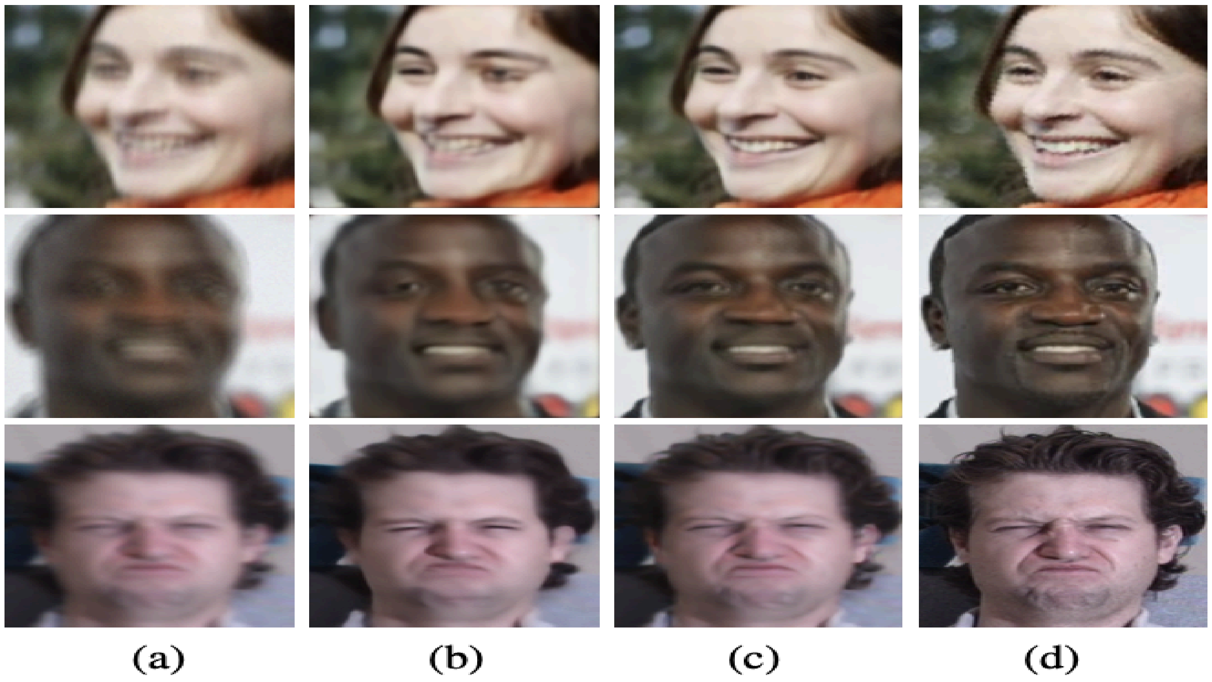
GNR638 Machine Learning for Remote Sensing II

Mini Project 2



Group Members:

- Aryan Adinath Popalghat (210020088)
- Divyam Gupta (210020044)
- Durgesh Sahane (210110096)

April 10, 2024

Indian Institute of Technology, Bombay

INDEX

1. Introduction

We are solving Problem 1 (Image Deblurring)
Image deblurring is a critical task in computer vision, essential for various applications such as surveillance, medical imaging, and photography. In this project, we tackle the problem of image deblurring using a modified U-Net architecture. The objective is to develop a deep learning model capable of effectively restoring sharp details from blurry images caused by Gaussian blur with varying kernel sizes and sigmas.

2. Data Preprocessing

The success of any machine learning model heavily relies on the quality and suitability of the dataset used for training. In this section, we outline the preprocessing steps applied to the dataset to ensure its compatibility with our task and to enhance the performance and robustness of our model.

1. Dataset Overview & Collection:

The given images are of the dimension 1280 x 720
So we have downscaled it to 256 x 256
(Note: Code for this downscaling is not added in all the files. Because we first downscaled the images and then did the further processing by writing code in a new file. At some places we have kept it but it is not there in all the files.)

2. Generating Blurred Images:

Blurred images were generated by applying Gaussian filters with different kernel sizes and sigmas to each sharp image.

The list of Gaussian filters applied is as follows:

1. Kernel size: (3, 3), Sigma: 0.3
2. Kernel size: (7, 7), Sigma: 1
3. Kernel size: (11, 11), Sigma: 1.6

3. Data Splitting:

We partitioned the dataset into training, validation. The split is 80% training and 20% testing (validation). The training set is used to train the model, the validation set is used to tune hyperparameters and monitor model performance during training.
And then we stored our model. We then tested our model on the evaluation set which is provided to us to gauge the performance of the model.

4. Data loading and Batching:

We utilized PyTorch's data loading utilities to efficiently load and preprocess the dataset. Data loaders were employed to generate mini-batches of data during training, allowing for parallel processing and memory optimization.

3. Model

We tailored the U-Net architecture to meet the parameter constraint while retaining its efficacy in capturing both local and global image features. The modified U-Net architecture comprises:

1. **Contracting path:** Comprising convolutional blocks with max-pooling layers to extract hierarchical features efficiently.
2. **Expanding Path:** Utilizing upsampling layers and skip connections to recover spatial information and fuse low-level features with high-level ones effectively.

3. **Final Convolution Layer:** To map the extracted features to the desired output space.

Architecture + Parameters:

```
----------------------------------------------------------------
        Layer (type)               Output Shape         Param #
================================================================
            Conv2d-1        [-1, 32, 256, 256]           1,760
            Conv2d-2        [-1, 32, 256, 256]           9,248
         MaxPool2d-3        [-1, 32, 128, 128]               0
            Conv2d-4          [-1, 64, 64, 64]          73,792
            Conv2d-5          [-1, 64, 64, 64]          36,928
         MaxPool2d-6          [-1, 64, 32, 32]               0
            Conv2d-7         [-1, 128, 16, 16]         295,040
            Conv2d-8         [-1, 128, 16, 16]         147,584
         MaxPool2d-9           [-1, 128, 8, 8]               0
           Conv2d-10           [-1, 256, 4, 4]       1,179,904
           Conv2d-11           [-1, 256, 4, 4]         590,080
        MaxPool2d-12           [-1, 256, 2, 2]               0
           Conv2d-13           [-1, 512, 1, 1]       4,719,104
           Conv2d-14           [-1, 512, 1, 1]       2,359,808
           Conv2d-15          [-1, 1024, 1, 1]       4,719,616
  ConvTranspose2d-16           [-1, 256, 4, 4]         262,400
           Conv2d-17           [-1, 256, 4, 4]         590,080
           Conv2d-18           [-1, 256, 4, 4]       1,179,904
           Conv2d-19           [-1, 512, 4, 4]       1,180,160
  ConvTranspose2d-20         [-1, 128, 16, 16]          65,664
           Conv2d-21         [-1, 128, 16, 16]         147,584
           Conv2d-22         [-1, 128, 16, 16]         295,040
           Conv2d-23         [-1, 256, 16, 16]         295,168
  ConvTranspose2d-24          [-1, 64, 64, 64]          16,448
           Conv2d-25          [-1, 64, 64, 64]          36,928
           Conv2d-26          [-1, 64, 64, 64]          73,792
           Conv2d-27         [-1, 128, 64, 64]          73,856
  ConvTranspose2d-28        [-1, 32, 256, 256]           4,128
```

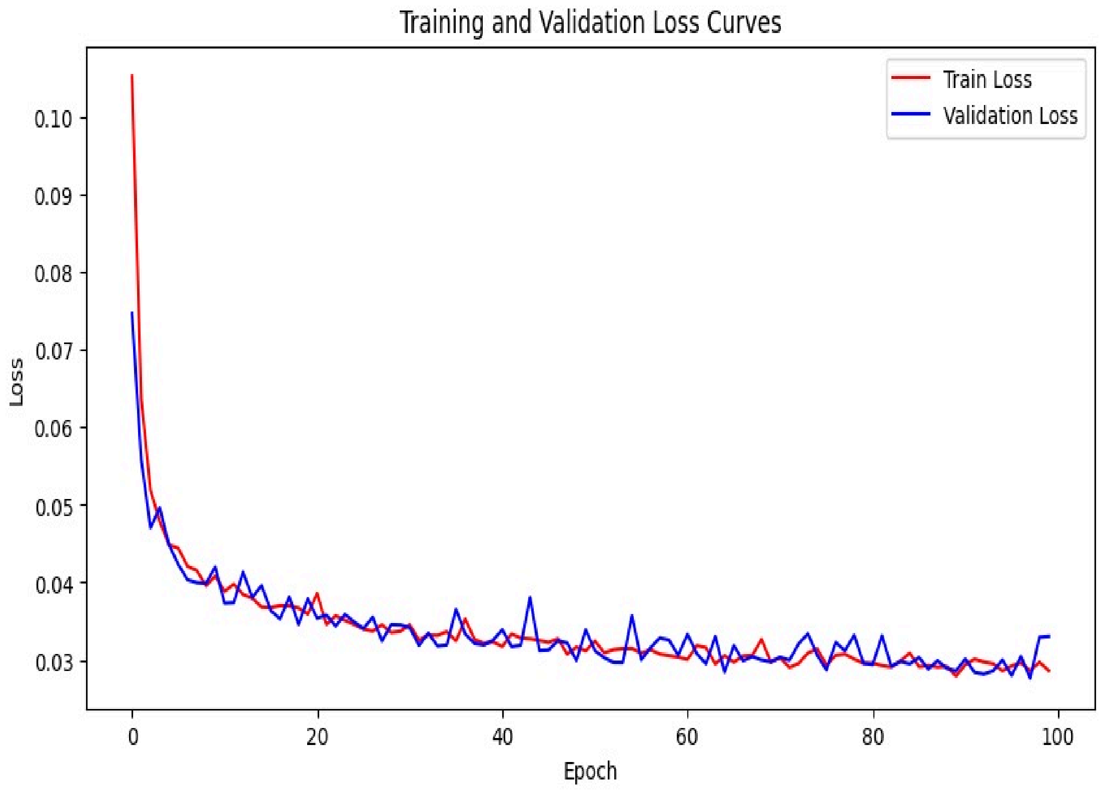
```
          Conv2d-29         [-1, 32, 256, 256]           9,248
          Conv2d-30         [-1, 32, 256, 256]          18,464
          Conv2d-31         [-1, 64, 256, 256]          18,496
          Conv2d-32          [-1, 3, 256, 256]             195
================================================================
Total params: 18,400,419
Trainable params: 18,400,419
Non-trainable params: 0
----------------------------------------------------------------
Input size (MB): 1.50
Forward/backward pass size (MB): 134.05
Params size (MB): 70.19
Estimated Total Size (MB): 205.75
----------------------------------------------------------------
```

4. Training

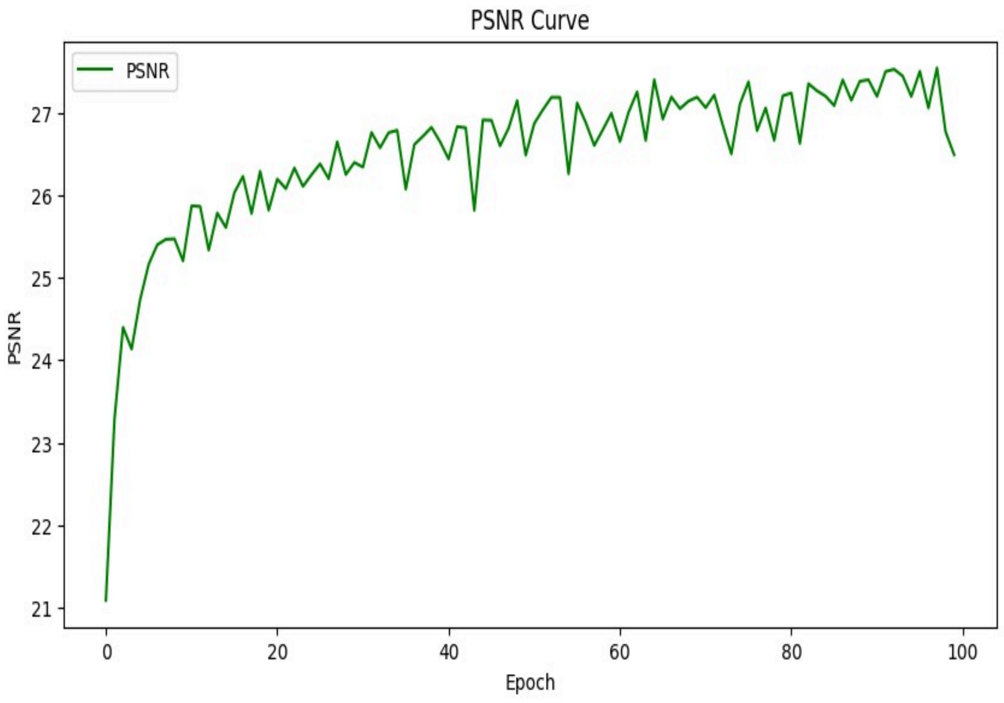
The modified U-Net model was trained using the sharp images (Set A) and their corresponding blurred counterparts (Set B). The training details are as follows:

- **Loss Function:** Mean Squared Error (MSE) loss function to measure the disparity between predicted and ground truth images.
- **Optimizer: AdamW** (It is a variant of the Adam optimizer, which stands for Adaptive Moment Estimation. The "W" in AdamW stands for "weight decay," which is a form of regularization.)
- **Learning rate:** 0.01
- **Batch size:** 2
- **Number of Epochs:** 100

1. Training Loss and Validation Loss Curve:



2. PSNR Curve:



Results

1. **Maximum PSNR: `27.53953346217939`**
2. **Minimum Train Loss: `0.02792118303477764`**
3. **Minimum Validation Loss: `0.027643241609136265`**

5. Evaluation

First we downscaled the blur images in the evaluation set provided to 256 x 256 (and stored them in the 'blur1' folder.)
We passed these blur images in our model to get sharp images (and stored them in the 'created_sharp' folder.)

And using the eval script which is provided to us we compared the images:

1. As instructed in the script we compared our 'created_sharp' images with their blur images (which were downscaled and stored in the 'blur1' folder.) And we got the following result:
   len(psnr_values): 150
   Average PSNR between corresponding images: `26.638761075846357 dB`
   (We have submitted this eval script)
2. We have also compared our 'created_sharp' images with their original 'sharp' images and we got the following result:
   Average PSNR between corresponding images:
   `26.29147169079316 dB`
   (We haven't submitted this file because we have to submit an evaluation file according to their script only. But to compare 'created_sharp' with 'sharp' we first downscaled their sharp images to 256 x 256 and then stored them in the 'sharp1' folder. And in evaluation file we just replace the path of 'blur1' by 'sharp1')

6. Qualitative Analysis:

The obtained average PSNR value of approximately 26.64 dB between the deblurred images and the original blurred images indicates a significant improvement in image quality.

- PSNR measures the fidelity of the deblurred images to the original blurred images.
- A higher PSNR value (obtained in point 2 of Evaluation section) suggests that the deblurred images closely resemble the original sharp images, with minimal distortion or loss of information.
- In our case, achieving an average PSNR of 26.64 dB indicates a substantial reduction in blurring artifacts and an effective restoration of image sharpness.

Conclusion: The high PSNR value demonstrates the capability of our deblurring model to accurately reconstruct sharp images from their blurred counterparts. This suggests that the model effectively captures and restores the lost details during the blurring process, resulting in visually pleasing and faithful representations of the original scene.

Submitted Files:

1. Model file: Final_Model.ipynb
2. To generate ‘created_sharp’ images: test_output.py
3. Evaluation: eval.py
4. Checkpoint: new_data.pt - Due to large size we can’t submit this file. So here we are providing a link for the checkpoint file. Click here to get the checkpoints.
   OR paste this link in your browser - https://drive.google.com/file/d/1Q79i_VJgoe49JKqBNzB7ZTVMcO2ew5L3/view?usp=sharing