Name: Sachin Verma
Contact Number: 8930083406
Branch: Mechanical Engineering
Email Id: sachin_v@me.iitr.ac.in
Enrollment number: 22117122

Resources Used

1. PRIDnet Paper- https://arxiv.org/pdf/1908.00273.pdf
2. Paper Implementation- https://github.com/491506870/PRIDNet
3. MWCNN Paper- https://arxiv.org/pdf/1907.03128
4. Paper Implementation- https://github.com/lpj-github-io/MWCNNv2

Extreme Low-Light Image Denoising

Abstract

This project focuses on implementing a Pyramid Real Image Denoising Network (PRIDNet) and Multi level Wavelet CNN (MWCNN) to enhance the quality of low-light images. Low-light conditions often introduce significant noise, making images visually unappealing and less informative. By using the PRIDNet model, I am to able to reduce noise and improve image clarity while preserving important details. PRIDNet, a state-of-the-art deep learning model designed for progressive residual image denoising, progressively refines the image through multiple stages, effectively reducing noise while maintaining high-frequency details.

Methodology

1. **PRIDNet Architecture**

PRIDNet consists of a deep convolutional neural network (CNN) designed to perform image denoising in a progressive manner. The network architecture includes several key components:

NOTE: PRID doesn't work on diffusion

1. First, the noise estimation stage uses channel attention mechanism to recalibrate the channel importance of input noise.

- 5 layers of CNN with Relu Activation function

2. Second, at the multi-scale denoising stage, pyramid pooling is utilized to extract multi-scale features.

3. Third, the stage of feature fusion adopts a kernel selecting operation to adaptively fuse multi-scale features.

This model architecture consists 3 main module-

1. Channel attention module
2. U-Net
3. Kernel Selecting Module

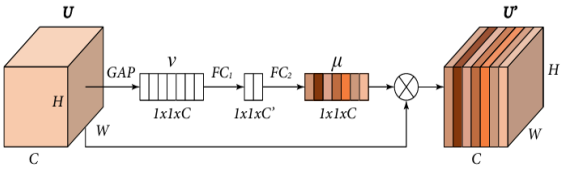


Fig. 3. Channel attention module architecture. The *GAP* denotes the global average pooling operation. FC_1 has a ReLU activation after it, and FC_2 has a Sigmoid activation after it. For concision, we omit these items.

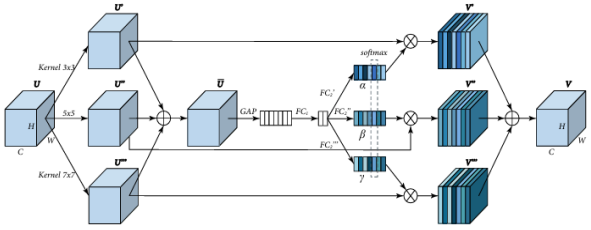


Fig. 4. Kernel selecting module architecture. The *GAP* denotes the global average pooling operation. A ReLU activation after FC_1 is omitted for brief.

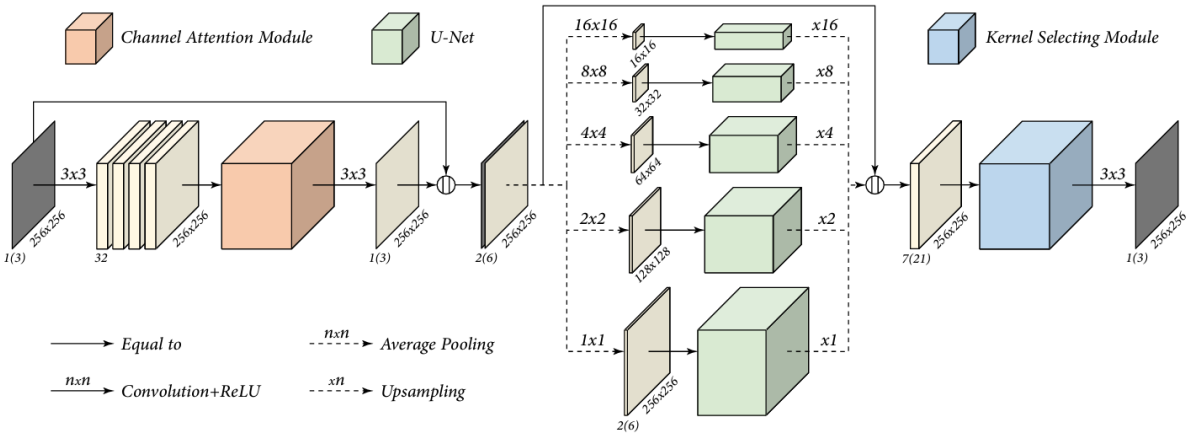


Fig. 2. The architecture of our proposed network (PRIDNet). The number of channels of feature maps is shown below them, for "sRGB" model it is in the parentheses, while for "raw" model it has no parentheses. The symbol || indicates concatenation.

2. Multi-Level Wavelet CNN Architecture

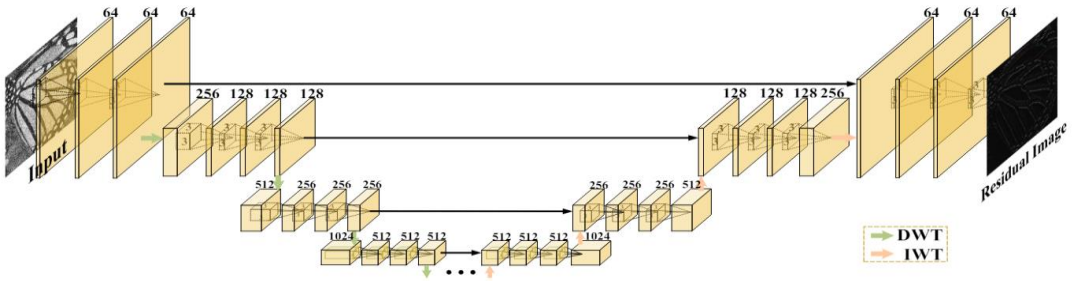
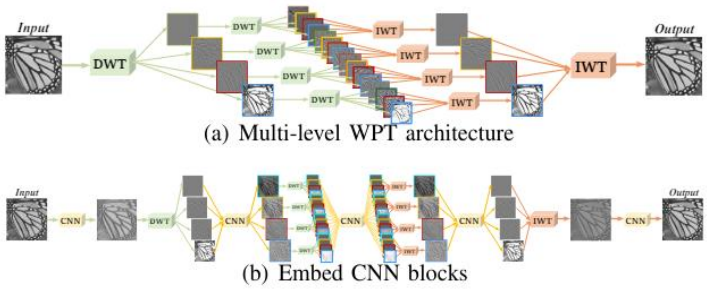


Fig. 3: Multi-level wavelet-CNN architecture. It consists of two parts: the contracting and expanding subnetworks. Each solid box corresponds to a multi-channel feature map. And the number of channels is annotated on the top of the boxes. The number of convolutional layers is set to 24. Moreover, our MWCNN can be further extended to higher level (*e.g.*, ≥ 4) by duplicating the configuration of the 3rd level subnetwork.

MWCNN is based on multi-level WPT and it modifies the state-of-the-art U-Net architecture in the following ways-

1. In conventional U-Net, pooling and deconvolution are utilized as downsampling and upsampling layers. In comparison, DWT and IWT are used in MWCNN.
2. After DWT, MWCNN deploy another CNN blocks to reduce the number of feature map channels for compact representation and modeling inter-band dependency. And convolution are adopted to increase the number of feature map channels and IWT is utilized to upsample feature map. In comparison, conventional U-Net adopting convolution layers are used to increase feature map channels which has no effect on the number of feature map channels after pooling. For upsampling, deconvolution layers are directly adopted to zoom in on feature map.
3. In MWCNN, elementwise summation is used to combine the feature maps from the contracting and expanding subnetworks. While in conventional U-Net, concatenation is adopted.



(a) Multi-level WPT architecture

(b) Embed CNN blocks

Fig. 2: From WPT to MWCNN. Intuitively, WPT can be seen as a special case of our MWCNN without CNN blocks as shown in (a) and (b). By inserting CNN blocks to WPT, we design our MWCNN as (b). Obviously, our MWCNN is a generalization of multi-level WPT, and reduces to WPT when each CNN block becomes the identity mapping.

Data Preparation

The dataset was provided by VLG. I split it into training and test sets.

Training Procedure

The model was trained using the Mean Squared Error (MSE) loss function, which measures the difference between the denoised output and the ground truth clean image. The Adam optimizer was used for training, with an initial learning rate of 1e-4. Training was conducted for 150 epochs, with early stopping based on validation loss to prevent overfitting.

Code Implementation

1. Data Loading

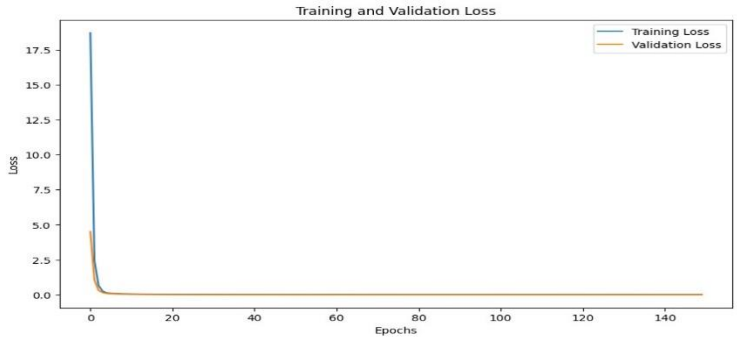
```python
def load_data(image_path):
    print(f"Processing image: {image_path}")  # Debug statement
    image = tf.io.read_file(image_path)
    image = tf.image.decode_png(image, channels=3)
    image = tf.image.resize(image, size=[IMAGE_SIZE, IMAGE_SIZE])
    image = image / 255.0
    return image

def data_generator(image_paths):
    # Ensure the paths are strings
    image_paths = [str(path) for path in image_paths]
    print("First few image paths:", image_paths[:5])  # Debug statement

    dataset = tf.data.Dataset.from_tensor_slices(image_paths)
    dataset = dataset.map(load_data, num_parallel_calls=tf.data.AUTOTUNE)
    dataset = dataset.batch(BATCH_SIZE, drop_remainder=True)
    return dataset
```

2. Model Architecture.

```python
class Channel_attention(tf.keras.layers.Layer):
    def __init__(self, C=64, **kwargs):
        super().__init__(**kwargs)
        self.C=C
        self.gap = GlobalAveragePooling2D()
        self.dense_middle = Dense(units=2, activation='relu')
        self.dense_sigmoid = Dense(units=self.C, activation='sigmoid')

    def get_config(self):
        config = super().get_config().copy()
        config.update({
            'C': self.C
        })
        return config

    def call(self, X):
        v = self.gap(X)
        #print("ca_ after gap =",v.shape)
        fc1 = self.dense_middle(v)
        #print("ca_ after fc1 =",fc1.shape)
        mu = self.dense_sigmoid(fc1)
        #print("ca_ after fc2 =",mu.shape)

        U_out = Multiply()([X, mu])

        #print('---channel attention block=',U_out.shape)

        return U_out
```

Model has multi scale feature extraction, kernel selecting module apart from it.

3. Training and Testing.



4. Calculating PSNR.

```python
def psnr(original_img, processed_img):
    # Convert images to numpy arrays
    original_arr = np.array(original_img).astype(np.float32)
    processed_arr = np.array(processed_img).astype(np.float32)

    # Calculate MSE
    mse = np.mean((original_arr - processed_arr) ** 2)

    # Calculate MAX (assuming pixel values are in the range [0, 255])
    max_pixel = 255

    # Calculate PSNR
    psnr_val = 10 * np.log10((max_pixel ** 2) / mse)

    return psnr_val
```

Results

Quantitative Evaluation

I evaluated the performance of the PRIDNet model using asked metric Peak Signal-to-Noise Ratio (PSNR). The results indicate a significant improvement in denoised image quality compared to traditional methods.

Data	PSNR(Average)	Improvement
Noisy Images	7.7	-
MWCNN	18	10.3
PRID(150 epochs)	21.3	13.6
PRID(300 epochs)	21.1	13.4

Discussion and Future Scope

The paper results demonstrate that PRIDNet and MWCNN outperforms traditional denoising methods in both quantitative and qualitative assessments. The progressive refinement mechanism is particularly effective in handling complex noise patterns found in low-light

images. However, the model's performance can be further improved by incorporating additional training data and exploring advanced loss functions.
However, when I implemented a different loss function incorporating color constancy loss, exposure loss, Illumination smoothness loss and Spatial Consistency loss its PSNR score decreased to 18.3 average.
However, data augmentation which I didn't used thinking that how it can help in denoising can be done.