GNR 638: MiniProject-2

Image Deblurring

Group Members

1. Ayush Patil 200070012
2. Margav Savsani 200050072
3. Sartaj Islam 200050128

Faculty Guide

1. Prof. Biplab Banerjee

Contents

1 Introduction

Have you ever taken a photo only to be disappointed that it turned out blurry? Unfortunately, cameras aren't perfect, and sometimes things like shaking hands or fast-moving objects interfere with shots resulting in blurry photos. That's where image deblurring comes in handy! Consider it as digital detectives working to undo any damage done by blur and restore sharper, clearer photos so that viewers can appreciate all their details.

Figure 1: Image Deblurring

An unclear photo can be caused by various things. Like solving a puzzle, image deblurring uses advanced algorithms to try and find the most likely path back to an original crisp image.

2 Model Architecture

We have used a Convolutional Neural Network (CNN) architecture to process the inputs.

```python
class CNNModel(nn.Module):
    def __init__(self):
        super(CNNModel, self).__init__()

        self.conv_layers = nn.Sequential(
            nn.Conv2d(3, 128, kernel_size=5, padding=1),
            nn.ReLU(),
            nn.Conv2d(128, 64, kernel_size=3, padding=1),
            nn.ReLU(),
            nn.Conv2d(64, 3, kernel_size=1, padding=1),
            nn.ReLU()
        )

    def forward(self, x):
        x = self.conv_layers(x)
        return x
```

Figure 2: Model Class

```
----------------------------------------------------------------
        Layer (type)               Output Shape         Param #
================================================================
            Conv2d-1        [-1, 128, 254, 446]           9,728
              ReLU-2        [-1, 128, 254, 446]               0
            Conv2d-3         [-1, 64, 254, 446]          73,792
              ReLU-4         [-1, 64, 254, 446]               0
            Conv2d-5          [-1, 3, 256, 448]             195
              ReLU-6          [-1, 3, 256, 448]               0
================================================================
Total params: 83,715
Trainable params: 83,715
Non-trainable params: 0
----------------------------------------------------------------
```

Figure 3: Model Summary

Following is the detailed structure of the model:

- **Input:** The model expects color images with 3 channels representing RGB.
- **Convolutional Layers:**
  - **Layer 1:** 128 filters with a kernel size of 5x5 are applied with padding to maintain the spatial dimensions of the input. ReLU activation introduces non-linearity.
  - **Layer 2:** 64 filters with a kernel size of 3x3 are applied with padding, further extracting features. ReLU activation is used again.
  - **Layer 3:** 3 filters with a 1x1 kernel size are applied, likely serving to map the features to the desired output dimensions. ReLU activation is employed.
- **Output:** The final output is likely the same spatial dimension as the input image. Since the last layer has 3 channels, it reinforces the assumption that the network aims to reconstruct a deblurred version of the input.
- **Dataset:** This is the same as was provided in the problem statement
- **Parameters:** The no. of parameters of the model was around **83K** which is significantly less than the specified limit of 15M parameters.

3 Training details

- **Training Methodology**: The model was trained from scratch for 19 epochs, indicating that the model's weights were initialized randomly and subsequently optimized throughout the training process.
- **Optimizer:** The Adam optimizer was used, a common adaptive learning rate algorithm known for its efficiency and stability in image-related tasks. The initial learning rate was set to 0.01.

- **Loss Function:** Mean Squared Error (MSE) loss was used to measure the difference between the model's deblurred output and the ground truth sharp images. MSE is a standard choice for image restoration tasks.
- **Model Checkpointing:** Model checkpoints were saved after every epoch, allowing for the potential resumption of training, the selection of the best-performing model, or further analysis.

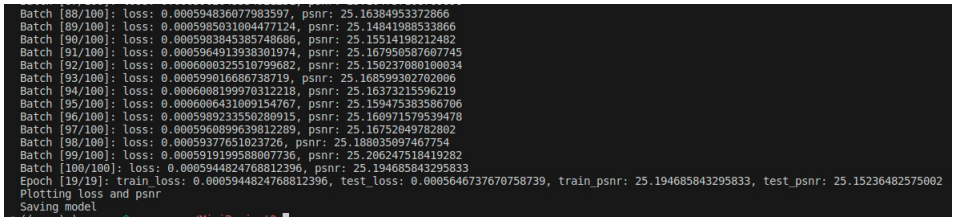

Figure 4: Training Logs

4 Training Curves

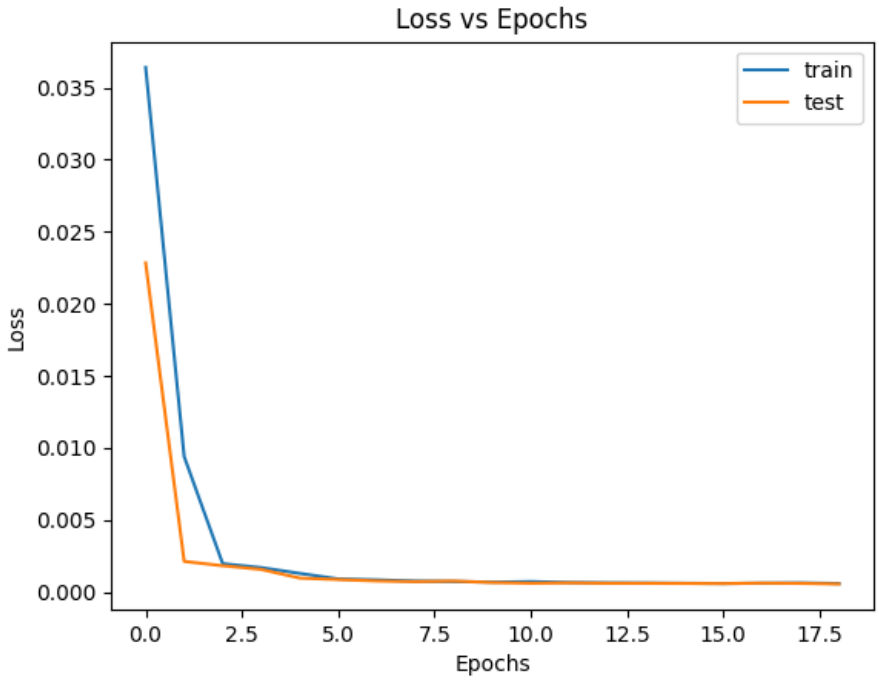

Figure 5: Loss vs Epochs

The training process resulted in a significant reduction of MSE loss from 0.0252 to 0.0005. This demonstrates the model's ability to learn and improve its deblurring predictions. It is visible that the train and test curves are almost identical and hence the model doesn't suffer from overfitting.

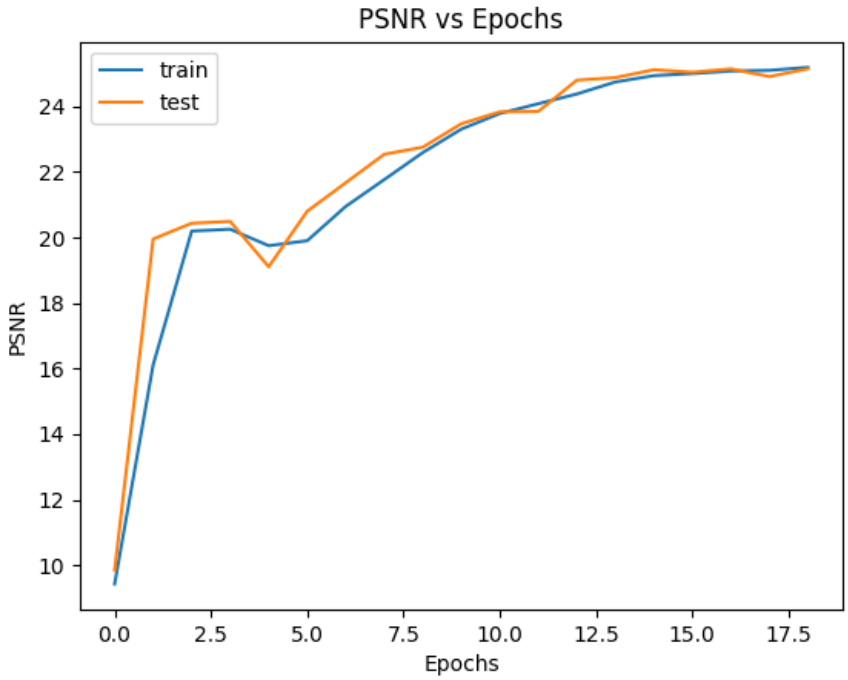

Figure 6: PSNR vs Epochs

A test **PSNR** of **25.15** was achieved. PSNR (Peak Signal-to-Noise Ratio) is a standard metric for evaluating image reconstruction quality, with higher values indicating better results.

5 Results

- **Initial Architectures:** We investigated encoder-decoder models (employing ConvTranspose2d layers), SRCNN, and UNET-like structures. These models achieved limited success, yielding a PSNR of around 10 and noticeable degradation in the deblurred images.
- **Transfer Learning Exploration:** Leveraging a ResNet-16-based encoder-decoder model for transfer learning resulted in a similar performance profile.

- **Breakthrough with Restormer:** The Restormer model, with approximately 7 million parameters, provided a substantial improvement, achieving a PSNR of 20.
- **Final Model and Tradeoffs:** Our final model stands out with significantly fewer parameters while outperforming the previous approaches, achieving a **PSNR** of **25.15**. This demonstrates an excellent tradeoff between model complexity and image deblurring performance.
- **Qualitative Observations:** The final model effectively deblurs images. However, a minor color grading effect was sometimes observed which if improved might better the PSNR score.

6 Code Implementation

Project Structure

- `models` : This directory contains all the model checkpoints
- `plots` : This directory contains the training plots
- `config.json` : Holds parameters for code execution
- `main.py` : The main file for training and evaluation
- `eval.py` : The file to be used for evaluation
- `setup_env.py` : Sets up the environment for the project.

Running the Program for Evaluation

To set up the virtual environment, run:

```
python3 setup_env.py
```

Create the custom_test directory(in the main directory) as was given in the evaluation script (containing blur and sharp folders). Then execute the following command which will create a new folder deblur inside the custom_test directory containing the deblurred images generated by our model and will also print the PSNR:

```
python3 eval.py
```