Image sharpening via knowledge Distillation via Restormer-Based Model

Submitted by

Team Aurum,

Manipal Institute of Technology

Problem Statement:

"Create a model to improve image sharpness while conferencing on video, problems such as lowered clarity because of insufficient bandwidth or an unreliable internet connection.

This project solves the task of learning a light super-resolution based heavy and high-performance teacher model. The goal is to learn an efficient CNN that closely approximates the Restormer model responses on cropped patch images so that it can produce fast and efficient image sharpening on high-resolution images.

The project was performed on Kaggle Notebooks and VSCode, using its free GPU runtime environment. The platform allowed us to access image datasets and conduct a lot of experimentation with the model.

Model Architecture:

Teacher Model: Restormer

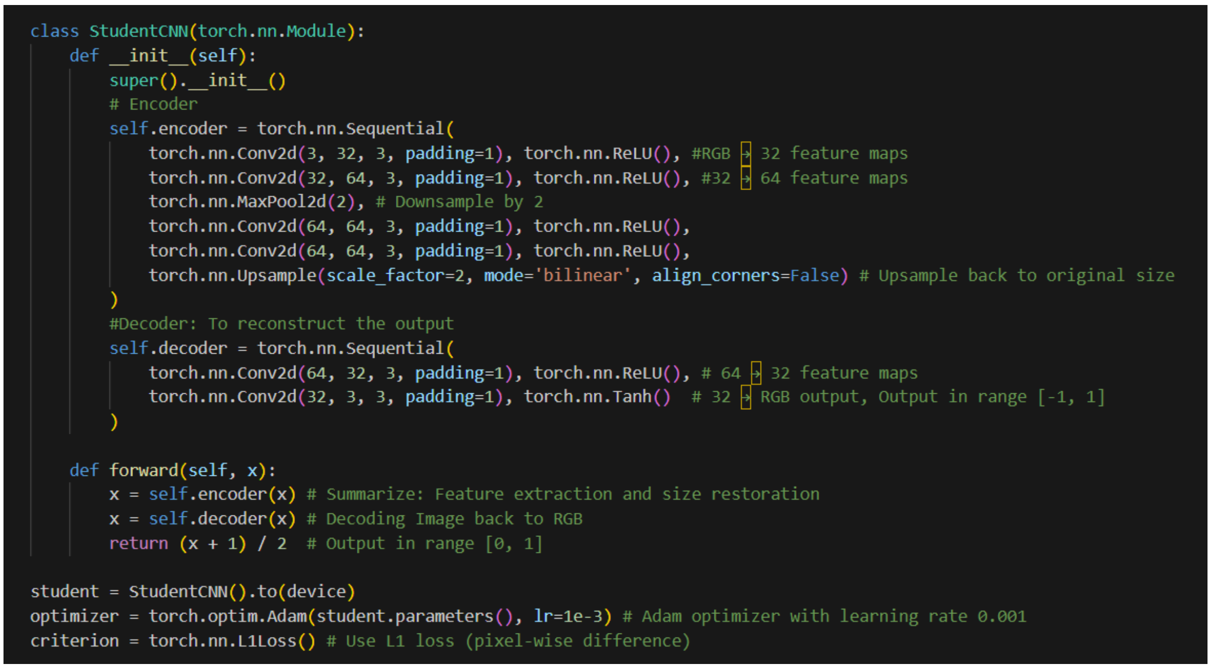
The teacher model is the pre-trained Defocus and Deblur Restormer, a transformer architecture designed for specific image restoration tasks. The Restormer model is not tested, but rather its parameters are kept constant and it is not updated with any gradients. Rather, it serves as a source of knowledge for the student model by giving high-quality sharpened images as outputs for provided blurred inputs. These outputs serve as targets, which the student tries to estimate while being trained.

Student Model: Lightweight CNN

The student model used in this paper is a CNN, made using PyTorch, to mimic the outputs of the teacher model, yet it is computationally lightweight and ideal for real-time inference. The network is made up of two parts, an encoder and a decoder. It is basically a 9-layer encoder-decoder CNN with 6 encoder layers and 3 decoder layers,

The encoder starts with two convolutional layers. The initial one converts the 3-channel RGB input into 32 feature maps, with a subsequent second convolution increasing it further to 64 channels. Both convolution layers are followed by a ReLU activation function to add non-linearity. A max-pooling layer halves the image sizes so that the network can extract global features. Two more convolutional layers with ReLU activations preserve the 64-channel dimension, which allows deeper feature extraction. To finish off, a bilinear up-sampling layer recovers the original spatial resolution and essentially reverses the max-pooling operation.

The decoder has two additional convolutional layers. The first one downsamples the feature dimensions from 64 to 32, followed again by ReLU activation. The last layer maps the 32-channel tensor back into a 3-channel RGB image with a Tan h activation function. This limits the output to the range [-1, 1], which is then scaled to the range [0, 1] for the forward pass to bring it in line with the format of the ground truth images.



```python
class StudentCNN(torch.nn.Module):
    def __init__(self):
        super().__init__()
        # Encoder
        self.encoder = torch.nn.Sequential(
            torch.nn.Conv2d(3, 32, 3, padding=1), torch.nn.ReLU(), #RGB → 32 feature maps
            torch.nn.Conv2d(32, 64, 3, padding=1), torch.nn.ReLU(), #32 → 64 feature maps
            torch.nn.MaxPool2d(2), # Downsample by 2
            torch.nn.Conv2d(64, 64, 3, padding=1), torch.nn.ReLU(),
            torch.nn.Conv2d(64, 64, 3, padding=1), torch.nn.ReLU(),
            torch.nn.Upsample(scale_factor=2, mode='bilinear', align_corners=False) # Upsample back to original size
        )
        #Decoder: To reconstruct the output
        self.decoder = torch.nn.Sequential(
            torch.nn.Conv2d(64, 32, 3, padding=1), torch.nn.ReLU(), # 64 → 32 feature maps
            torch.nn.Conv2d(32, 3, 3, padding=1), torch.nn.Tanh()  # 32 → RGB output, Output in range [-1, 1]
        )

    def forward(self, x):
        x = self.encoder(x) # Summarize: Feature extraction and size restoration
        x = self.decoder(x) # Decoding Image back to RGB
        return (x + 1) / 2  # Output in range [0, 1]

student = StudentCNN().to(device)
optimizer = torch.optim.Adam(student.parameters(), lr=1e-3) # Adam optimizer with learning rate 0.001
criterion = torch.nn.L1Loss() # Use L1 loss (pixel-wise difference)
```

The model is trained with the Adam optimizer of learning rate 1e-3, and L1 loss is employed as the reference criterion to estimate the pixel-wise difference between the target and predicted images.

The student model is trained to mimic the teacher output by minimizing a weighted loss consisting of three components.

L1 loss between student's output and teacher's output, L1 loss with the ground truth patch, perceptual loss between high-level features using VGG16 between student output and ground truth.

Data Sources:

The data used in this project is the Image Super Resolution – Unsplash dataset that contains high-resolution images, usually with sizes approximately 1920×1080. Such images are appropriate for training and testing super-resolution models under normal visual scenarios.

To have clean and uniform preprocessing, only images with height and width divisible by 8 were chosen. This restriction prevents edge artifacts when doing down sampling and patch extracting, as well as maintaining compatibility with the Restormer model, which demands input size to be divisible by 8. From the raw data, this filtering was done using OpenCV, and all valid images were kept for subsequent processing.

In order to add randomness without losing reproducibility, the random seed was fixed (seed = 42) for all splitting processes. The available images were randomly divided into a training set

(80%) and a test set (20%) with train_test_split. The training set was divided into two equal parts, referred to as train_A and train_B, and each was used to train a different instance of the student model. From the test set, 50 images were randomly sampled to form a showcase set, intended for qualitative evaluation and final output visualization.

For training, every image in train_A and train_B was divided into non-overlapping patches of dimensions 256×256, with a stride of 256. Patch-wise training enables the model to learn from local structures and optimizes the use of the GPU's memory. The overall number of patches created was automatically calculated by looping through the height and width of each image. Patches were saved under their corresponding directories for train_A and train_B.

In parallel, the source test and showcase high-resolution images were merely replicated unchanged. These were employed in inference after training to assess both performance through SSIM (Structural Similarity Index Metric) and visual quality.

Training Process:

Training was conducted with image patches from the train_A subset, comprised of non-overlapping 256 × 256 patches obtained through a stride of 256. For realistic simulation of blurring, each patch was initially down sampled by 2 times using linear interpolation followed by up sampling back to original size using bicubic interpolation. These blurred patches were then fed to both the teacher and student networks for supervised learning.

The student model was trained within a knowledge distillation setup, with the teacher model producing high-quality deblurring for the identical blurred inputs. The student was optimized to copy these outputs and match the ground truth. Training was done over 15 epochs with the Adam optimizer at a learning rate of 1e-3. Training was done with a batch size of 1 because of patch-wise processing.

A composite loss function was employed for supervision, which was a combination of three terms:

L1 loss between student output and teacher output

L1 loss between student output and the ground truth patch,

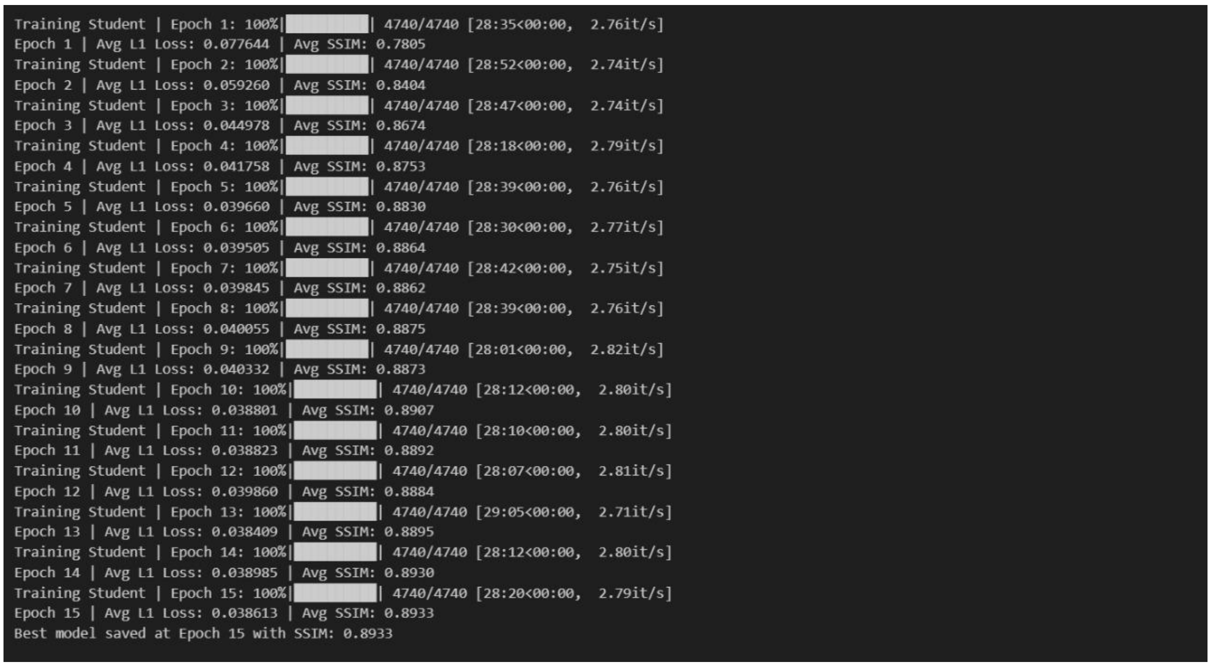
Perceptual loss, calculated using intermediate feature maps from an early stopped pre-trained VGG16 network.

The overall loss formula was given as:

$$\text{Loss} = 0.9 \times L1(S,T) + 0.1 \times L1(S,GT) + 0.005 \times \text{Perceptual}(S,GT)$$

where S represents the student output, T represents the teacher output, and GT represents the ground truth image. For calculating perceptual loss, both the ground truth and student output are resized to 224×224, and non-trainable intermediate layers of VGG16 are used to extract features.

Two student models were trained separately: one using train_A and the other using train_B, yielding two individually optimized weight files: Train_A.pth and Train_B.pth.



```
Training Student | Epoch 1: 100%|██████████| 4740/4740 [28:35<00:00,  2.76it/s]
Epoch 1 | Avg L1 Loss: 0.077644 | Avg SSIM: 0.7805
Training Student | Epoch 2: 100%|██████████| 4740/4740 [28:52<00:00,  2.74it/s]
Epoch 2 | Avg L1 Loss: 0.059260 | Avg SSIM: 0.8404
Training Student | Epoch 3: 100%|██████████| 4740/4740 [28:47<00:00,  2.74it/s]
Epoch 3 | Avg L1 Loss: 0.044978 | Avg SSIM: 0.8674
Training Student | Epoch 4: 100%|██████████| 4740/4740 [28:18<00:00,  2.79it/s]
Epoch 4 | Avg L1 Loss: 0.041758 | Avg SSIM: 0.8753
Training Student | Epoch 5: 100%|██████████| 4740/4740 [28:39<00:00,  2.76it/s]
Epoch 5 | Avg L1 Loss: 0.039660 | Avg SSIM: 0.8830
Training Student | Epoch 6: 100%|██████████| 4740/4740 [28:30<00:00,  2.77it/s]
Epoch 6 | Avg L1 Loss: 0.039505 | Avg SSIM: 0.8864
Training Student | Epoch 7: 100%|██████████| 4740/4740 [28:42<00:00,  2.75it/s]
Epoch 7 | Avg L1 Loss: 0.039845 | Avg SSIM: 0.8862
Training Student | Epoch 8: 100%|██████████| 4740/4740 [28:39<00:00,  2.76it/s]
Epoch 8 | Avg L1 Loss: 0.040055 | Avg SSIM: 0.8875
Training Student | Epoch 9: 100%|██████████| 4740/4740 [28:01<00:00,  2.82it/s]
Epoch 9 | Avg L1 Loss: 0.040332 | Avg SSIM: 0.8873
Training Student | Epoch 10: 100%|██████████| 4740/4740 [28:12<00:00,  2.80it/s]
Epoch 10 | Avg L1 Loss: 0.038801 | Avg SSIM: 0.8907
Training Student | Epoch 11: 100%|██████████| 4740/4740 [28:10<00:00,  2.80it/s]
Epoch 11 | Avg L1 Loss: 0.038823 | Avg SSIM: 0.8892
Training Student | Epoch 12: 100%|██████████| 4740/4740 [28:07<00:00,  2.81it/s]
Epoch 12 | Avg L1 Loss: 0.039860 | Avg SSIM: 0.8884
Training Student | Epoch 13: 100%|██████████| 4740/4740 [29:05<00:00,  2.71it/s]
Epoch 13 | Avg L1 Loss: 0.038409 | Avg SSIM: 0.8895
Training Student | Epoch 14: 100%|██████████| 4740/4740 [28:12<00:00,  2.80it/s]
Epoch 14 | Avg L1 Loss: 0.038985 | Avg SSIM: 0.8930
Training Student | Epoch 15: 100%|██████████| 4740/4740 [28:20<00:00,  2.79it/s]
Epoch 15 | Avg L1 Loss: 0.038613 | Avg SSIM: 0.8933
Best model saved at Epoch 15 with SSIM: 0.8933
```

Fig: Average SSIM from training using dataset A

Model Evaluation:

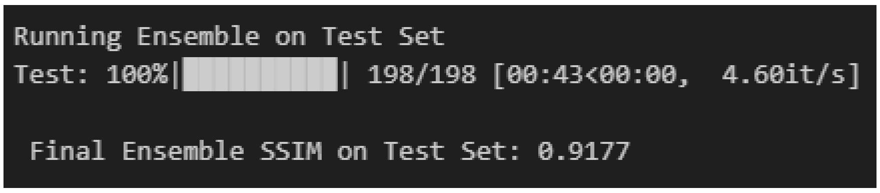
Both the test and showcase datasets were evaluated. Each image was down sampled and up sampled using the same pipeline prior to evaluation. The student models received the blurred image as input and their respective outputs were combined (ensembled) to generate the final reconstruction.

This approach reduces learned bias in single models and takes advantage of train_A and train_B diversity to provide stronger and visually good outputs.

The Structural Similarity Index (SSIM) was employed as the main evaluation criterion, calculated for every deblurred image relative to the ground truth. The best model with the highest average SSIM on the training patches was saved as best_student.pth.

This training and ensemble assessment strategy guaranteed that the final student output maintained quality while leveraging the joint strengths of two independently trained networks.

Ultimately, in testing the student model, it achieved an average SSIM score of 0.9177.



```
Running Ensemble on Test Set
Test: 100%|██████████| 198/198 [00:43<00:00,  4.60it/s]

 Final Ensemble SSIM on Test Set: 0.9177
```

Fig: Average SSIM after evaluation

Conclusion:

This work illustrates a real-world application of distilling knowledge for image super-resolution, with Restormer being the teacher and a small CNN as the student. The student network was able to successfully learn to approximate the teacher's outputs on high resolution image patches. Perception loss was utilized to further improve visual quality over plain pixel-wise loss. Real-time deployment or applications are feasible under this framework in low-compute devices.

Contributions:

Aditya Toley: Training on dataset A, ensemble logic, test

Nikhil Narayanan O: Training on dataset B, dataset management

Guided by,

Subraya Krishna Bhat,

Assistant Professor, Department of Mechanical & Industrial Engineering