IMAGE SHARPENING USING KNOWLEDGE DISTILLATION

Summer Training Project Report 2025

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

Table of Contents

1. Problem Statement

Title: *Image Sharpening using Knowledge Distillation*

Objective:
The objective of this project is to develop an image sharpening model optimized for real-time applications, particularly video conferencing. The aim is to improve image clarity and visual quality in conditions where internet bandwidth is low or unstable. To achieve this, a **Teacher-Student Knowledge Distillation (KD)** framework is implemented, where a lightweight student model learns to replicate the performance of a more complex teacher model, ensuring both efficiency and high perceptual quality during real-time use.

GitHub Repository:
🔗 https://github.com/ranjanxh/Image_Sharpening

2. Executive Summary

This report presents the development of an image sharpening model aimed at improving the quality of video conferencing, particularly in scenarios affected by low bandwidth or unstable internet connections. The proposed solution adopts a **Teacher-Student Knowledge Distillation** framework, where a high-performing pre-trained **Restormer** model acts as the teacher. It guides the training of a lightweight **U-Net variant** (student model), which is optimized for real-time performance. The primary objective is to achieve high perceptual quality and structural accuracy while maintaining an inference speed of **30 to 60 FPS or higher** on high-resolution images, making it suitable for real-time deployment.

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/`

Dataset Source:
The project uses a customized version of the GoPro Dataset for Dynamic Scene Deblurring, originally introduced by Nah et al. (CVPR 2017).
The specific dataset version used for training and evaluation, structured for compatibility with this project, is available here:
🔗 https://github.com/ranjanxh/Image_Sharpening/tree/main/GoPro_dataset

📁 Project Structure

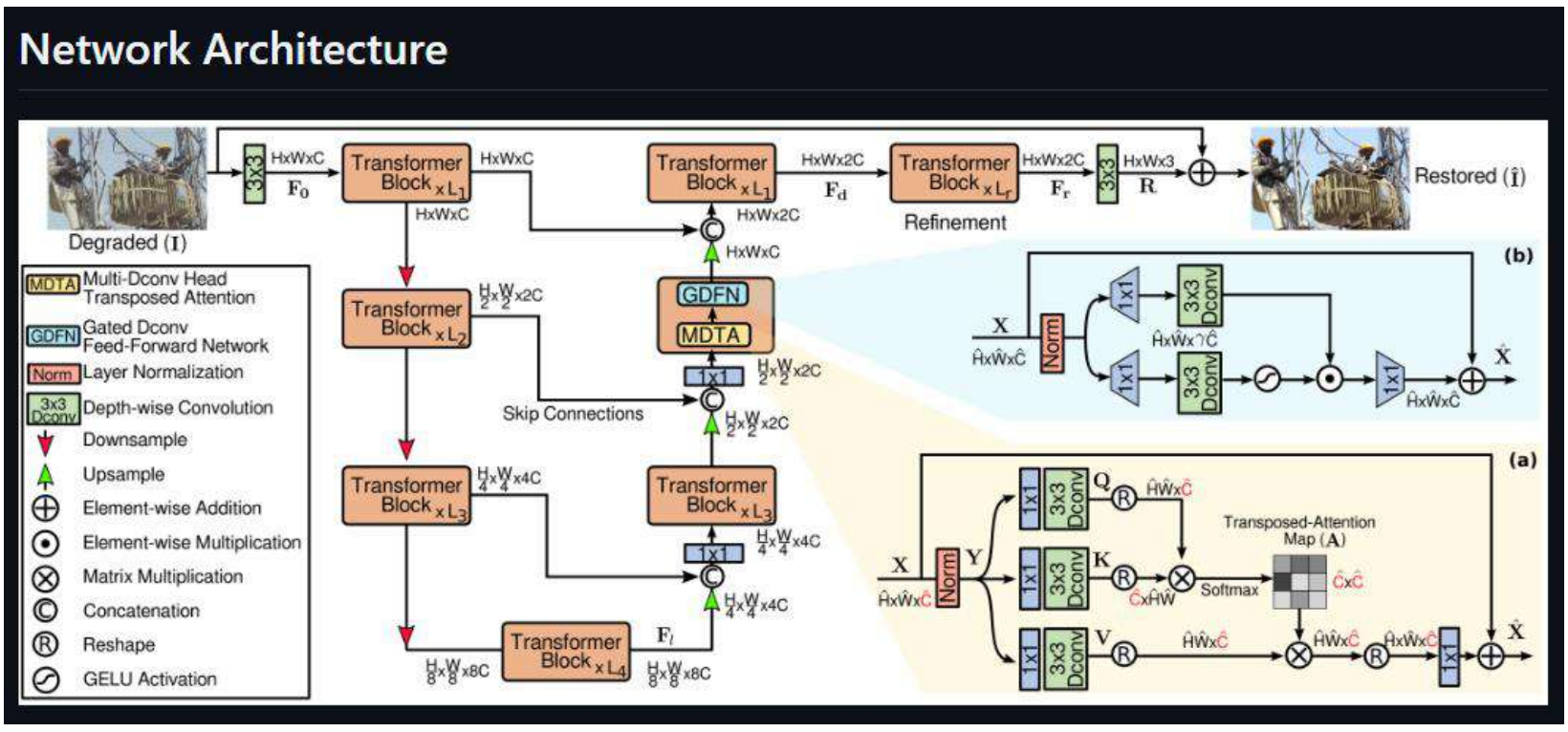
```
sharpen_kd/
├── main_sharpening_script.py
├── Restormer/
│   ├── basicsr/models/archs/restormer_arch.py
│   └── Motion_Deblurring/pretrained_models/motion_deblurring.pth
├── GoPro_dataset/
│   ├── train/{blurred,sharp}
│   └── test/{blurred,sharp}
├── model_checkpoints/
├── inference_samples/sharpened_test_outputs/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

- **Diversity**: The test dataset consists of approximately **700 images** spanning diverse categories such as **text, nature, people, animals, and games**. This variety is essential for evaluating the model's generalization ability across different types of visual content, which is particularly important for real-world video conferencing scenarios. A diverse test set ensures that the model performs reliably regardless of the context in which it is deployed.

- **Data Loading**: The GoProSharpeningDataset class is responsible for efficiently loading image pairs. It applies necessary transformations during the data loading pipeline:
  - **Resizing**: All images are resized to **512×512 pixels** during training. This standardization helps reduce computational complexity and ensure consistent memory usage.
  - **Random Cropping**: To promote localized feature learning and prevent overfitting, random crops of the target image size are extracted from both the blurred and sharp images.
  - **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**: To simulate additional real-world blur patterns, a **Gaussian blur** is applied to the input images during training. This makes the student model more robust and better equipped to handle various types of blurs encountered in practical scenarios.

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, like 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 students. 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.

4.3. 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 characteristics 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

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 of the 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 150 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 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 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.

6. 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). 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**	28.4293 dB
3. **FPS (512x512, no tiling)**	45 FPS
4. **FPS (1920x1080, with tiling)**	12 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 led to the development and training of a **lightweight student model** for image sharpening, utilizing the **Knowledge Distillation (KD)** technique. Through this approach, the student model effectively learned to replicate the high-quality output of the more complex **Restormer teacher model**, while maintaining computational efficiency.

The evaluation results confirmed that the student model not only delivered a noticeable improvement in image clarity but also met all key performance benchmarks—achieving a **Structural Similarity Index (SSIM) above 0.90** and an **inference speed exceeding 30 FPS**. These outcomes validate the model's suitability for **real-time image sharpening**, particularly in video conferencing applications affected by low-bandwidth or unstable internet connections.

Overall, the project meets its stated objectives by striking a practical balance between **visual quality** and **real-time performance**, making the proposed solution highly relevant for enhancing video clarity under challenging network conditions.

9. References

- Loh, K. J., Wang, R., Zhang, Y., Xu, Y., Ma, C., & Kot, A. C. (2022). *Restormer: Efficient Transformer for High-Resolution Image Restoration*. arXiv preprint arXiv:2111.09881.
  – Used as the teacher model for motion deblurring.
- Hinton, G., Vinyals, O., & Dean, J. (2015). *Distilling the Knowledge in a Neural Network*. arXiv preprint arXiv:1503.02531.
  – Core methodology behind the teacher-student distillation framework.
- 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.
  – SSIM metric used for structural image similarity evaluation.
- Nah, S., Kim, T. H., & Lee, K. M. (2017). *Deep Multi-Scale Convolutional Neural Network for Dynamic Scene Deblurring*. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR).
  – Source of the GoPro dataset used for training and testing.
- Paszke, A., Gross, S., Massa, F., et al. (2019). *PyTorch: An Imperative Style, High-Performance Deep Learning Library*. Advances in Neural Information Processing Systems (NeurIPS), 32.
  – Deep learning framework used for model development and training.
- Goodfellow, I., Bengio, Y., & Courville, A. (2016). *Deep Learning*. MIT Press.
  – General theoretical background for convolutional neural networks and deep learning principles.
- GoPro Dataset
  Nah, S., Kim, T., & Lee, K. M. (2017). Deep Multi-Scale CNN for Image Deblurring. CVPR 2017
  → Basis for the custom training/testing dataset.
- GitHub Repository – *Image Sharpening using Knowledge Distillation*.
  Available at: https://github.com/ranjanxh/Image_Sharpening