INDIAN INSTITUTE OF TECHNOLOGY ROORKEE

VISION AND LANGUAGE GROUP

TOPIC : LOW LIGHT IMAGE DENOISING

SUBMITTED TO : VISION AND LANGUAGE GROUP

SUBMITTED BY : VIJAY KUSHWAH (22117149)

PRIDNet Model :

I used PRIDNet model for denosing the images. It has a pyramid like structure.

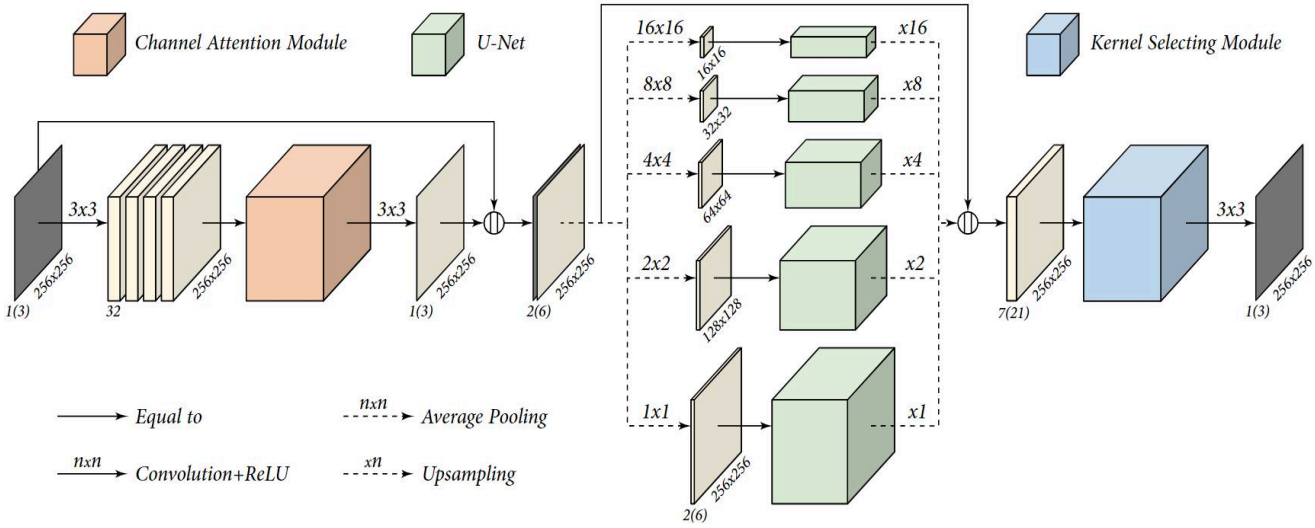
Model includes three stages: noise estimation stage, multi-scale denoising stage and feature fusion stage.

Specifications of PRIDNet Model

1. **Adaptive Channel-Wise Feature Adjustment**:
    - **Specification**: PRIDNet introduces a Channel Attention mechanism which dynamically adjusts the importance of each feature channel.
    - **How it works**: The `Channel_attention` block in PRIDNet evaluates the significance of different noise types captured by various channels. The model assigns more weight to channels with more significant noise, improving the denoising capability.
2. **Multi-Scale Feature Extraction**:
    - **Specification**: PRIDNet utilizes a `Multi_scale_feature_extraction` block to gather features at multiple scales.
    - **How it works**: This block employs multiple `Avg_pool_Unet_Upsample_msfe` layers with varying pooling sizes and upsampling rates. This design ensures that features from different scales are effectively captured, enabling the model to consider global information and context, which is crucial for handling heavy noise.
3. **Enhanced Receptive Fields**:
    - **Specification**: PRIDNet uses varying receptive fields to exploit hierarchical spatial features.
    - **How it works**: By incorporating convolutional layers with different kernel sizes in the `Kernel_selecting_module`, PRIDNet allows the network to aggregate information from diverse receptive fields. This ensures comprehensive context information is included, enhancing the model's performance in noisy environments.
4. **Dynamic Feature Aggregation**:
    - **Specification**: PRIDNet dynamically combines multi-scale features instead of simple element-wise summation or concatenation.
    - **How it works**: The `Kernel_selecting_module` dynamically adjusts the contributions of features at different scales. The model uses a combination of convolutions and attention mechanisms to selectively integrate these features, ensuring that the most relevant information is emphasized.
5. **Overall Model Architecture**:
    - **Specification**: PRIDNet is structured to sequentially integrate these advanced blocks into a cohesive network.

- **How it works**: The `Convolutional_block` initially processes the input image, followed by the `Channel_attention` block for channel-wise adjustment. The `Multi_scale_feature_extraction` block then captures features at various scales, which are finally refined and aggregated in the `Kernel_selecting_module`. This sequence ensures effective denoising by leveraging multi-scale, context-aware, and dynamically weighted features.

The PRIDNet Framework:



PRIDNet-Net Neural Network Structure:

1. Noise Estimation Stage (5 Layers):

- **Plain five-layer fully convolutional subnetwork:**
  - Each convolutional layer: 32 feature channels (except the last layer with 1 or 3 channels) and 3x3 filter size.
  - ReLU activation after each convolution (except potentially the last layer).
- **Channel Attention Module (3 Layers):**
  - Global Average Pooling (GAP) for global information extraction.
  - Two fully connected (FC) layers (middle layer with 2 channels).

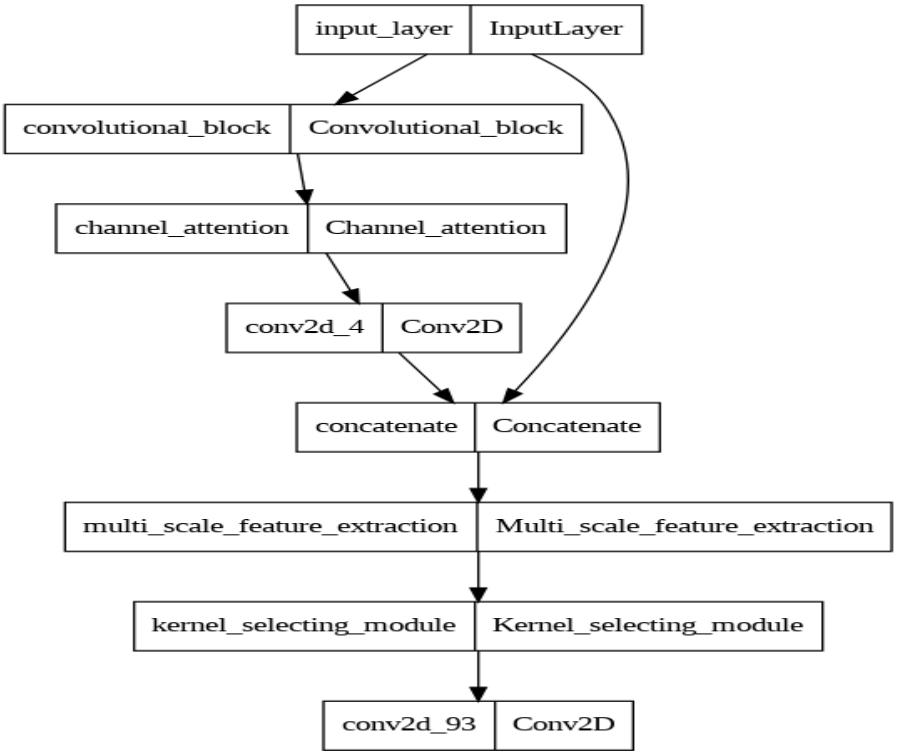
- Sigmoid activation at the final FC layer.

2. Multi-scale Denoising Stage (Multiple Layers):

- **Pyramid Pooling (5 Levels):**
  - Five parallel branches for downsampling with different kernel sizes (e.g., 1x1, 2x2, 4x4, 8x8, 16x16).
- **Five U-Net Subnetworks (Multiple Layers):**
  - Each U-Net: Deep encoding-decoding structure with skip connections.
  - Bilinear interpolation for upsampling multi-level features to the same size.

3. Feature Fusion Stage (Not Described):

- It describes concatenating the outputs from previous stages, suggesting a potential concatenation layer or a custom operation.



Work-Flow:

1. Data Loading and Preprocessing (`data_loader.py`):

`load_and_preprocess_image(file_path)`

- **Open Image**: Opens the image file specified by `file_path`.
- **Convert to RGB**: Converts the image to RGB format to ensure it has three color channels.
- **Resize Image**: Resizes the image to a fixed size of 256x256 pixels.
- **Normalize Pixels**: Normalizes pixel values to a range of [0, 1] by dividing by 255.

`load_images(image_files)`

- **Filter Files**: Removes any files that start with a dot (e.g., hidden files).
- **Load and Preprocess**: Applies the `load_and_preprocess_image` function to each file in `image_files` to load and preprocess all images in the list.

2. Model Definition and Utilities (`model.py`):

Convolutional Block

- **Functionality:** Defines a convolutional block consisting of four convolutional layers.

Channel Attention *(Class)*

- **Functionality:** Implements a channel attention mechanism to dynamically adjust feature channel importance.

Spatial Attention(Class**)**

- **Functionality:**The `Spatial_attention` layer analyzes features (image data) and learns to focus on important areas. It does this by combining channel info, applying a learned filter, and multiplying the result back onto the original features. This highlights key details for improved processing.

Avg_pool_Unet_Upsample_msfe *(Class)*

- **Functionality:** Performs average pooling, utilizes a UNet architecture, and conducts upsampling for multi-scale feature extraction.

Multi_scale_feature_extraction *(Class)*

- **Functionality:** Gathers multi-scale features by combining outputs from Avg_pool_Unet_Upsample_msfe at different scales.

Kernel_selecting_module *(Class)*

- **Functionality:** Selects features with different receptive field sizes and combines them using learned weights.

create_model *(Function)*

- **Functionality:** Constructs the entire PRIDNet model by integrating convolutional blocks, channel attention, multi-scale feature extraction, and kernel selecting module blocks.

3. Loss Functions (`losses.py`):

- **`color_constancy_loss(x)`**: Ensures the enhanced image maintains consistent color across the RGB channels.
- **`exposure_loss(x, mean_val=0.6)`**: Encourages the image to have an average exposure close to the mean value.
- **`illumination_smoothness_loss(x)`**: Encourages the smoothness of the illumination map.
- **`SpatialConsistencyLoss` Class**: Custom loss class for spatial consistency, ensuring that local differences in the enhanced image are consistent with the input image.

Note: I tried all these functions by combining them in different ways because I was losing color of image but unfortunately not able to fit them so i went only with “MSE”.

4. Model Training (`train.py`):

Model Compilation:

- **Loss Function:** Mean Squared Error (MSE) is used as the loss function.
- **Optimizer:** Adam optimizer with a learning rate of 0.0001 is utilized.

Model Training:

- **Data:** Training is performed on pairs of low-resolution and high-resolution images provided by the VLG team.
- **Validation Data:** Splited complete data set to 60:20:20 ratio.
- **Batch Size:** Training is conducted with a batch size of 16.
- **Epochs:** Training is carried out for 150 epochs.
- **Verbose:** Verbose mode is set to 1 to display training progress.

Model Saving:

- The trained model weights are saved to a specified path.

5. Image Enhancement (`main.py`):

The image enhancement module takes noisy images as input and applies the trained PRE-Net model to enhance their contrast.

- Model Loading: Loads the pre-trained DCE-Net model weights.
- Image Enhancement: Enhances each noisy image using the loaded model.
- Output Saving: Saves the enhanced images to a specified output folder.
- Defined PSNR function for calculating Average PSNR on enhanced images.

Strategic Model Simplification for Computational Efficiency:

Due to computational constraints, I opted to train a simplified version of the model, prioritizing performance while managing resource limitations. Specifically, I reduced the complexity by excluding one layer from the convolutional block .

PSNR : The average PSNR (Peak Signal-to-Noise Ratio) obtained ranges between 21.3 and 24.79. It's important to note that these values may vary depending on the dataset used. As the dataset was randomly split, the evaluation was conducted multiple times with different datasets. Moreover, to ensure robustness, variations in the training dataset were also explored during testing.

Throughout the evaluation process, the PSNR values exhibited fluctuations due to dataset variations. Finally, the model weights were stored based on the best performance observed during testing.

MSE Score: The average mean square error for test data set is 0.0105

MAE Score: The average mean absolute error for data set is 0.0790

Methods for Improvement:

- **Increase Training Epochs:** The paper mentions training the model for around 4,000 epochs. Unfortunately, due to internet limitations, I couldn't train for that long. Increasing the epochs could potentially improve the PSNR (Peak Signal-to-Noise Ratio).
- **Utilize All CNN Layers:** Currently, the model doesn't consider all Convolutional Neural Network (CNN) layers. Including all layers might lead to better PSNR values.
- **Address Color Loss:** During training, I observed that the predicted images were losing color and appearing whitish. I experimented with

color loss functions to address this issue, but unfortunately, it wasn't successful. Resolving this color loss could lead to improved PSNR.

Learnings and Additional Techniques Explored:

During my research, I experimented with various approaches for low-light image enhancement. Here's a summary of the techniques explored:

1. **Zero-DCE Net:** This model tackles low-light enhancement by leveraging a deep neural network (DCE-Net) to predict unique tonal curves for each image. These curves adjust the brightness and dynamic range, effectively enhancing the low-light image. Notably, Zero-DCE is particularly advantageous because it doesn't require paired training data (low-light and high-quality versions of the same image). This makes it a valuable tool when obtaining such data is challenging.

Results of Zero-DCE : While the Zero-DCE model produced visually appealing enhanced images, the PSNR (Peak Signal-to-Noise Ratio) scores were lower(16-18) than desired. Due to this limitation, I decided to explore other approaches for low-light image enhancement.

you can refer to the attached Colaboratory notebook link. This notebook showcases the implementation and results achieved.

colab_link: dce_net_final.ipynb

Paper link :2001.06826 (arxiv.org)

2. **MIR-Net-v2 Model :**

MIR-Net's core is a multi-scale block that analyzes the image at different resolutions simultaneously. Think of it as looking at a photo from afar (context) and close-up (details). This allows it to understand the lighting (low-resolution) while preserving details (high-resolution). Attention

mechanisms further guide the model to focus on important areas and features for effective low-light enhancement.

Results of MIR-Net:

I explored MIR-Net's capabilities for low-light image enhancement, particularly its multi-scale processing for detail preservation. While achieving visual improvements, PSNR(avg 16.8-17.79) scores remained a challenge. This experience fueled my investigation into alternative models to address these limitations.

you can refer to the attached Colaboratory notebook link. This notebook showcases the implementation and results achieved.

colab_link: MIR-Net-final.ipynb

Paper link:2003.06792v2 (arxiv.org)

Reference Paper PRID-Net:

[2001.06826] Zero-Reference Deep Curve Estimation for Low-Light Image Enhancement (arxiv.org)

Colab_link PRID-Net: PRID1.ipynb

(If links are not clickable plz copy and paste)