GNR 638

Machine Learning for Remote Sensing – II

Mini Project 2

Report

Shubham Raj (22B0665)

Kartik Singh (22B0692)

Arnav (22B0664)

GNR 638 Mini Project 2

Image Deblurring

1. Download sharp images from here: Google Drive Link
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) with an upper limit of 15M parameters.
5. Test set will be provided along with ground truth later. We will also provide 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.
7. You need to submit:
   - Codes
   - Final checkpoint
   - Report (contains model architecture, training details, training curves, qualitative and quantitative results)

Link to Saved Checkpoint and GitHub Repo

Link to GitHub Repository here
Link to Report and Checkpoint here

How To Use?

- Download the train dataset from here
- Download the eval dataset and eval script from here
- Clone/Download the Repo `git clone https://github.com/shubham282raj/gnr638-project-2.git`
- Extract both train and test dataset in `dataset` folder in root directory
- Run `main.ipynb`, this will
  - Process the dataset and make blur and sharp dataset
  - Create PyTorch dataloader
  - Define the Model (UNet)

- Train the model
- Plot Train/Val Losses averaged over few 100 batches

- To Evaluate on eval dataset and eval script, run `eval.ipynb`, this will
  - Go through test dataset and make it's prediction making blurry images sharp
  - Run the provided script in `utils` to print the **PSNR Score**

Architecture Used: UNet

- The UNet Model structure was first motivated from a GitHub Repository from **milesial** where they used UNet and implemented it for **Image Segmentation** in `PyTorch` with over 30 Million Parameters
- Link to there GitHub Repo here
- We used just the model architecture for our project
- There UNet Model had
  - 4 Encoders
  - 4 Decoders
  - Over 30 Million Parameters
- Changes we made in out UNet Model
  - 3 Encoders for 7.7M params and 2 Encoders for 1.9M params
  - 3 Encoders for 7.7M params and 2 Encoders for 1.9M params
  - Skip connections between same horizontal layers (UNet Architecture)
  - 7.7 Million Parameters

Device Specification

There device used to train and evaluate the model had

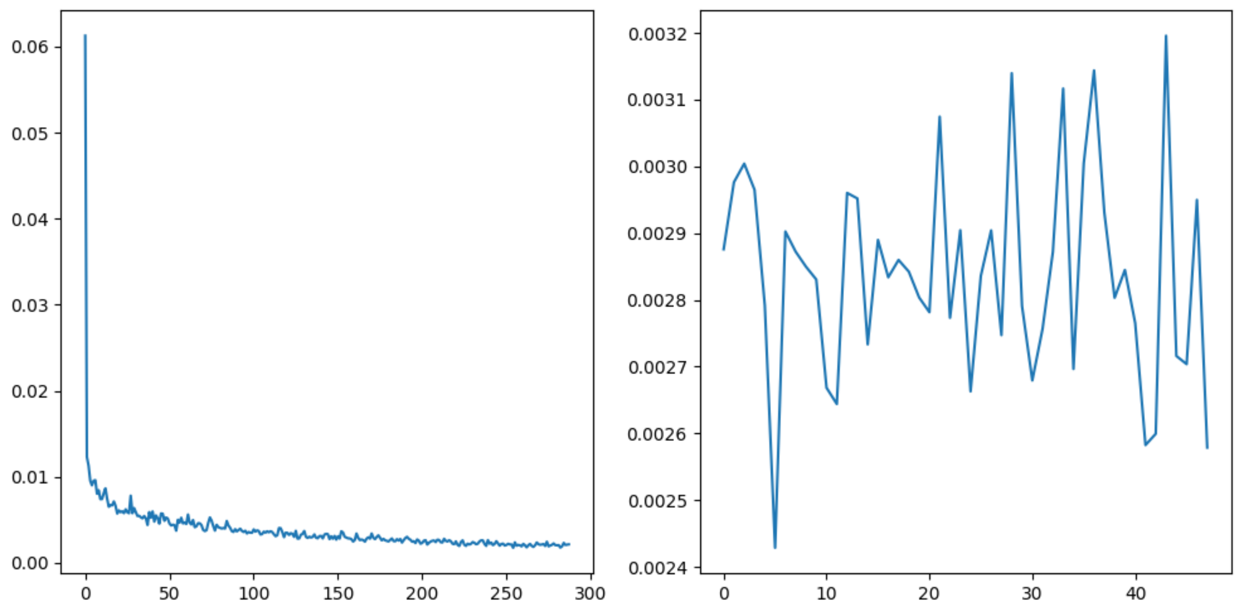
- AMD Ryzen 7 5800H CPU
- NVIDIA GeForce RTX 3050 Laptop GPU
- 8 GB 3200 MHz RAM

Training Details

- 80/20 Train Val Split
- Total Number of (blur, sharp) image pair = 72,000
- Total Number of parameters in UNet Model = 7.7M
- Batch Size = 2
- Epochs Trained = 2
- Criterion Used - MSE Loss
- Optimizer Used - Adam (lr=0.001)
- Total Time Taken = 3.6 Hours
- Train Iter Batches per second = 4.3 it/s (For 7M params)
- Val Iter Batches per second = 11.0 it/s (For 7M params)
- Moving average was taken over per 100 batches in an epoch

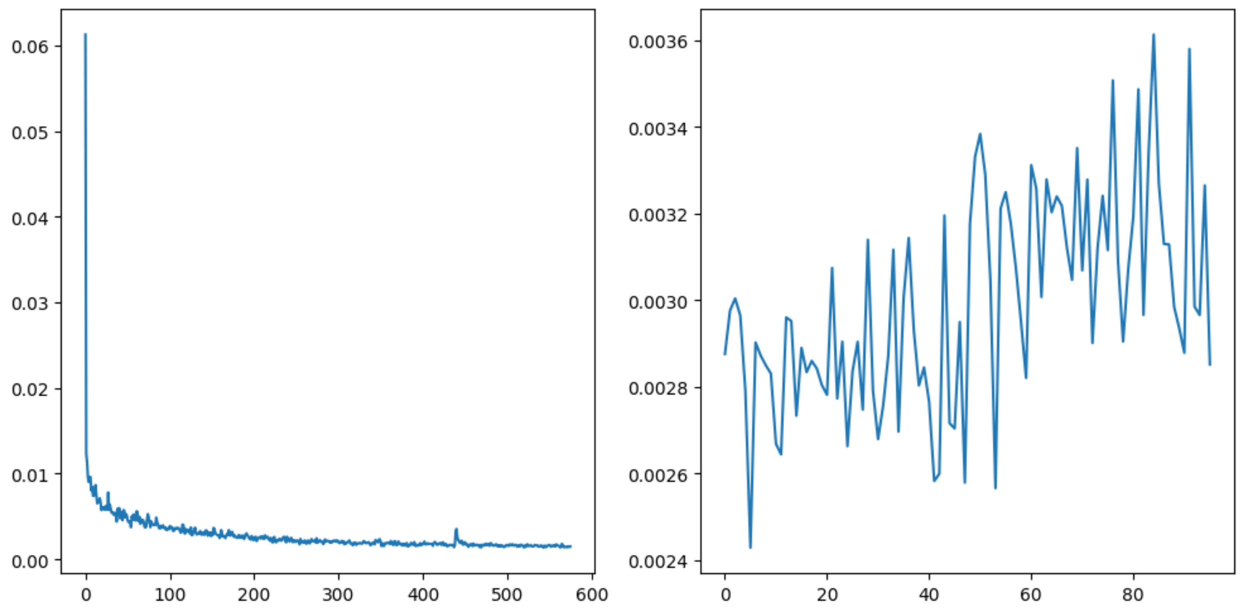
Train/Val Losses Curve

UNet Model with 7.7 Million Parameters (1 Epoch)



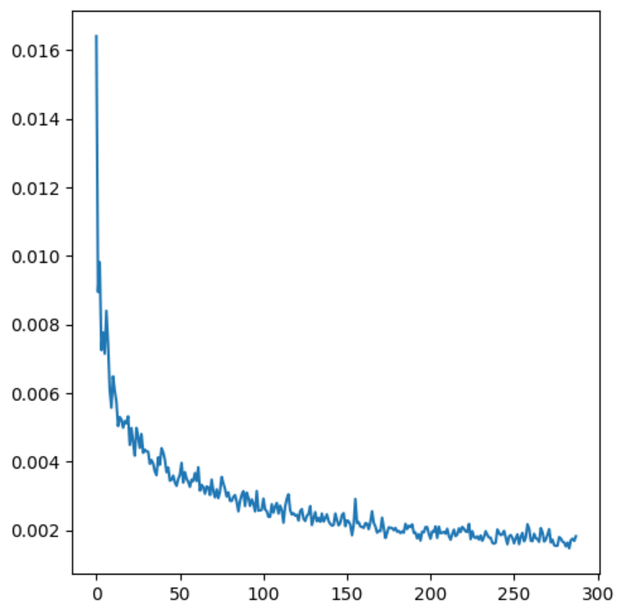
Average PSNR between corresponding images: 25.353461741615302 dB

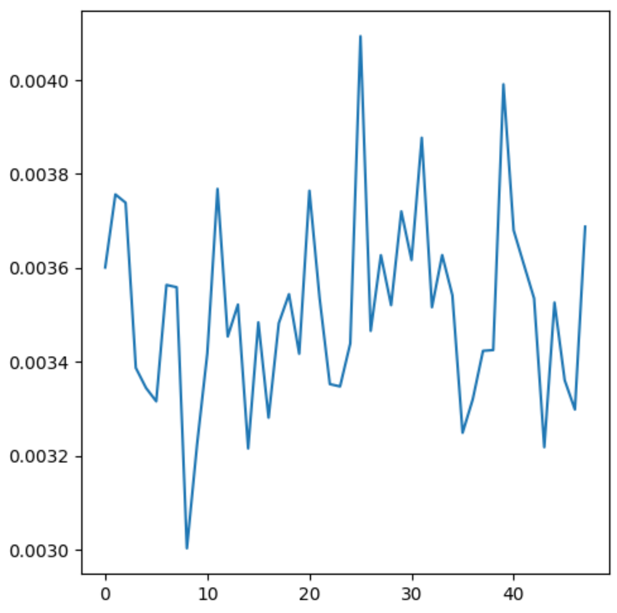
UNet Model with 7.7 Million Parameters (2 Epochs)



Average PSNR between corresponding images: 27.556361211650493 dB

UNet Model with 1.86 Million Parameters (1 Epoch)





Average PSNR between corresponding images: 26.815195069990057 dB

Summary

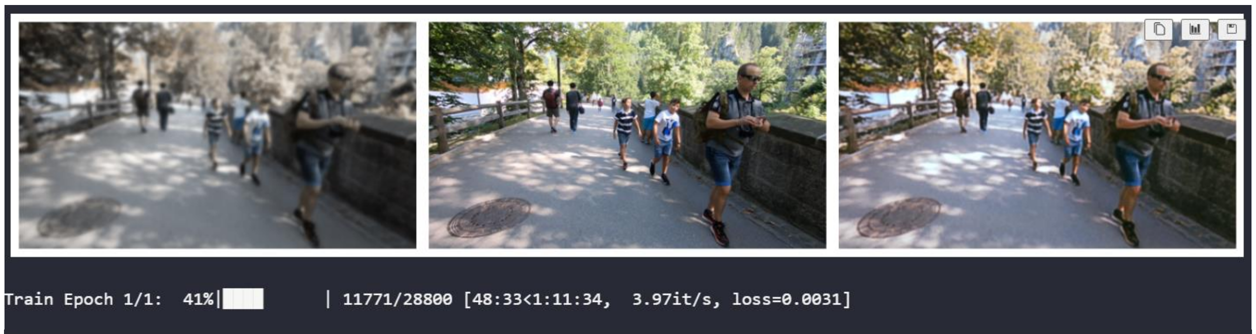
- Total Parameters: 7.7M
- Train Time: 3.6 Hours
- UNet with 7.7M parameters
  - Train Time – 1.8 Hours for 1 epochs
  - Average PSNR between corresponding images: 25.353461741615302 dB
- UNet with 7.7M parameters
  - Train Time – 3.6 Hours for 2 epochs
  - Average PSNR between corresponding images: 27.556361211650493 dB
- UNet with 1.86M parameters
  - Train Time – 1.48 Hours for 1 epoch
  - Average PSNR between corresponding images: 26.815195069990057 dB

Results

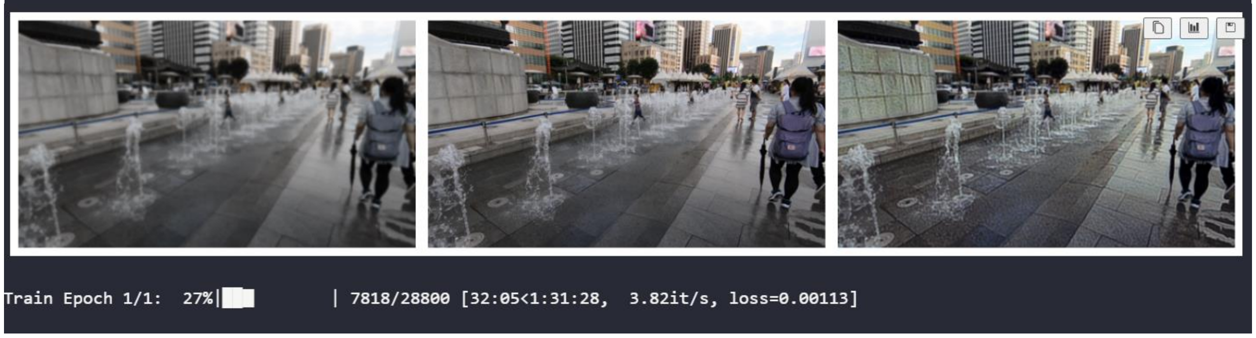
Although UNet with 7.7M parameters seems to perform better in terms of PSNR score, but taking time taken to train the model for 2 epochs and no much significant difference in PSNR scores, it seems one should go with UNet Model with 1.86 Million Parameters that performed pretty descent with such low number of parameters and just 1.5 hours of training. On the other hand the model with 7.7M parameters for 1 epoch seems to performs not that great although it took 1.8 hours.

Some Examples of Deblurring (7.7 Million Param with 2 Epochs)

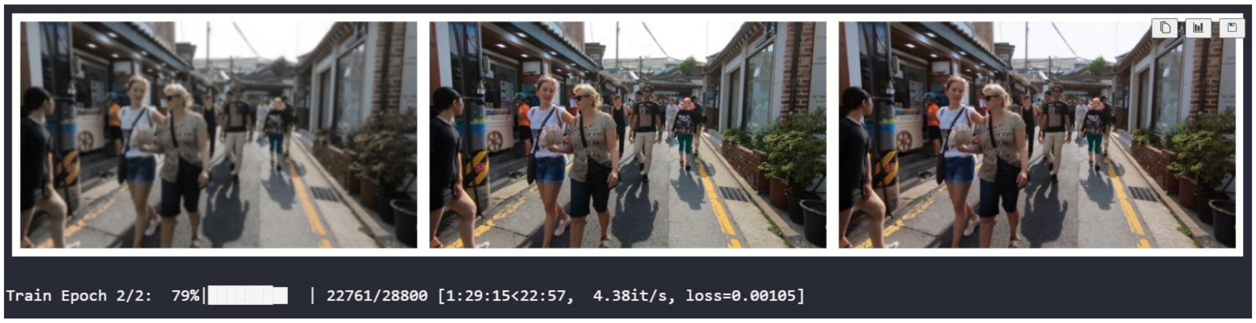
Blur Image Sharp Image Model Output Image



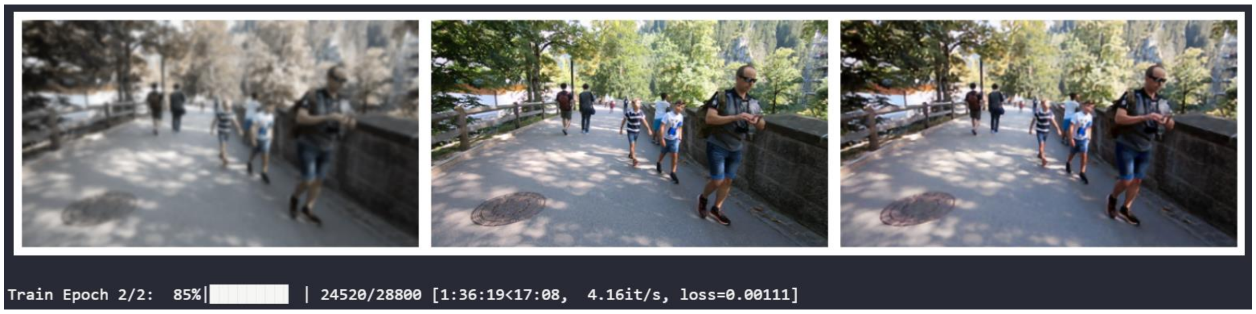
```
Train Epoch 1/1:  41%|████      | 11771/28800 [48:33<1:11:34,  3.97it/s, loss=0.0031]
```



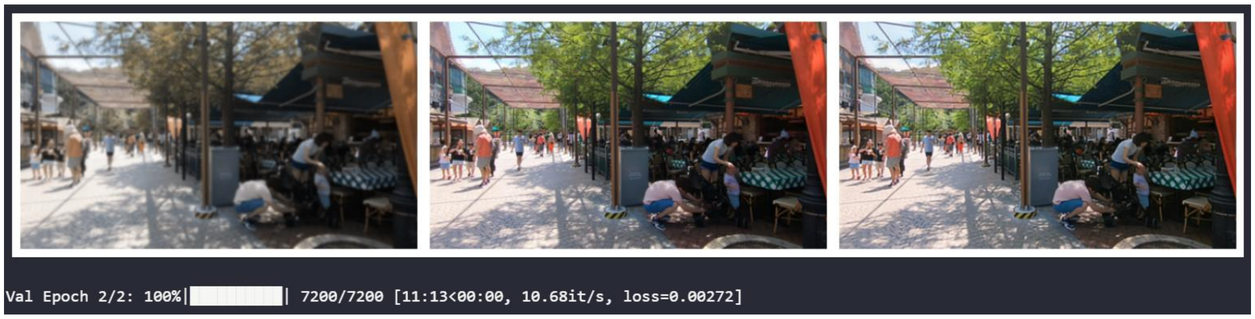
```
Train Epoch 1/1:  27%|██▋       | 7818/28800 [32:05<1:31:28,  3.82it/s, loss=0.00113]
```



```
Train Epoch 2/2:  79%|███████▉  | 22761/28800 [1:29:15<22:57,  4.38it/s, loss=0.00105]
```



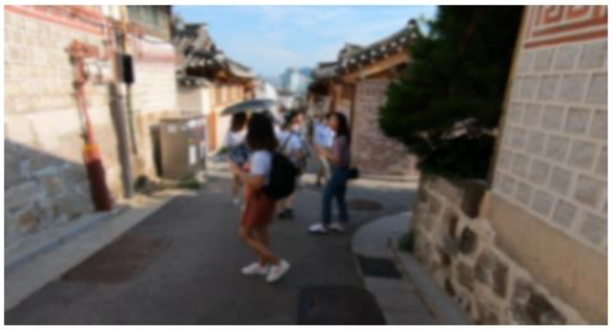
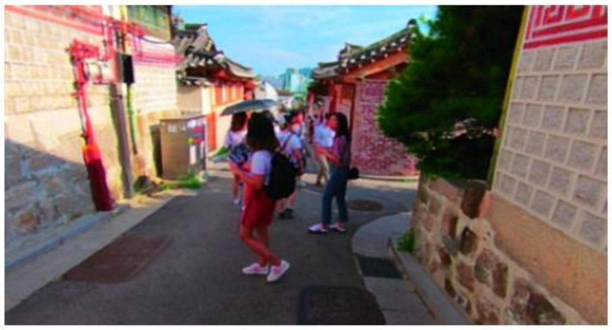
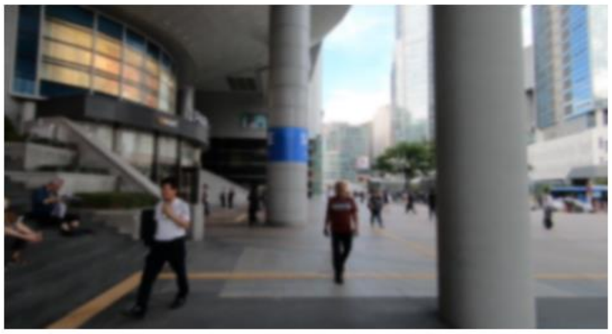
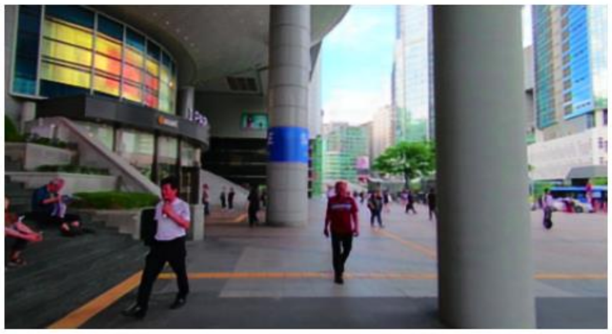
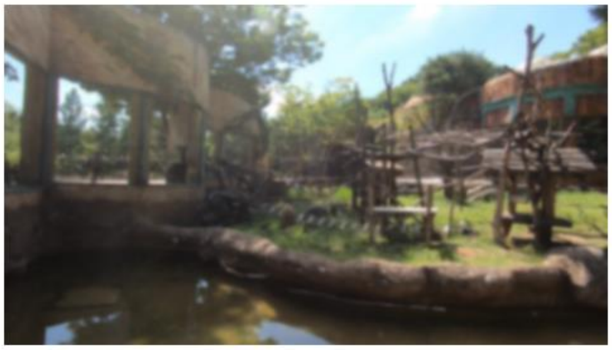
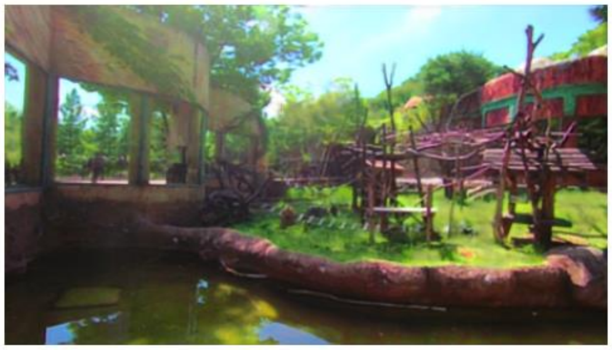
```
Train Epoch 2/2:  85%|████████▌ | 24520/28800 [1:36:19<17:08,  4.16it/s, loss=0.00111]
```



```
Val Epoch 2/2: 100%|██████████| 7200/7200 [11:13<00:00, 10.68it/s, loss=0.00272]
```

Image Results on Test set provided(7.7 Million Parameters with 2 Epochs)

Blur Image Model's Output

Reason For Moving Average in Plotting Graph

As discussed over webmail, TA asked to take average over thousand points, and it worked best over 200 images for us.

Here is the email from the TA,

Hey,

You can take an average loss over thousand images and get 72 points per epoch. Curve with more points heps reader to understand stability of your network.

Second : Try decreasing the parameters and if you find decrease in performance of the model mention that in your report.

Important Note : Write this in your report in last page that you discussed with me. So, that it doesn't effect the evaluation.

Best wishes,

Sai.