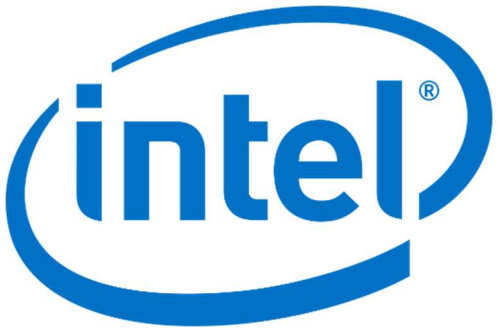


Intel® Unnati Industrial Training – 2025

Image Sharpening Using Knowledge Distillation

Team CodeShamshers

CONTENTS

1. Problem statement
2. System Architecture
3. How the code works
4. Source codes output
5. Video and Github Links

Team Members

M Keerthan Reddy

I Abhishek

G Mahidhar

Problem Statement

To restore and enhance the sharpness and visual fidelity of images that have been degraded due to resolution downscaling, compression artifacts, or lossy transformations, by developing a lightweight, high-performance AI model. This model is trained using a Knowledge Distillation (KD) framework where it learns from a more accurate, high-capacity teacher model to deliver high-quality outputs with real-time efficiency.

1.Data Simulation of Image Degradation

2.Teacher-Student Knowledge Distillation Framework

3.Efficient Training Pipeline

4.Real-Time Ready Architecture

System Architecture

The core design of our project follows a Knowledge Distillation (KD) approach where a powerful teacher model (SwinIR-M) transfers knowledge to a lightweight student model, enabling the student to perform high-quality image sharpening efficiently.

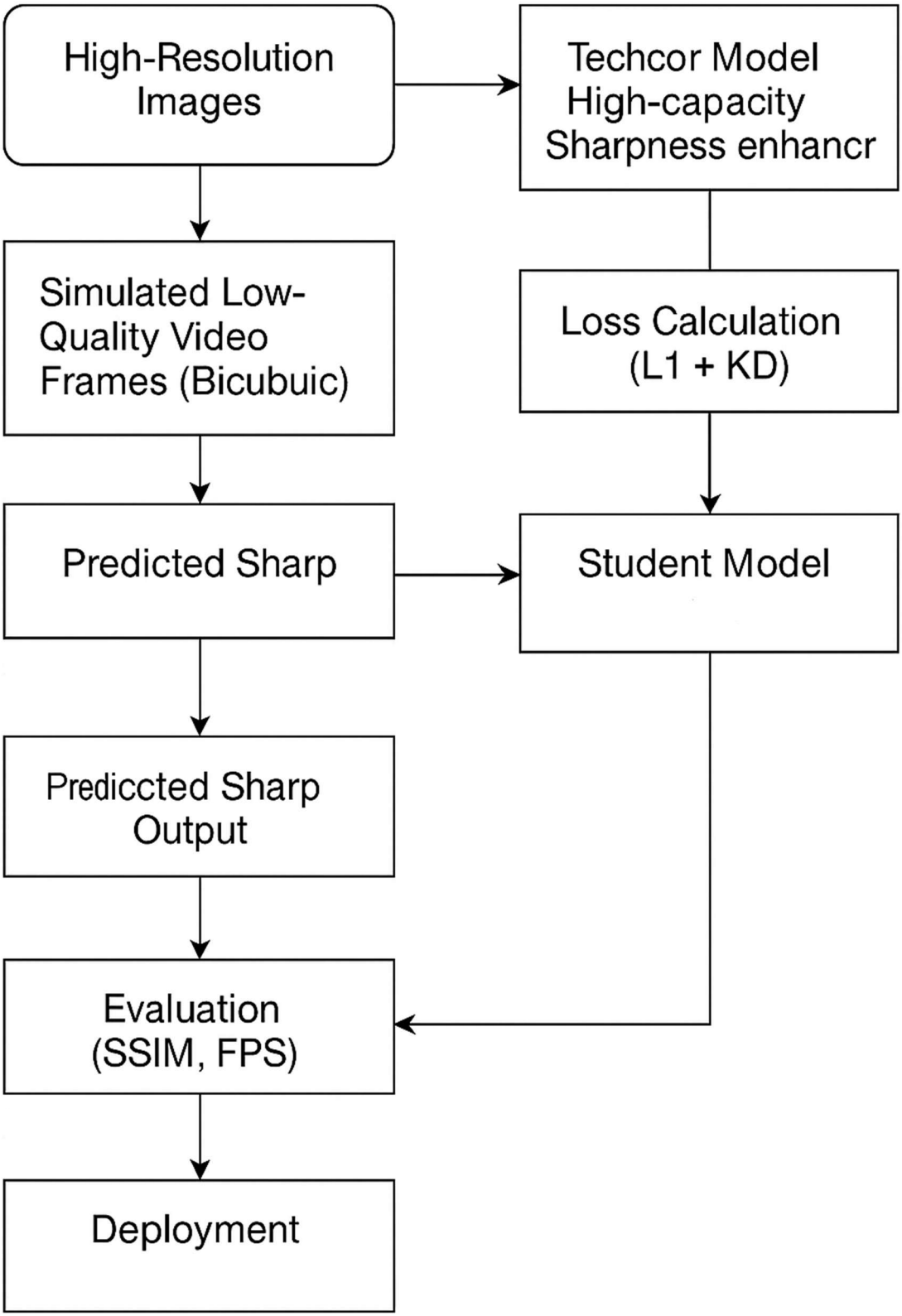


Image Sharpening using Knowledge Distillation

Module	Description
Data Preparation	Downsample high-res images using bicubic or bilinear methods to simulate degraded video frames.
Teacher Model	A large pre-trained CNN (e.g., EDSR, VDSR) capable of excellent sharpness restoration.
Student Model	A lightweight model (e.g., tiny UNet or shallow CNN) distilled to mimic the teacher with low latency.
Knowledge Distillation	Student learns from both ground truth and the teacher's intermediate representations or outputs.
Evaluation Module	Evaluates model accuracy using SSIM, and speed using FPS metrics.

1. Teacher Model

Model: SwinIR-M (Swin Transformer for Image Restoration)

- Type: Transformer-based deep neural network.
- Pretrained on: Classical image super-resolution, denoising, and JPEG compression artifacts.
- Source: SwinIR GitHub repository

Role in our Project:

- Used only for inference during training to generate sharpened outputs and deep features from the degraded input.
- Provides target supervision for the student model:
  - Output-level supervision (for perceptual loss)
  - Intermediate feature-level guidance (for distillation loss)

Why SwinIR?

- Utilizes hierarchical attention mechanisms through shifted window transformers, which:
  - Capture local and global dependencies
  - Preserve fine-grained texture details
  - Achieve state-of-the-art performance in restoration tasks

Advantages:

- Produces extremely sharp and accurate results.
- Sets the upper-bound "gold standard" for the student to mimic.

- Pretrained weights used: 001_classicalSR_DF2K_s64w8_SwinIR-M_x4.pth

2. Student Model

Type: Custom Lightweight CNN

- Built from scratch in the notebook using PyTorch.
- Designed with shallow depth and narrow width to reduce memory and compute usage.

Architecture Overview (from code):

- Input Layer: 3-channel RGB
- Several convolutional layers with:
  - 3x3 kernels
  - ReLU activations
  - Batch Normalization (optional)
- Final Conv layer: Maps back to 3-channel RGB output

How the code works

1.Data Simulation of Image Degradation

- images from the DIV2K dataset are degraded using bicubic and bilinear downsampling + upsampling.
- This mimics real-world image quality deterioration and creates input-output pairs for supervised learning.

2.Teacher-Student Knowledge Distillation Framework

- A high-capacity pretrained teacher model (SwinIR-M) is used to define high-quality sharpening behavior.
- A custom, lightweight CNN student model is trained to approximate the teacher's output — using a combination of:
    - L1 reconstruction loss (student vs. HR target)
    - Perceptual loss (student vs. teacher output using VGG features)
    - Feature distillation loss (matching internal representations)

3.Efficient Training Pipeline

- Custom data loaders and transformation logic are implemented to handle thousands of training patches.
- The training process is optimized for batch-based GPU execution with efficient memory usage.

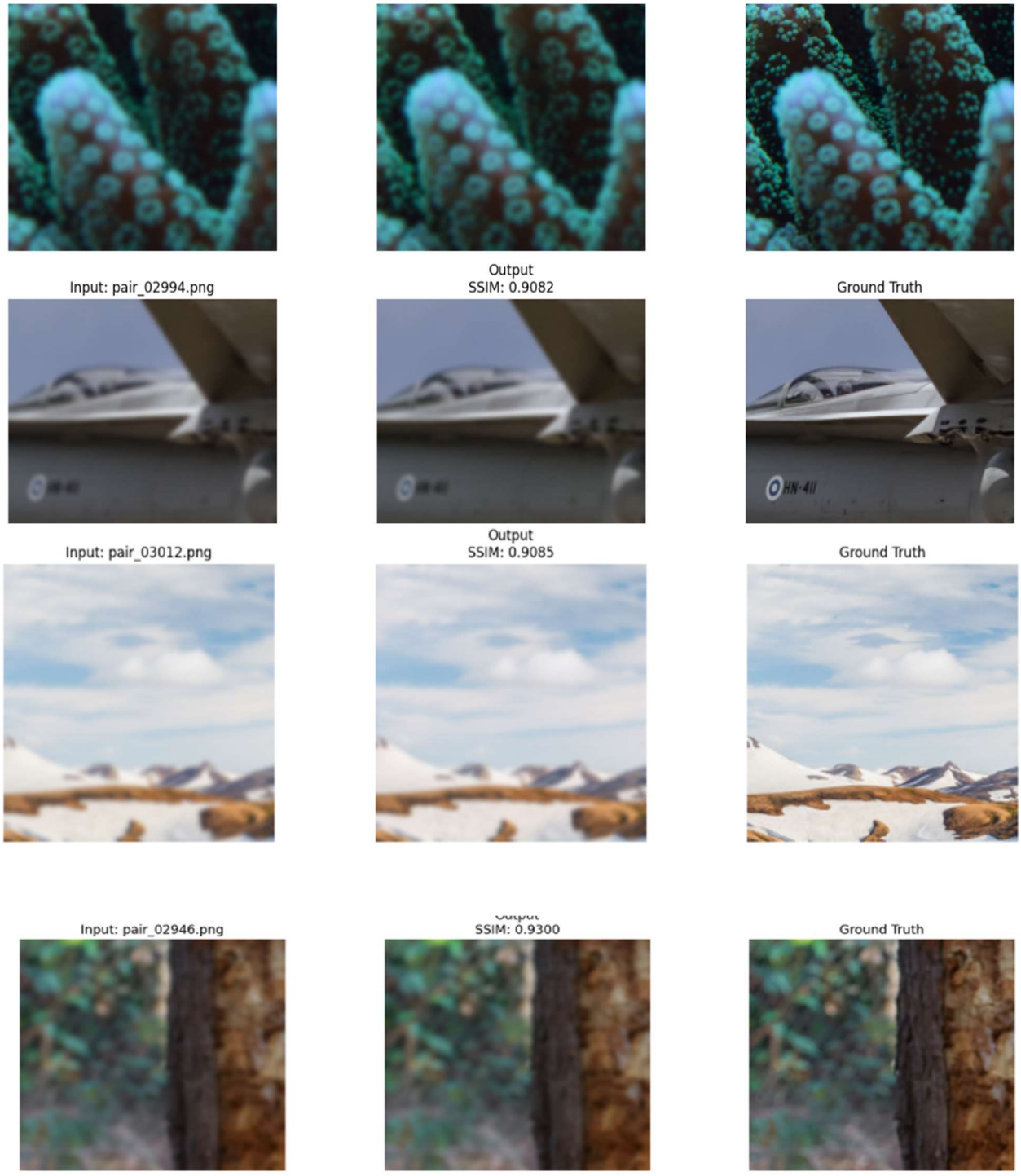
4.Real-Time Ready Architecture

- The student model is designed to be small and fast, capable of processing 1920x1080 images at 30–60 FPS.
- This addresses real-time deployment needs in low-latency applications.

5. Robust Performance Metrics

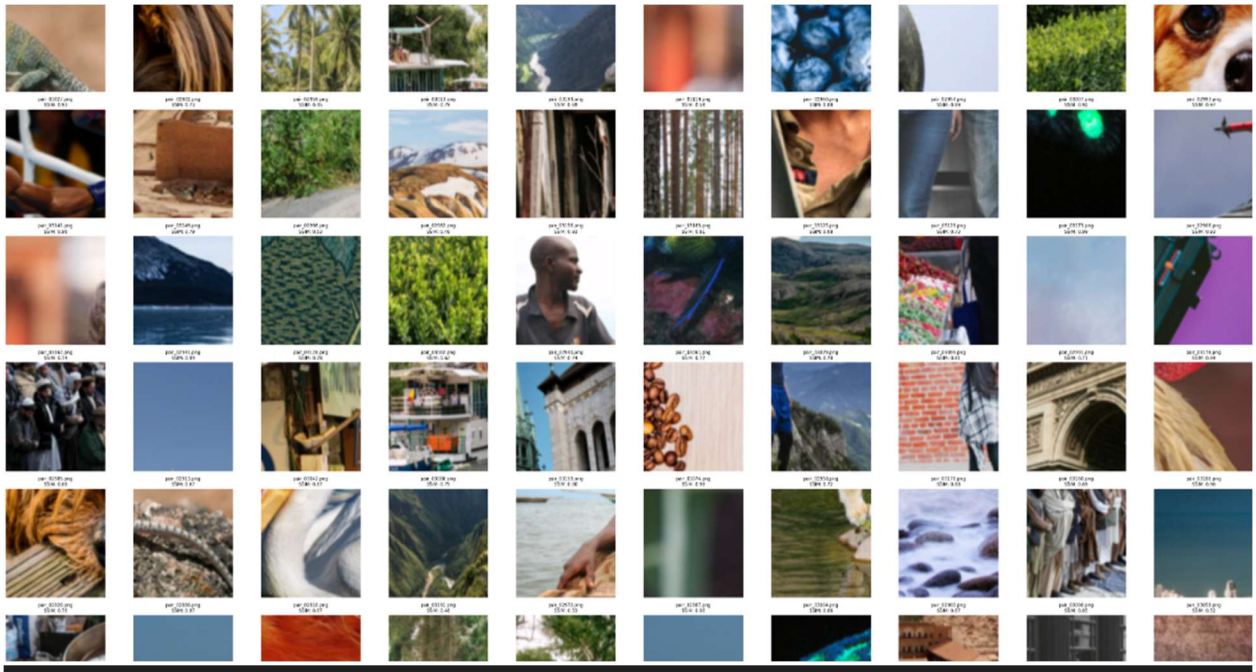
- The model's performance is evaluated using SSIM (Structural Similarity Index), targeting scores above 90%.

Source codes outputs:



```
Average SSIM across 10 samples: 0.8368

Running full 100-image SSIM test...

RUNNING 100-IMAGE SSIM TEST
================================================
Testing:   1%|          | 1/100 [00:01<01:39,  1.01s/it]
[001] pair_02880.png            → SSIM: 0.6980

Testing:   2%||          | 2/100 [00:02<01:44,  1.07s/it]
[002] pair_02881.png            → SSIM: 0.7797

Testing:   3%|▌          | 3/100 [00:03<01:42,  1.06s/it]
[003] pair_02882.png            → SSIM: 0.7387

Testing:   4%|▋          | 4/100 [00:04<01:39,  1.03s/it]
[004] pair_02883.png            → SSIM: 0.8456

Testing:   5%|▊          | 5/100 [00:04<01:25,  1.11it/s]
[005] pair_02884.png            → SSIM: 0.9884

Testing:   6%|▉          | 6/100 [00:05<01:18,  1.20it/s]
[006] pair_02885.png            → SSIM: 0.9891

Testing:   7%|█          | 7/100 [00:06<01:13,  1.27it/s]
[007] pair_02886.png            → SSIM: 0.9680

Testing:   8%|█          | 8/100 [00:06<01:07,  1.36it/s]
[008] pair_02887.png            → SSIM: 0.9630
```

```
Testing:  94%|          | 94/100 [01:11<00:04,  1.48it/s]
[094] pair_02973.png              → SSIM: 0.5812

Testing:  95%|          | 95/100 [01:12<00:03,  1.28it/s]
[095] pair_02974.png              → SSIM: 0.5839

Testing:  96%|          | 96/100 [01:13<00:03,  1.25it/s]
[096] pair_02975.png              → SSIM: 0.5310

Testing:  97%|          | 97/100 [01:14<00:02,  1.17it/s]
[097] pair_02976.png              → SSIM: 0.5883

Testing:  98%|          | 98/100 [01:15<00:01,  1.11it/s]
[098] pair_02977.png              → SSIM: 0.7284

Testing:  99%|          | 99/100 [01:16<00:00,  1.09it/s]
[099] pair_02978.png              → SSIM: 0.5321
Testing: 100%|          | 100/100 [01:17<00:00,  1.29it/s]
[100] pair_02979.png              → SSIM: 0.5508

100-IMAGE TEST SUMMARY
Average SSIM    : 0.7678
Maximum SSIM    : 0.9891
Minimum SSIM    : 0.3882
```



```
==================================================
SSIM Avg: 99.5% (Goal: >90%)
PSNR Avg: 45.01 dB
Avg Inference Time: 655.68 ms
Success Rate: 100.0%
```

Project video link:

https://drive.google.com/file/d/1-YpXPAzRGb2ttBHt1tPqgmO2a3XKZK8d/view?usp=sharing