GNR 638

Mini-Project 2

Denoising of Images using Convolutional Neural Networks

Amruta Parulekar & Hemant Hajare

20d070009 20d070037

Indian Institute of Technology, Bombay

Contents:

1) Problem statement

1. Download sharp images
2. Downscale the images to (256,448). (Set A)
3. Create a set of images by applying different gaussian filters to each image: (Set B)
    a. Kernel size = 3x3 , sigma = 0.3
    b. Kernel size = 7x7 , sigma = 1
    c. Kernel size = 11x11 , sigma = 1.6
4. Design a network to deblur images (Set B -> Set A) Upper Limit is 15 M parameters.
5. Test set will be provided along with ground truth later. We will also providing an evaluation script. Please report PSNR score according to this score.
6. Please use numpy==1.24.4, PIL==10.2, scikit-image==0.21 for preprocessing and running evaluation script.

2) Data preprocessing

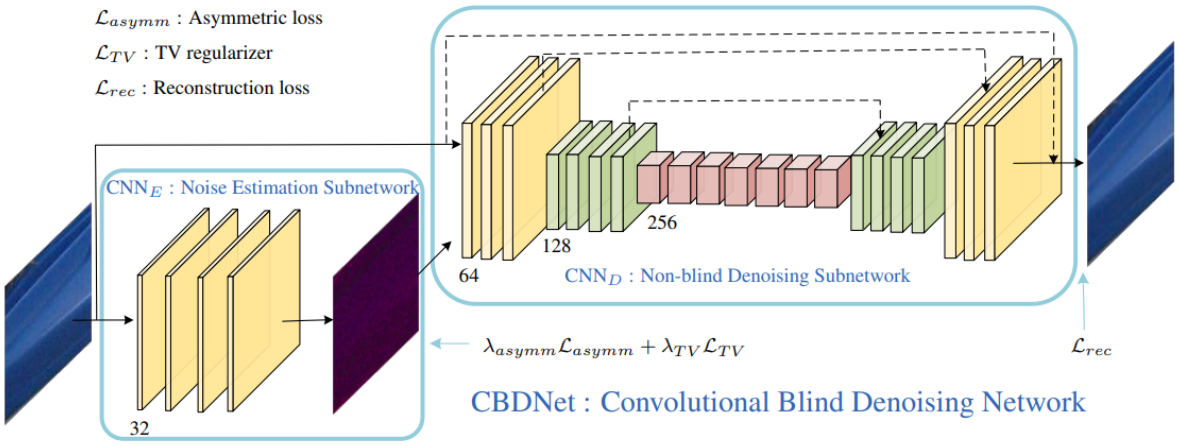
We began by downsizing the images to (256,448) pixels, forming Set A.
Then, we created Set B having 3N images by applying different Gaussian filters to each image:
a. 3x3 kernel size and sigma of 0.3.
b. 7x7 kernel size with sigma set to 1.
c. 11x11 kernel size and sigma of 1.6.
This yielded a varied dataset, ranging from mild to significant blurring, enabling versatile experimentation in image processing tasks.
A CSV file was made of corresponding image and target paths

3) Experimentation

a) Choosing a base model

Two existing architectures for denoising of images were studied. We finally chose CBDNet due to numerous possible modifications that could be made in its architecture. After picking an architecture suitable for the task, modifications were made in it .

1. CBDNet



Overall, this architecture aims to estimate noise in the input image and then remove that noise using a series of convolutional layers. It follows a typical encoder-decoder architecture commonly used in image denoising tasks, with skip connections to preserve spatial information.

a) NoiseEstimationSubnetwork:

- This subnetwork is responsible for estimating the noise in the input image.
- It takes a 3-channel input image (presumably RGB) and passes it through a series of convolutional layers with ReLU activations.
- The architecture consists of 5 convolutional layers (conv1 to conv5), where each layer has a kernel size of 3x3 and padding of 1 to preserve spatial dimensions.
- The final convolutional layer (conv5) outputs a 3-channel map representing the estimated noise.

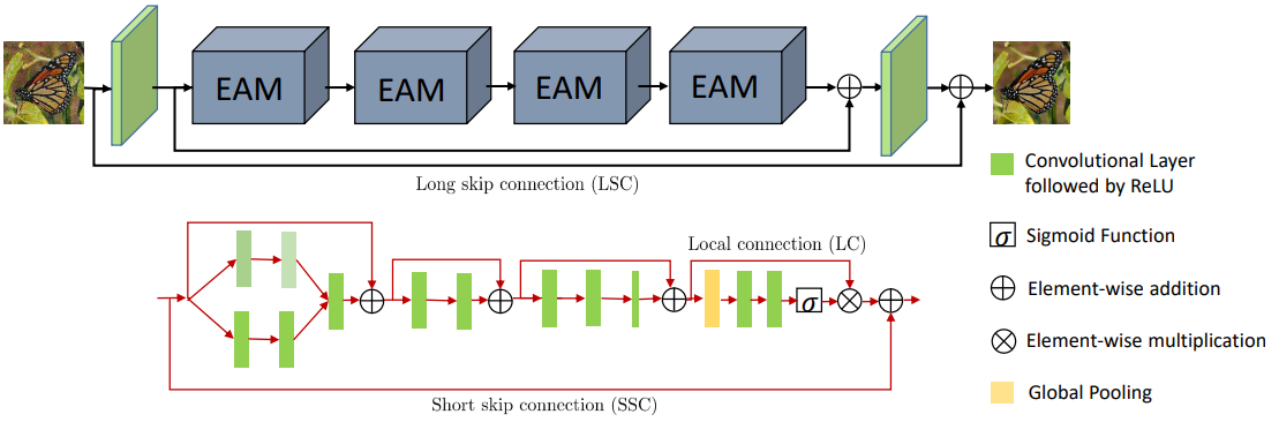
b) NonBlindDenoisingSubnetwork:

- This subnetwork aims to perform non-blind denoising, which means it removes noise from the input image using the estimated noise map.
- It starts with a series of convolutional layers (conv1 to conv5) with ReLU activations and a max-pooling layer (pool1) to downsample the feature maps.
- Then, it continues with more convolutional layers (conv6 to conv11) and another max-pooling layer (pool2).
- After that, it applies transposed convolutional layers (upsample1 and upsample2) to upsample the feature maps.
- The upsampled feature maps are then added back to feature maps from earlier layers (add1 and add2) before passing through more convolutional layers.
- Finally, the output is obtained through a 1x1 convolutional layer (out).

c) CBDNet (Combined Denoising and Deblurring Network):

- This is the main model that combines the noise estimation subnetwork and the non-blind denoising subnetwork.
- In the forward pass, it first estimates the noise map using the NoiseEstimationSubnetwork.
- Optionally, it concatenates the estimated noise map with the input image before passing it to the denoising subnetwork (commented out in the code).
- Then, it applies the non-blind denoising subnetwork to the input (optionally concatenated with the noise map) to obtain the denoised output.

2. RIDNet



Overall, RIDNet leverages the residual learning concept along with dense connections and attention mechanisms provided by the EAM module to effectively denoise images while preserving important details. It's a deep architecture capable of capturing both local and global features, making it suitable for challenging denoising tasks.

a) EAM (Enhanced Attention Module):

- EAM is a module designed to capture long-range dependencies and enhance feature representation.
- It consists of multiple convolutional layers with different dilation rates and a global average pooling layer.
- The module begins with two sets of dilated convolutional layers (conv1 and conv2) with increasing dilation rates to capture features at different receptive field sizes.
- The outputs of these dilated convolutions are concatenated and passed through another convolutional layer (conv5) to fuse the features.
- The fused features are then added to the input features (x) to create a residual connection (add1).
- Subsequently, several convolutional layers (conv6 to conv10) follow, enhancing the feature representation further.
- The module concludes with a global average pooling layer (gap) to capture global context, followed by a couple of convolutional layers (conv11 and conv12) to refine the features.

b) RIDNet:

- RIDNet is the main architecture that utilizes multiple instances of the EAM module to denoise images.
- It begins with an initial convolutional layer (conv1) to process the input image.
- This is followed by four instances of the EAM module (eam1 to eam4), each processing the features successively.
- After the final EAM module, there's a convolutional layer (conv2) that produces the denoised output.
- The final output is obtained by adding the denoised features (conv2) to the input features (x), forming a residual connection (out).

3. Why CBDNet over RIDNet?

CBDNet offers a simpler architecture than RIDNet, making it easier to modify and adapt for specific denoising tasks. With fewer layers and components, CBDNet is more straightforward to understand and implement, providing flexibility for experimentation and customization. Its explicit noise estimation subnetwork provides additional information about input noise characteristics, potentially enhancing denoising performance. Moreover, CBDNet's stability during training, attributed to its simplicity, ensures smoother convergence and better generalization, particularly beneficial when dealing with limited or noisy datasets. Overall, CBDNet's simplicity and ease of modification make it an advantageous choice for us as we seek flexibility in architecture design and adaptation to our denoising requirements.

b) Modifications of architecture

1. Modification in skip connections

- Additional skip connections were added to the network.
    - The output of conv12 was directly added to conv4 instead of being added to conv5.
    - The output of conv13 was added to conv12 instead of directly passing it to conv14.
    - The output of conv14 was directly added to conv2 instead of being added to conv3.

Benefits:

- The skip connections might help in preserving more detailed information, as these connections bypass fewer layers compared to the original design.
- Skip connections allow easier flow of gradient information
- The model may better capture low-level features, which could lead to better denoising performance, especially in preserving image details.
- These changes potentially simplify the flow of information and may help in training the model more efficiently, as there are fewer intermediate connections between layers.
- Overall, these changes aim to optimize the network architecture for better denoising performance and potentially faster convergence during training.
- However, the actual impact would depend on the specific dataset and denoising task. Hence, testing these changes is necessary to validate their effectiveness.

2. Modifications in max-pool layers

- Depth was Increased:
    - The number of convolutional layers was increased from 10 to 16.
    - An additional pooling layer was introduced, augmenting the depth of the network.
- Changes were Made in Pooling:
    - One more pooling layer (pool3) was added to the network.
- Up-sampling Layers were Added:
    - Another up-sampling layer (upsample3) was included in the network.

Benefits

- Increasing the depth of the network and adding more layers can potentially increase the model's capacity to capture complex patterns and features in the input images, which might result in improved denoising performance.
- Additional pooling layers can help in capturing hierarchical features at different levels of abstraction.
- The introduction of more up-sampling layers can enhance the network's ability to reconstruct the image while preserving spatial information, which is crucial for denoising tasks.
- These changes may lead to a more powerful and expressive model, capable of handling a wider range of denoising scenarios and producing higher-quality denoised images.
- However, it's essential to note that increasing the depth and complexity of the network also increases the computational cost and the risk of overfitting, especially if the dataset is not sufficiently large or diverse. Therefore, proper training and validation was necessary to assess the actual benefits of these changes.

3. Modifications in kernel sizes

- Adjustments in Convolutional Layers:
    - The kernel size for conv1 and conv2 was increased to 7x7 with padding of 3, which resulted in a larger receptive field for the initial layers.
    - Similar changes were applied to conv12, which was modified to have a kernel size of 5x5 with padding of 2.
    - The kernel size for the remaining convolutional layers remained unchanged.

Benefits:

- By increasing the kernel size of certain convolutional layers, the network can capture larger spatial features, potentially enhancing its ability to denoise images effectively.
- Overall, these modifications are expected to enhance the performance and robustness of the denoising network, enabling it to handle a wider range of input images and produce more accurate denoised results.
- However, it's crucial to conduct thorough testing and validation to assess the actual benefits of these changes and ensure that the network performs optimally on the task.

c) Model specification choices

1. Choosing a learning rate

The learning rate is a critical hyperparameter in training machine learning models. It determines the size of steps taken during optimization to update the model's parameters. Setting it too high might lead to unstable training or overshooting the optimal solution, while setting it too low could result in slow convergence or getting trapped in local minima. Finding the right balance for the learning rate is essential for achieving optimal model performance and convergence during the training process.

We experimented with learning rates of 0.005,0.001 and 0.01 for our final model architecture.

2. Choosing amount of training data

The optimal amount of training data can vary depending on factors such as the complexity of the problem, the quality of the data, and the model architecture being used. In some cases, having too much irrelevant or noisy data can negatively impact performance, while in others, having insufficient data might lead to overfitting. Therefore, it's essential to strike a balance and ensure that the training dataset is sufficient to capture the underlying patterns of the problem without introducing unnecessary complexity.

We experimented with training data of 5000 and 10000 images on our final model architecture, due to computation constraints.(train-val split of 0.8-0.2)

3. Choosing the optimizer and loss function

We employed the Adam optimizer with its adaptive learning rate for training the image denoising model, ensuring efficient optimization in the high-dimensional parameter space. This was paired with a suitable loss function like Mean Squared Error (MSE) to guide the denoising process effectively. We fine-tuned hyperparameters and monitored performance on validation data to optimize model generalization and denoising quality.

3. Choosing a batch size

Larger batch sizes leverage parallel processing for faster computation but may require more memory and could lead to overfitting with smaller datasets. Smaller batch sizes might generalize better but converge slower. We used a fixed batch size of 32 for our experiments.

4)Results

a) Architecture experiments

The chosen base model CBDNet had architecture modifications made to it and PSNR was calculated on the provided test set. The PSNR of the original blurred images with the sharp images was 26.68. Higher PSNR indicated less noisy image.

Model architecture details	PSNR using script and test-set provided
Base CBDNet	27.83
Adding more skip connections	26.57
Removing skip connections	12.21
Increasing depth, adding pooling and upsampling layers	28.42
Increasing kernel sizes	27.22

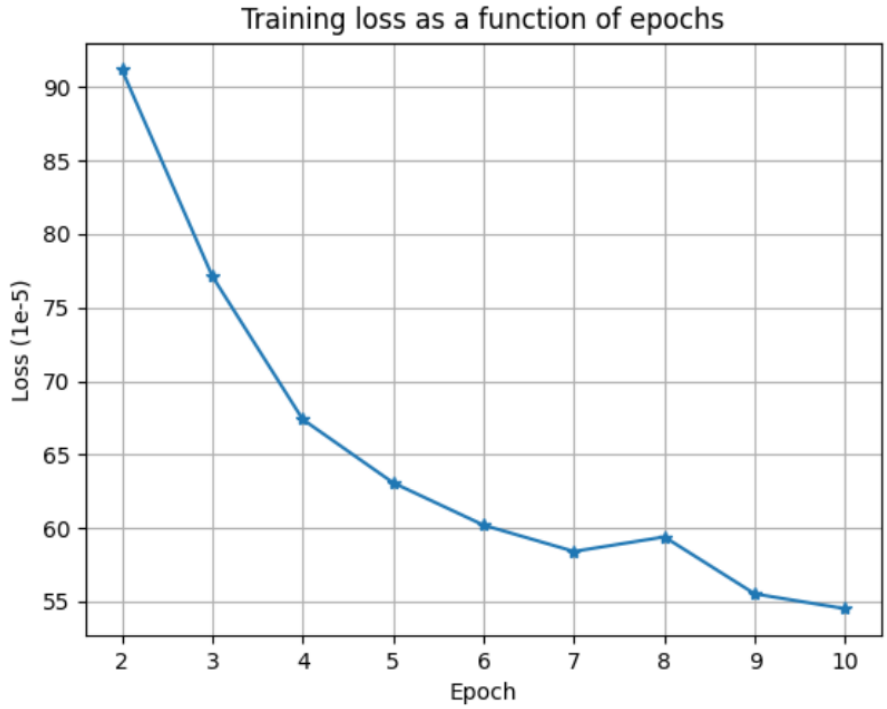
b)Hyperparameter tuning experiments

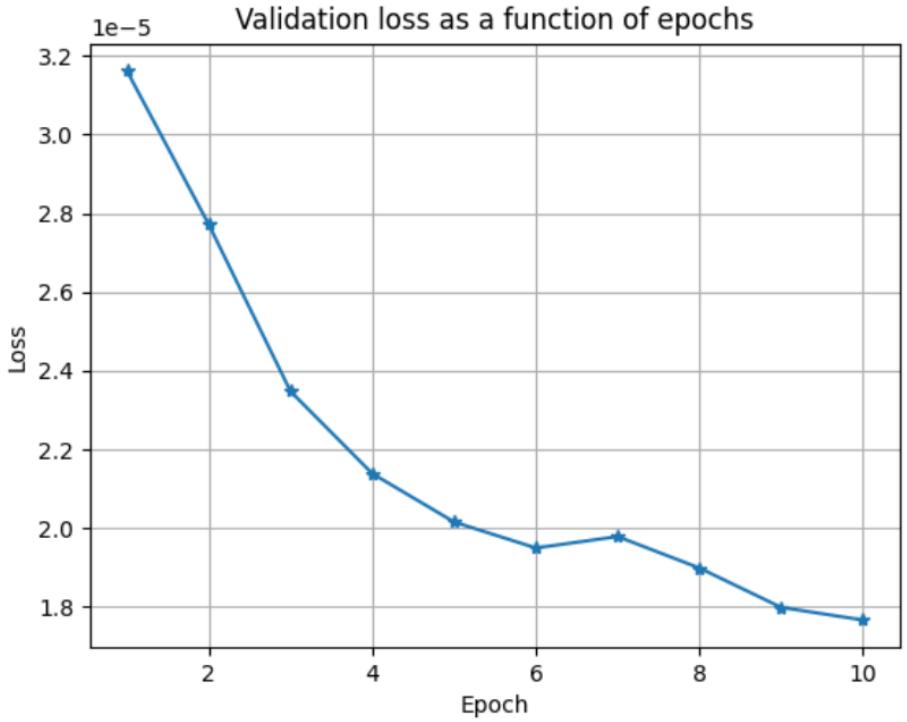
A train-val split of 0.8-0.2 was used on the model after increasing depth, adding pooling and upsampling layers. Training details were varied as follows and PSNR was calculated on the provided test-set

Model training details	PSNR using script and test-set provided
0.005 LR, 10 epochs, 5000 images	28.42
0.01 LR, 5 epochs, 5000 images	26.06
0.001 LR, 20 epochs, 5000 images	28.08
0.005 LR, 10 epochs, 10000 images	

b)Training curves

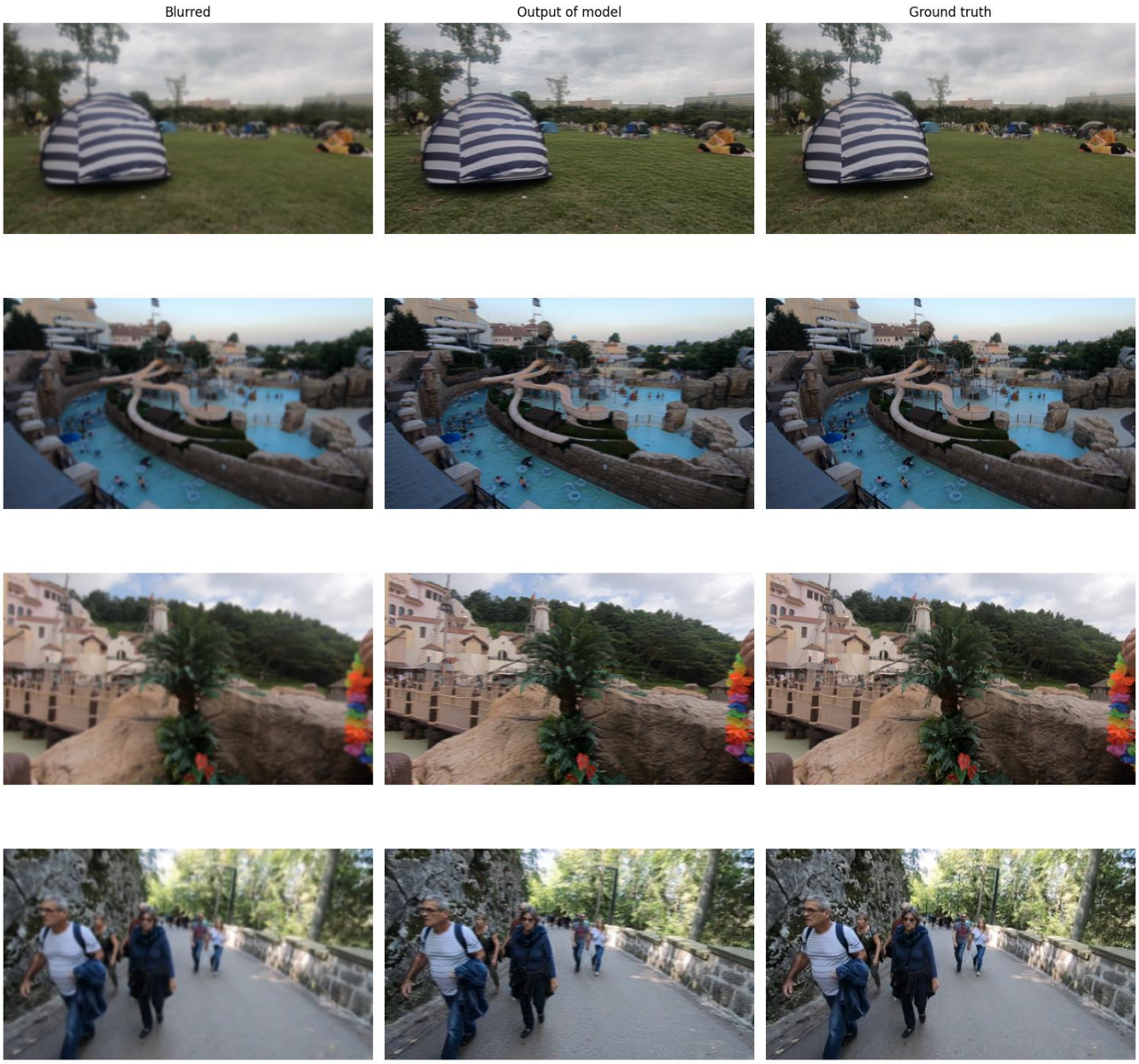
Training and validation losses for the best model were plotted as a function of epochs





c)Qualitative results

The model sharpened the blurred images and brought them closer to the target



5)Conclusion-best results

Best results were obtained after architectural manipulation of the CBDNet reference model, like increasing depth, and adding pooling and upsampling layers. The optimal training details were 0.005 learning rate, 10 epochs and 5000 training samples with a 0.8-0.2 train–val split. The PSNR obtained between translated images and sharp images of the provided test set by this model was 28.42