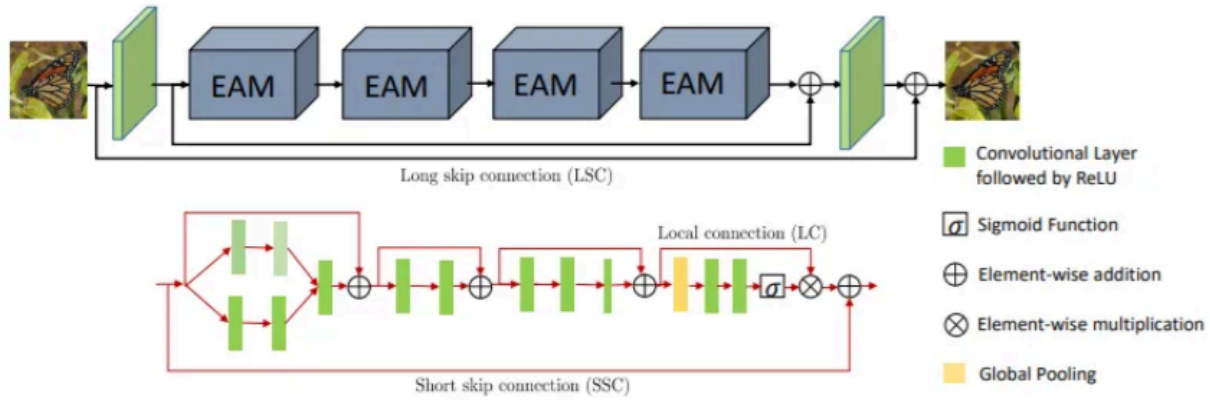
```
def load_and_patch_images(folder, image_size, patch_size):
    patches = []
    for filename in os.listdir(folder):
        img_path = os.path.join(folder, filename)
        try:
            img = load_img(img_path, target_size=image_size)
            img_array = img_to_array(img)
            img_patches = create_patches(img_array, patch_size)
            patches.extend(img_patches)
        except Exception as e:
            print(f"Error loading image {img_path}: {e}")
    return np.array(patches)

# Parameters
image_size = (512, 512)  # Resize images to this size
patch_size = 256  # Patch size
```

3. **Creating Patches:** Each image was divided into 256 x 256 pixel patches, resulting in 4 patches per image. This step was necessary to handle large images effectively and increase the training data**.**

```
def create_patches(image, patch_size):
    patches = []
    h, w, _ = image.shape
    for i in range(0, h, patch_size):
        for j in range(0, w, patch_size):
            patch = image[i:i+patch_size, j:j+patch_size]
            patches.append(patch)
    return np.array(patches)
```

MODEL ARCHITECTURE



Different green colors of the convolution layers denote different dilations while the smaller size of the convolution layer means the kernel is 1 × 1. The second row shows the architecture of each EAM.

This network is composed of three main modules as follows :

- **Feature Extraction Module:** It is composed of only one convolutional layer to extract initial features from the noisy input. I've used 64 filters with kernel size=3 for the convolutional layer.

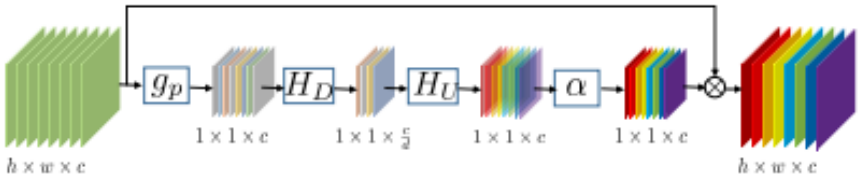


Figure 3. The feature attention mechanism for selecting the essential features.

- **Feature Learning Residual on Residual Module:** It is composed of a network called Enhancement Attention Modules (EAM) that uses a residual on the residual structure with local skip and short skip connections. The initial part of EAM uses wide receptive fields through kernel dilation and branched convolutions thereby capturing global and diverse information from the input

image. Additional features are learned using a residual block of two convolutions followed by an enhanced residual block (ERB) of three convolutions. Finally, it is given to a feature attention block that gives more weight to the important features.

We can increase the depth of the RIDNet network by increasing the number of EAM blocks. However, in the research paper, they restricted the network to four EAM blocks only.

- **Reconstruction Module:** The output of the final EAM block is given to the reconstruction module which is again composed of only one convolutional layer that gives the denoised image as output.

TRAINING PROCEDURE

- **Training Loop :** The model is trained for 15 epochs, with each epoch involving multiple iterations over the entire training dataset.
- **Batch Processing :**
    1. The data loader provides batches of data to the model. Each batch contains 4 patches (batch size = 8).
    2. For each batch, the model performs a forward pass to predict the denoised images.
    3. The loss (MSE) is calculated by comparing the predicted denoised images to the ground truth high-quality images.
    4. The optimizer updates the model parameters to minimize the loss.

PERFORMANCE METRICS

Performance was evaluated using the Peak Signal-to-Noise Ratio (PSNR) metric, which measures the quality of the denoised image compared to the original clean image.

The mathematical representation of the **PSNR** is as follows:

$$PSNR = 20\log_{10}\left(\frac{MAX_f}{\sqrt{MSE}}\right)$$

Figure 1 - Peak Signal-to-Noise Equation

where the **MSE** (*Mean Squared Error*) is:

$$MSE = \frac{1}{mn}\sum_{0}^{m-1}\sum_{0}^{n-1}\|f(i,j) - g(i,j)\|^2$$

Figure 2 - Mean Squared Error Equation

This can also be represented in a text based format as:

*MSE = (1/(m*n))*sum(sum((f-g).^2))*

*PSNR = 20*log(max(max(f)))/((MSE)^0.5)*

Legend:
f represents the matrix data of our original image
g represents the matrix data of our degraded image in question
m represents the numbers of rows of pixels of the images and i represents the index of that row
n represents the number of columns of pixels of the image and j represents the index of that column
MAX_f is the maximum signal value that exists in our original "known to be good" image

```python
def psnr(y_true, y_pred):
    max_pixel = 1.0
    return tf.image.psnr(y_true, y_pred, max_val=max_pixel)
```

Implementation using TensorFlow

After the training is complete, the model is evaluated on the test set to obtain the final loss and PSNR values. The final PSNR value achieved is **19.96,** indicating the model's ability to denoise images.

REFERENCES

- Real Image Denoising with Feature Attention
- A blog on image denoising
- Peak Signal-to-Noise Ratio