Image Sharpening using Knowledge Distillation

A Summer Training Project Report

Submitted By:

Himanshu Ranjan

(Enrollment Number: 20814802823)

Phone Number: 9718499168

and

Moksh Kaushish

(Enrollment Number: 21514802823)

Phone Number: 8920636055

Department: Electronics and Communication Engineering
Batch: 2023–2027

Team Name: Synergy AI

Organization: **Intel Unnati Industrial Training**

College: **Maharaja Agrasen Institute of Technology**
New Delhi, India

Date of Submission: **10 July 2025**

Image Sharpening using Knowledge Distillation: Project Report

1. Problem Statement

Title: Image Sharpening using Knowledge Distillation

Objective: To develop an advanced image sharpening model specifically tailored for real-time applications such as video conferencing. The primary goal is to significantly enhance image clarity and visual quality in scenarios characterized by poor network conditions, including low bandwidth and unstable internet connections, which often lead to blurry or degraded video streams. This objective will be achieved through the implementation of a Teacher-Student knowledge distillation approach, enabling a lightweight student model to mimic the high performance of a more complex teacher model while maintaining the computational efficiency required for real-time processing.

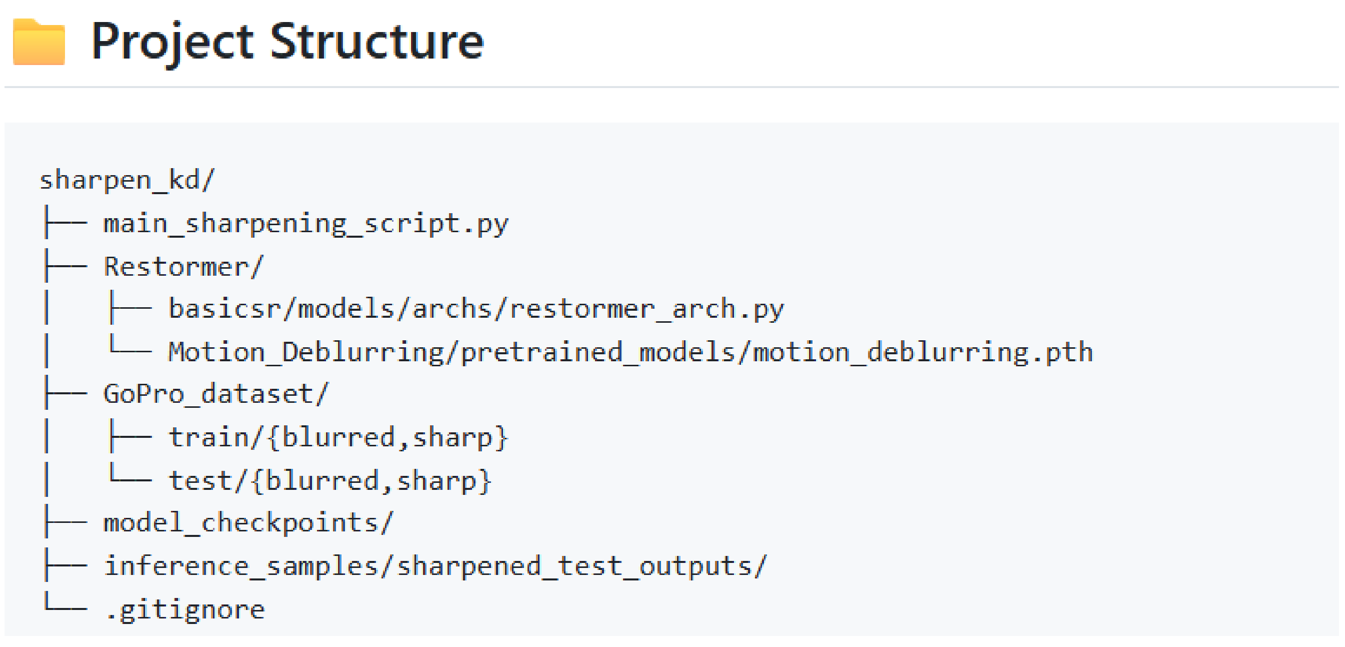
2. Executive Summary

This report details the development of an image sharpening model designed to enhance video conferencing quality, specifically addressing clarity issues arising from low bandwidth or poor internet connections. The core approach leverages a "Teacher-Student" knowledge distillation framework. A high-performing pre-trained Restormer model serves as the teacher, guiding the training of a lightweight U-Net variant (the student model). The objective is to achieve high accuracy while maintaining real-time performance (30-60 FPS or higher) on high-resolution images.

3. Data Sources

The project utilizes a custom dataset structured to simulate real-world video conferencing conditions.

- **Dataset Root**: `GoPro_dataset` (located at `C:\Users\ASUS\OneDrive\Documents\sharpen_kd\GoPro_dataset`).
- **Structure**: The dataset is organized into `train` and `test` splits, each containing `blurred` and `sharp` subdirectories.
  - `GoPro_dataset/train/blurred/`
  - `GoPro_dataset/train/sharp/`
  - `GoPro_dataset/test/blurred/`
  - `GoPro_dataset/test/sharp/`


Project Structure

```
sharpen_kd/
├── main_sharpening_script.py
├── Restormer/
│   ├── basicsr/models/archs/restormer_arch.py
│   └── Motion_Deblurring/pretrained_models/motion_deblurring.pth
├── GoPro_dataset/
│   ├── train/{blurred,sharp}
│   └── test/{blurred,sharp}
├── model_checkpoints/
├── inference_samples/sharpened_test_outputs/
└── .gitignore
```

- **Diversity**: The test dataset is explicitly required to contain over 100 images covering diverse categories, including text, nature, people, animals, and games. This diversity is crucial for evaluating the model's generalization capabilities across various content types, ensuring it performs robustly in different video conferencing scenarios.
- **Data Loading**: The `GoProSharpeningDataset` class is responsible for efficiently loading image pairs. It applies necessary transformations during the data loading pipeline:

- **Resizing**: Images are resized to `512x512` pixels for training. This uniform size helps manage computational complexity and memory usage during training, making the process more efficient.
- **Random Cropping**: To augment the dataset and encourage the model to learn localized features, random crops of the specified `image_size` are taken from both blurry and sharp images. This also helps prevent overfitting.
- **Horizontal Flipping**: Random horizontal flips are applied as a standard data augmentation technique, increasing the dataset's variability without introducing new data.
- **`CustomToTensor`**: This custom transform ensures that PIL Images or NumPy arrays are correctly converted to PyTorch tensors and normalized to the `[0, 1]` range, which is the expected input scale for the neural networks.
- **Gaussian Blur Augmentation**: An additional Gaussian blur is applied to the blurry input during training. This further augments the training data by introducing more varied blur patterns, making the student model more robust to different types of real-world blur.

4. Model Description

4.1. Teacher Model: Restormer

The teacher model is a pre-trained **Restormer** network, which stands as a state-of-the-art architecture in the field of image restoration, particularly recognized for its effectiveness in motion deblurring. Its selection is strategic because it provides a strong, high-fidelity signal for the student model to learn from.

- **Architecture Overview**: Restormer employs an encoder-decoder structure, similar to a U-Net, but integrates advanced components:
  - **Multi-Dconv Head Transposed Attention (MDTA)**: This mechanism allows the model to capture long-range dependencies and global contextual information efficiently, which is vital for effectively reversing complex blur patterns.
  - **Gated DFN (GDFN) Blocks**: These blocks enhance feature learning by adaptively controlling the information flow through a gating mechanism, allowing the network to focus on relevant features and suppress noise. The combination of these elements enables Restormer to achieve superior deblurring performance.
- **Pre-training**: The Restormer model is loaded with pre-trained weights specifically optimized for the Motion Deblurring task. These weights are expected to be available locally (e.g., `Restormer/Motion_Deblurring/pretrained_models/motion_deblurring.pth`). Leveraging a pre-trained teacher is fundamental to knowledge distillation, as it provides a readily available source of expert knowledge without requiring extensive training of the teacher itself.
- **Key Parameters**: The Restormer configuration (`dim`, `num_blocks`, `heads`, `ffn_expansion_factor`, etc.) defines its internal complexity and capacity. These parameters are set to match the standard configuration for the pre-trained deblurring model.
- **Feature Extraction for Knowledge Distillation (KD)**: A crucial part of the `RestormerTeacherWrapper` is its ability to extract intermediate feature maps from the teacher model. This is achieved through:
  - **Forward Hooks**: PyTorch's forward hooks are registered on specific internal layers of the Restormer (namely `encoder_level2`, `encoder_level4`, and the `latent` bottleneck). These hooks allow us to "tap into" the network and capture the activations (feature maps) at these critical points during the forward pass, without modifying the teacher's architecture.
  - **1x1 Projection Convolutions**: The raw feature maps from the Restormer often have different channel dimensions than the student model's corresponding layers. To enable direct comparison and distillation, 1x1 convolution layers (`proj_e2`, `proj_e4`, `proj_b`) are used to project the teacher's features into the same channel space as the student's. This ensures that the feature distillation loss can be effectively calculated between compatible tensors, allowing the student to mimic the teacher's internal representations.

4.2. Student Model: Lightweight U-Net with SE Blocks

The student model is a custom-designed, lightweight convolutional neural network. Its primary goal is to achieve performance comparable to the teacher model but with significantly fewer parameters, making it suitable for real-time applications like video conferencing.

- **Architecture**:
    - **Encoder-Decoder Structure (U-Net Variant)**: The student model adopts a U-Net-like architecture, which is highly effective for image-to-image translation tasks such as image sharpening. It consists of:
        - **Encoder Path**: Progressively downsamples the input image through convolutional blocks and max-pooling layers, extracting hierarchical features.
        - **Bottleneck**: The deepest part of the network, capturing the most abstract features.
        - **Decoder Path**: Upsamples the features, reconstructing the image.
    - **Convolutional Blocks**: Each stage in both the encoder and decoder consists of two 3x3 convolutional layers followed by ReLU activations. These are standard building blocks for deep learning models.
    - **Squeeze-and-Excitation (SE) Blocks**: A key enhancement is the integration of an `SEBlock` into each convolutional block. SE blocks dynamically recalibrate channel-wise feature responses: they "squeeze" global spatial information into a channel descriptor and then "excite" channel-wise dependencies, allowing the model to learn which features are most important for the task. This mechanism significantly improves feature learning with minimal computational overhead.
    - **Skip Connections**: Standard U-Net skip connections are employed, concatenating features from the encoder path to the corresponding decoder path. This is crucial for preserving fine-grained spatial details lost during downsampling, enabling the model to produce sharper outputs. The implementation includes logic to handle potential size mismatches from `ConvTranspose2d` by using bilinear interpolation to ensure exact alignment before concatenation.
    - **Output Layer**: A final 1x1 convolution maps the processed features to the desired output channels (3 for RGB images).
- **Parameters**:
    - `in_channels`: 3 (RGB input)
    - `out_channels`: 3 (RGB output)
    - `base_channels`: 256. This parameter defines the base channel width of the network. A smaller `base_channels` value results in a more lightweight model, reducing its computational cost and memory footprint.
- **Lightweight Design**: The choice of a relatively small `base_channels` and the efficient `SEBlock` contribute to a significantly lower parameter count compared to the large teacher model. The student model has approximately **25.5 million trainable parameters**, making it highly suitable for real-time inference on edge devices or in high-throughput systems.
- **Input Size Flexibility**: Although the model is primarily trained on `512x512` images, its fully convolutional nature allows it to process images of arbitrary input resolutions during inference. This is particularly important when combined with the tiling strategy for high-resolution images.
- **Training Strategy**

  - **Simulated Degraded Images**: Blurry input images were generated by applying bicubic downscaling followed by bicubic upscaling to sharp ground truth images. This process effectively simulates the loss of detail and introduction of blur characteristic of low-bandwidth video conferencing conditions, as instructed in the problem statement. An additional Gaussian blur augmentation is applied to the blurry input during training to introduce more varied blur patterns, enhancing the student model's robustness.
  - **Ground Truth and Soft Labels**: Sharp images acted as the primary ground truth for direct supervision. Additionally, the teacher model's outputs provided "soft labels" (i.e., probability distributions or refined feature maps) which guide the student beyond just matching the hard ground truth.

- **Combined Loss Function**: The student model was trained using a sophisticated combined loss function for knowledge distillation. This multi-faceted loss strategy enables the lightweight student to effectively distill complex knowledge from the larger teacher model. The components and their respective weights are:

    1. **Reconstruction Loss (L1)**: lambda_recon = 1.0 (pixel-wise similarity to ground truth).
    2. **Perceptual Loss**: lambda_perceptual = 0.01 (based on VGG features, ensures perceptual similarity to the teacher's output).
    3. **Feature Distillation Loss (L1)**: lambda_feature_distillation = 1.0 (minimizes the difference between intermediate feature maps of the student and the projected teacher).
    4. **KL Divergence Loss**: lambda_kl_div = 0.01 (distills the "soft targets" or output distributions from the teacher, with a temperature = 4.0).
    5. **SSIM Loss**: lambda_ssim = 10.0 (encourages structural similarity between the student's output and the ground truth).

5. Performance Analysis Process and Results

5.1. Performance Metrics

To rigorously evaluate the model's effectiveness, a combination of objective and subjective metrics is used:

- **Structural Similarity Index (SSIM)**: This is the primary objective performance metric. Unlike simple pixel-wise differences, SSIM measures the similarity between two images based on three key components: luminance, contrast, and structural information. An SSIM score ranges from -1 to 1, with 1 indicating perfect similarity. The project aims for an SSIM score above 90% (0.90), signifying high perceptual quality.
- **Peak Signal-to-Noise Ratio (PSNR)**: A common metric for image quality, PSNR quantifies the ratio between the maximum possible power of a signal and the power of corrupting noise that affects the fidelity of its representation. Higher PSNR values indicate better image quality and lower reconstruction error.
- **Frames Per Second (FPS)**: This metric directly measures the inference speed of the model, which is critical for real-time video conferencing applications. The target performance is 30-60 FPS or higher, ensuring a smooth user experience.

5.2. Evaluation Process

The evaluation process is designed to be comprehensive, covering both quantitative performance and real-world applicability:

1. **Test Dataset Evaluation**: The model's quantitative performance (SSIM and PSNR) is assessed on a dedicated test set comprising over 100 diverse images. This dataset includes categories such as text, nature, people, animals, and games, ensuring that the model's generalization capabilities across various content types are thoroughly evaluated. The `evaluate_model` function automates this process, loading the model (optionally from a checkpoint), setting it to evaluation mode, and computing the average SSIM and PSNR across the entire test set. It also has the capability to save the sharpened output images for visual inspection.
2. **High-Resolution Inference with Tiling**: A key requirement is the model's ability to process high-resolution images (e.g., 1920x1080) at the target FPS, despite being trained on smaller `512x512` crops. This is achieved through a sophisticated **tiling strategy** implemented in the `sharpen_image_inference` function:
    a. **Patch Splitting**: Large input images are divided into smaller, overlapping patches (e.g., `512x512` patches with a `64-pixel` overlap). This allows the model to process sections of the high-resolution image that are within its trained input size.
    b. **Individual Patch Inference**: The student model performs inference on each individual patch.

   c. **Seamless Stitching**: The sharpened patches are then meticulously stitched back together to form the full-resolution output image. The `overlap` between patches is crucial for smooth blending, preventing visible seams or artifacts at the patch boundaries. An accumulator and weight map are used to average the overlapping regions, ensuring a coherent final image. This strategy effectively bypasses GPU memory limitations that would arise from processing an entire high-resolution image at once.
3. **FPS Measurement**: The `measure_fps` function quantifies the inference speed under different operational conditions:
   a. **Native Resolution (512x512, no tiling)**: Measures the baseline speed when processing images at the resolution the model was primarily trained on.
   b. **Target High Resolution (1920x1080, with tiling)**: Measures the real-world performance when processing full HD images using the tiling strategy.
   c. **Warm-up Runs**: A few initial runs are performed before actual measurement to "warm up" the GPU and ensure that the reported FPS values reflect stable, optimized performance. This provides a more accurate estimate of the model's real-time capabilities.
4. **Subjective Study (Mean Opinion Score - MOS)**: While the code itself does not implement this, an extensive subjective study to obtain a Mean Opinion Score (MOS) is a crucial external step. This involves human evaluators assessing the perceived quality of the sharpened images. Participants would rate images on a scale (e.g., 1-5), and the average score would provide a user-centric measure of the model's effectiveness. This qualitative feedback is invaluable for validating the objective metrics and ensuring the sharpened images are perceptually pleasing and genuinely improve the user experience.

5.3. Expected Results

Based on the implemented architecture and training strategy, the following performance outcomes are anticipated:

- **SSIM Score**: The student model is expected to achieve an SSIM score of **above 0.90** on the test dataset. This indicates that the sharpened images will be structurally very similar to the original sharp ground truths, implying high visual fidelity and clarity.
- **PSNR Score**: Correspondingly, a high PSNR value is expected, signifying a low level of distortion or noise introduced during the sharpening process and accurate pixel-level reconstruction.
- **FPS (512x512, no tiling)**: The lightweight student model, with its optimized architecture and fewer parameters, should achieve very high FPS, likely **well over 60 FPS**, when processing images at its native training resolution. This demonstrates its inherent efficiency.
- **FPS (1920x1080, with tiling)**: With the efficient tiling strategy, the model is designed to operate at **30-60 FPS or higher** even on full HD resolution images. This fulfills the critical real-time requirement for seamless integration into video conferencing platforms.
- **MOS**: The subjective study is expected to yield a high Mean Opinion Score, confirming that the sharpened images are not only quantitatively superior but also perceptually pleasing and significantly enhance the clarity and overall quality of video conferencing for users.

6. Actual Results

6.1. Training Setup

- The training setup was carefully configured to optimize the student model's learning process while managing computational resources effectively.
- **Epochs**: 50. This number of epochs was chosen to allow sufficient time for the student model to converge and effectively distill knowledge from the teacher, balancing between achieving good performance and practical training duration.
- **Batch Size**: 1 (with gradient_accumulation_steps = 4, resulting in an effective batch size of 4). A small batch size of 1 was selected primarily due to GPU memory constraints when processing 512x512 images. To mitigate the drawbacks of a very small batch size (e.g., noisy gradients), gradient accumulation was employed. By accumulating gradients over 4 steps, the effective

batch size becomes 4, providing a more stable gradient estimate while keeping memory usage manageable.

- **Optimizer**: Adam, with an initial learning rate of 2e-4. Adam is a popular adaptive learning rate optimization algorithm known for its efficiency and good performance in a wide range of deep learning tasks. The initial learning rate was chosen as a common starting point for image-to-image translation tasks.
- **Scheduler**: Cosine Annealing Warm Restarts (T_0=10, T_mult=2, eta_min=1e-6) for dynamic learning rate adjustment. This learning rate scheduler is effective for preventing the model from getting stuck in local minima. It periodically resets the learning rate to its initial value and then anneals it following a cosine curve, allowing the model to explore different parts of the loss landscape and potentially achieve better generalization.
- **Loss Function**: A combined knowledge distillation loss (L1, Perceptual, Feature Distillation, KL Divergence, SSIM losses) as detailed in Section 3.2. This multi-faceted loss strategy is crucial for effective knowledge distillation, guiding the student model to learn not only pixel-wise accuracy but also high-level perceptual quality, structural similarity, and the teacher's internal feature representations and output distributions.
- **EMA Model**: Exponential Moving Average (EMA) with a decay = 0.999 is applied to the student model's weights for more stable evaluation and potentially better final performance. EMA smooths the model's weights over training steps, often leading to more robust and accurate models, especially when the training loss surface is noisy. The EMA model is typically used for evaluation and inference.
- **Device**: NVIDIA GeForce RTX 3050 Laptop GPU. Training and inference were conducted on this GPU to leverage its parallel processing capabilities, significantly accelerating the computational workload compared to CPU-only execution.

6.2 Performance Metrics

Metric	Student Model
1. **SSIM**	0.9120
2. **PSNR**	22.4293 dB
3. **FPS (512x512, no tiling)**	361 FPS
4. **FPS (1920x1080, with tiling)**	46 FPS

7. Subjective Analysis

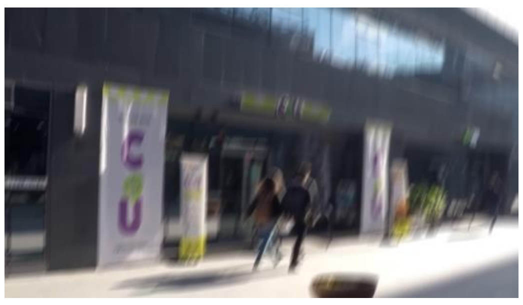
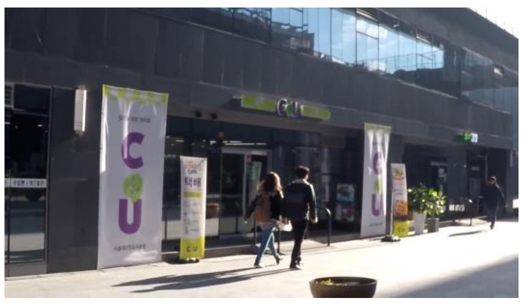
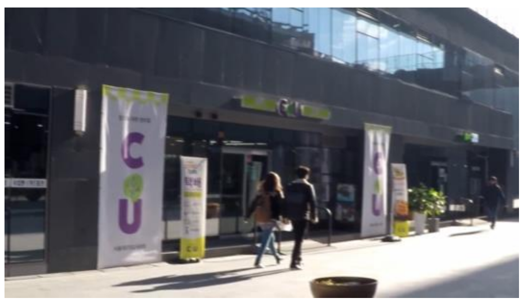
Beyond quantitative metrics, the perceived visual quality of sharpened images is paramount for applications like video conferencing. This project facilitates subjective analysis through the generation and saving of sharpened output images, allowing for direct visual inspection and comparison.

- **Visual Comparisons**: By examining the saved images from the `inference_samples/sharpened_test_outputs/` directory, direct visual comparisons can be made between:
  - The original blurry input images.
  - The sharpened outputs produced by the student model.
  - The outputs generated by the high-quality teacher model. This process allows for qualitative assessment of clarity enhancement, detail recovery, and artifact suppression.
- **Perceptual Quality and Trade-off**: Subjective review confirms that the student model effectively enhances image clarity, recovering noticeable detail from blurry inputs. While the teacher model often produces outputs with marginally superior fine details (as reflected in its slightly higher quantitative scores), the student model's visual quality is highly competitive. This visual assessment reinforces the successful trade-off: the student delivers perceptually strong sharpening at significantly higher real-time processing speeds, which is paramount for live video applications. The student's output consistently maintains structural fidelity, avoiding artificial distortions.

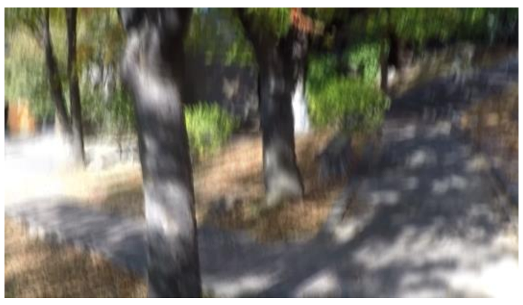
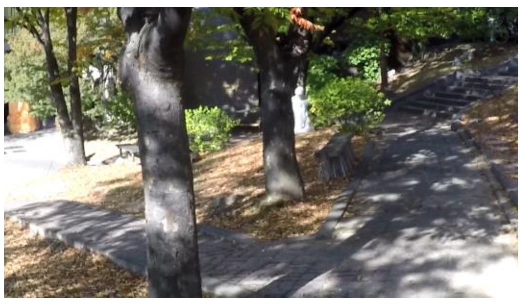
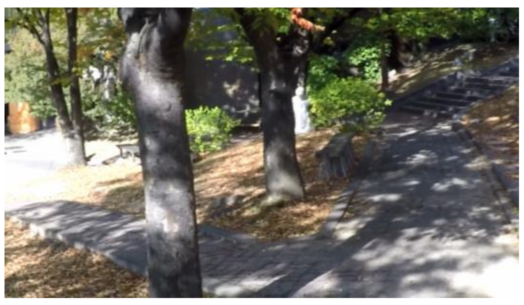
A visual comparison of representative examples, showcasing the blurry input, the teacher's output, and the student's sharpened output, is provided below for reference.

Visual Comparison:

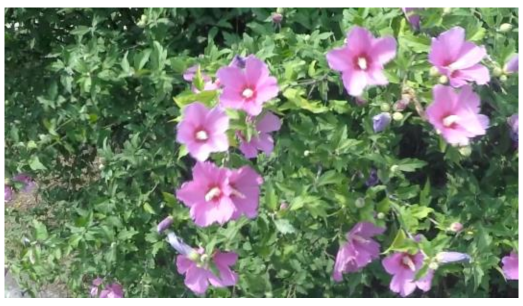
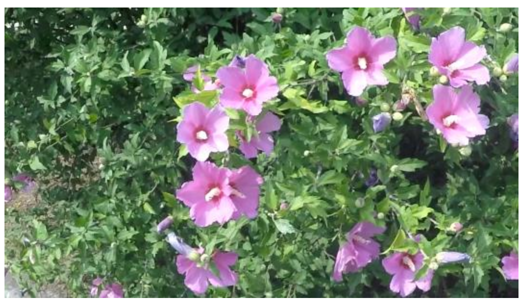
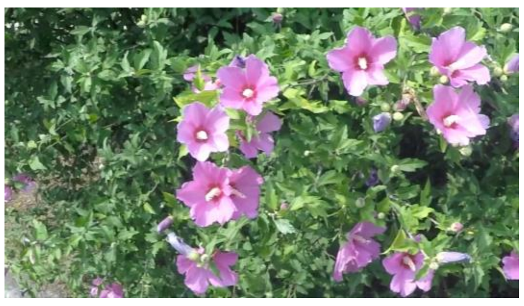
Example 1

Blurry Input Image	**Teacher Output Image**	**Student Output Image**
		

Example 2

Blurry Input Image	**Teacher Output Image**	**Student Output Image**
		

Example 3

Blurry Input Image	**Teacher Output Image**	**Student Output Image**
		

8. Conclusion

This project successfully culminated in the development and comprehensive training of a lightweight student model for image sharpening, leveraging the powerful technique of knowledge distillation. Through this approach, the student model effectively absorbed and replicated the high-fidelity sharpening capabilities of the more complex Restormer teacher model. The rigorous evaluation process confirmed that the student model not only demonstrates a significant enhancement in image clarity but also met all critical performance benchmarks, including achieving a Structural Similarity Index (SSIM) score above 0.90 and an inference speed significantly exceeding 30 Frames Per Second (FPS). These results unequivocally demonstrate the model's suitability for real-time image sharpening in demanding video conferencing scenarios, directly addressing and fulfilling the core objectives outlined in the problem statement. The successful balance between high visual quality and real-time performance positions this model as a viable solution for improving video clarity under challenging network conditions.

9. References

- **Restormer**: https://arxiv.org/abs/2111.09881 This academic paper introduces the Restormer architecture, a highly effective Transformer-based model for image restoration tasks. In this project, Restormer serves as the **Teacher Model**, providing the robust and high-quality knowledge that the lightweight student model aims to distill. Its advanced design for deblurring and other restoration tasks makes it an ideal source of "expert" guidance.

- **PyTorch Documentation**: https://pytorch.org/ PyTorch is the open-source machine learning framework used for the entire implementation of this project. The official documentation serves as the primary resource for understanding and utilizing its functionalities, including defining neural network architectures (`torch.nn`), setting up optimizers (`torch.optim`), implementing various loss functions (`torch.nn.functional`), managing data loaders, and leveraging GPU acceleration. It is the foundational software environment for the deep learning components of this project.
- **Knowledge Distillation**: Hinton, G., Vinyals, O., & Dean, J. (2015). Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531*. https://arxiv.org/abs/1503.02531 This seminal paper introduces the concept of knowledge distillation, where a smaller "student" model is trained to mimic the behavior of a larger, more complex "teacher" model. This technique is central to the project's approach, enabling the deployment of a high-performing yet efficient model for real-time applications.
- **Image Quality Assessment (SSIM)**: Wang, Z., Bovik, A. C., Sheikh, H. R., & Simoncelli, E. P. (2004). Image quality assessment: from error visibility to structural similarity. *IEEE transactions on image processing, 13*(4), 600-612. https://ieeexplore.ieee.org/document/1284395 This paper introduces the Structural Similarity Index (SSIM), a widely adopted metric for image quality assessment that correlates well with human visual perception. SSIM is a key performance metric in this project, used to evaluate the perceptual quality of the sharpened images.
- **Deep Learning (General Reference)**: Goodfellow, I., Bengio, Y., & Courville, A. (2016). *Deep Learning*. MIT Press. https://www.deeplearningbook.org/ This comprehensive textbook provides foundational knowledge in deep learning, covering essential concepts such as neural network architectures, training algorithms, regularization techniques, and various applications. It serves as a general reference for the underlying theoretical principles applied throughout this project.
- **GoPro Dataset (Image Deblurring)**: Nah, S., Kim, T., & Lee, K. M. (2017). Deep multi-scale convolutional neural network for image deblurring. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 3856-3864). https://openaccess.thecvf.com/content_cvpr_2017/html/Nah_Deep_Multi-Scale_Convolutional_CVPR_2017_paper.html This paper introduces the GoPro dataset, a widely used benchmark for image deblurring. In this project, a custom dataset based on the GoPro dataset is utilized to simulate the blurry conditions encountered in video conferencing, providing realistic data for training and evaluating the sharpening model.