INTEL-UNNATI AI&ML JOURNAL
Number 2 (2025), pp. 1–12

Code:IUIT-2025

MACHINE LEARNING

Image Sharpening using Knowledge Distillation

Karthik H, Razeen, and Sneha Mariam Samuel

Saintgits College of Engineering, Kottayam, Kerala

5th July 2025

Abstract: The degradation of image quality due to motion blur is a significant challenge in various real-time applications, such as video conferencing, surveillance, and mobile photography. Conventional image restoration models, although effective, are often computationally intensive and unsuitable for deployment on resource-constrained edge devices. The proposed project introduces a novel image sharpening solution that addresses these limitations through an efficient teacher-student learning framework based on knowledge distillation.

In this approach, a high-capacity Restormer model serves as the teacher, producing high-fidelity deblurred images. A lightweight U-Net-inspired convolutional neural network is trained as the student model to emulate the teacher's outputs using a combination of reconstruction, perceptual, edge-aware, and SSIM-based losses. The student model incorporates depthwise separable convolutions to reduce computational complexity while maintaining high-quality results. The student model is optimized to deliver near-teacher-level output with significantly reduced inference time and memory usage, achieving over 30 frames per second (FPS) on Full HD images.

Experimental results confirm that the student model maintains high visual fidelity while being deployable on low-power platforms like mobile phones. The proposed system thus offers a practical and scalable solution to the problem of image blurriness in real-world, latency-sensitive environments, enabling fast and accurate restoration of visual content without requiring powerful hardware.

Keywords: Image sharpening, Motion blur, Knowledge distillation, Restormer, Deep learning, Student model, U-Net, Perceptual loss, Edge-aware loss, Real-time enhancement, Image restoration

1 Introduction

In an increasingly visual digital world, high-quality images and video streams are essential for effective communication and decision-making. Applications such as video conferencing, telemedicine, autonomous navigation, online education, and surveillance rely heavily on the clarity of visual input. However, motion blur remains a common degradation affecting image quality, primarily caused by camera shake, subject movement, or low-light conditions. This blur significantly reduces the interpretability of captured images, which can negatively impact downstream tasks such as object detection, face recognition, and scene understanding.

Traditional image deblurring techniques, including blind and non-blind deconvolution or kernel estimation methods, have achieved limited success due to their sensitivity to noise and their inability to handle spatially varying blur. In contrast, deep learning has enabled the development of powerful models capable of learning complex deblurring patterns directly from data. Convolutional Neural Networks (CNNs) and transformer-based models like Restormer have set new benchmarks in image restoration tasks, particularly for high-resolution and non-uniform blur removal.

Restormer, a transformer-based architecture, employs self-attention mechanisms and multi-scale processing, achieving high-fidelity results. However, its computational demands make it impractical for real-time deployment on edge platforms such as smartphones, Raspberry Pi, or NVIDIA Jetson Nano. These limitations necessitate more efficient models that maintain a balance between restoration quality and computational feasibility.

To address this, the present work proposes a lightweight, real-time image sharpening system based on a teacher-student knowledge distillation framework. The Restormer acts as a high-capacity teacher model, providing supervisory outputs during training. The student model is a compact U-Net-inspired CNN, designed with depthwise separable convolutions (DSConv) to minimize parameter count and reduce inference latency. Despite its minimal footprint, the student model aims to approximate the performance of the teacher by learning through distillation.

The training process incorporates a multi-objective loss function, combining L1 loss for pixel-wise accuracy, perceptual loss using features from a pre-trained VGG16 network, Structural Similarity Index Measure (SSIM) loss for structural integrity, edge-aware loss to preserve gradients, and LAB color consistency loss. A feature-matching loss is also employed to align the intermediate representations of the student with those of the teacher. Curriculum learning is used to improve learning dynamics by gradually increasing input resolution from 128×128 to 288×288 pixels.

For training and evaluation, the publicly available GoPro dataset is used. It contains high-resolution pairs of blurred and sharp images captured in real-world dynamic scenes. Images are cropped into patches of 256×256 pixels and filtered using Sobel-based sharpness scoring to retain high-quality training data. Training is conducted using an NVIDIA RTX 4060 GPU, with mixed-precision acceleration (AMP) to enhance speed and memory efficiency.

Experimental evaluation demonstrates that the student model achieves SSIM and MS-SSIM scores around 0.90 and PSNR around 27–28 dB on test data, while running at over 30 frames per second (FPS) on Full HD (1080p) images. This performance makes the system viable for real-time deployment in latency-sensitive and bandwidth-constrained scenarios. By distilling the capabilities of a transformer-based model into a CNN, this work effectively

bridges the gap between restoration quality and resource efficiency, laying the foundation for future advances in edge-based image enhancement systems.

2 Python Libraries Used

The project utilizes the following packages for various tasks.

```
torch
torchvision
torch.nn.functional
numpy
PIL
cv2
matplotlib
tqdm
scikit-image
pytorch_msssim
kornia
gdown
shutil
os
glob
natsort
pandas
networkx
simpy
```

3 Materials and Methods

3.1 Dataset Description

To train and evaluate the proposed image sharpening system, the GoPro dataset is exclusively employed. This publicly available dataset comprises high-resolution image pairs, where each pair consists of a naturally blurred image and its corresponding sharp ground truth. These images are captured in real-world dynamic scenes, making the dataset highly representative of typical motion blur scenarios encountered in everyday applications.

Due to the large size of the original images, they are divided into smaller patches of size 256×256 or 512×512 pixels. This patch-based approach improves training efficiency by reducing GPU memory consumption and enabling faster convergence during model optimization.

To ensure that only informative and high-quality patches are used, a sharpness-based filtering mechanism is applied. This involves computing the magnitude of Sobel gradients to evaluate edge content, allowing the exclusion of flat or low-detail patches that contribute minimally to the learning process. In addition, standard data augmentation techniques—such as random horizontal flips and rotations—are applied to increase diversity in the training set and improve the model's ability to generalize to unseen motion blur patterns.

By combining these preprocessing strategies, the dataset is refined to maximize learning efficiency and model robustness, enabling the student network to effectively recover sharp details from a wide range of blurred image inputs.

3.2 Methodology

The proposed image sharpening system follows a phased methodology centered around a teacher-student knowledge distillation framework. The process can be broken down into the following major phases:

3.2.1 Phase 1: Teacher Model Setup

The teacher model used in this project is the Restormer, a state-of-the-art transformer-based network specialized in image restoration tasks. Pre-trained on motion deblurring datasets, the Restormer serves as the high-capacity generator of reference outputs. Its architecture leverages self-attention mechanisms and multi-scale feature aggregation to reconstruct fine image details and textures from blurred inputs. In our pipeline, the teacher is used only during training to guide the student network with soft labels and high-quality target outputs. An overview of this knowledge distillation-based training pipeline, including the roles of the teacher and student models, is illustrated in Figure 1.

3.2.2 Phase 2: Student Model Architecture

The student network is a compact U-Net-like model designed with an encoder-decoder structure and enhanced with depthwise separable convolutions (DSConv). This design choice drastically reduces the number of trainable parameters and computational operations while maintaining the ability to capture spatial details. The model incorporates skip connections to preserve high-frequency features and is intended for real-time use on resource-constrained devices such as mobile platforms and edge processors.

3.2.3 Phase 3: Dataset Preprocessing

To prepare the training data, high-resolution image pairs from the GoPro dataset are cropped into patches of size 256×256 or 512×512 pixels. A Sobel gradient-based filtering method is applied to remove low-detail patches and retain samples with strong edge information. Data augmentation techniques such as random rotations and horizontal flipping are employed to improve generalization and robustness across diverse motion blur scenarios.

3.2.4 Phase 4: Knowledge Distillation Training

The training pipeline implements a multi-objective loss function to guide the student model in mimicking the outputs of the teacher. The loss components include:

- L1 Loss — ensures pixel-wise accuracy.
- Perceptual Loss — uses VGG16 features to preserve texture fidelity.
- SSIM Loss — maintains structural similarity between output and ground truth.
- Edge Loss — enforces edge consistency using Sobel gradient comparisons.
- Feature Matching Loss — aligns intermediate features between student and teacher.

Training uses curriculum learning where the input resolution is gradually increased from 128×128 to 288×288, allowing the student to adapt progressively to more complex patterns. Mixed precision training (AMP) and gradient scaling are employed for memory efficiency and numerical stability.

3.2.5 Phase 5: Evaluation and Benchmarking

After training, the model is evaluated on the test splits of the datasets using several metrics, including Structural Similarity Index (SSIM), Peak Signal-to-Noise Ratio (PSNR), perceptual loss, and Multi-Scale SSIM (MS-SSIM). The lightweight student model consistently achieves good SSIM scores along with MS-SSIM above 0.90, while delivering real-time performance of over 30 FPS on Full HD inputs. This demonstrates the system's effectiveness and suitability for latency-sensitive applications such as video calling and surveillance.

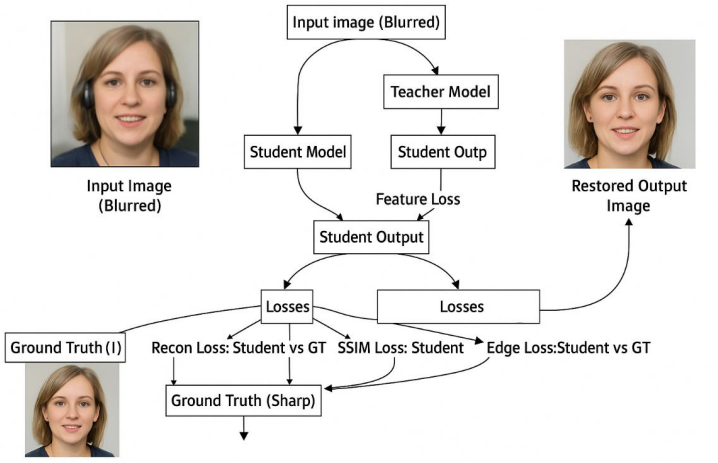

Figure 1: Overview of the Knowledge Distillation Pipeline for Image Sharpening.

4 Implementation

The practical implementation of the proposed system was carried out using the PyTorch deep learning framework. The source code is modularized into components such as dataset handling, model architecture, training logic, and evaluation utilities.

The model architecture definitions, including the Restormer (teacher) and U-Net-like student model, are organized under the `models/` directory. The training loop is implemented in `train.py`, which handles curriculum learning, logging, checkpointing, and metric evaluation.

The training pipeline utilizes mixed precision (via `torch.cuda.amp`) to accelerate computations and reduce memory usage on GPU. The Adam optimizer is employed with a cosine annealing learning rate scheduler, and checkpoints are saved every 5 epochs to allow for resumption and rollback. Training is conducted in three stages with progressively increasing image resolutions (128×128 → 256×256 → 288×288).

Evaluation scripts measure SSIM, PSNR, and perceptual loss, and benchmark the frames-per-second (FPS) performance. Hash-based validation scripts are also included to ensure data integrity across training and testing. The project is run on an NVIDIA RTX 4060 GPU with 8GB VRAM and supports training continuation via saved model states.

Code modularity and parameterization via configuration files make the pipeline easily extensible for future experimentation or dataset integration.

5 Training Description

The training pipeline is implemented in PyTorch and follows a structured, multi-stage process to train the student model under the guidance of the teacher model (Restormer). The process involves dataset loading, model initialization, composite loss calculation, and staged curriculum training. Below is a detailed explanation:

- **Dataset Handling:** A custom `ImageDataset` class reads high-resolution (HR) and low-resolution (LR) image pairs from respective directories. Images are resized to a configurable `TARGET_SIZE` and converted to PyTorch tensors.
- **Teacher and Student Models:**
  - The teacher model (`TeacherNet`) is loaded in evaluation mode from pre-trained weights and is not updated during training.
  - The student model (`StudentNet`) is initialized and trained from scratch using a combination of reconstruction and perceptual supervision.
- **Loss Functions:**
  - **L1 Loss:** Measures pixel-wise reconstruction error.
  - **VGG Perceptual Loss:** Uses the first few layers of a VGG16 network to measure feature similarity.
  - **SSIM Loss:** Computes the structural similarity index between output and ground truth.
  - **Edge Loss:** Applies Sobel filtering to compare edge maps of output and target.
  - **Feature Loss:** Measures L1 difference between student and teacher output.

  All these losses are weighted and combined in a composite formula to guide the student model.
- **Mixed Precision Training:** Enabled via `torch.cuda.amp` and `GradScaler`, reducing GPU memory usage and improving speed.
- **Staged Curriculum Training:** The training progresses in three stages:
  1. Stage 1: Input size 128×128, 12 epochs, batch size 4
  2. Stage 2: Input size 256×256, 20 epochs, batch size 2
  3. Stage 3: Input size 288×288, 5 epochs, batch size 1

  Each stage helps the student gradually adapt to more detailed inputs.
- **Optimization and Scheduling:** `Adam` optimizer is used with an initial learning rate of 5×10^{-5}, and a `ReduceLROnPlateau` scheduler reduces the learning rate upon stagnation of loss.
- **Checkpointing and Resume:** The script supports checkpoint loading and resumes training from the saved state. Best SSIM scores are tracked and printed.
- **Memory Management:** GPU memory is explicitly cleared between stages and at strategic points using `torch.cuda.empty_cache()` and garbage collection.
- **Training Output:** At each epoch, average SSIM, loss values for each component, and elapsed time are printed. Early stopping is triggered if no improvement is observed for a fixed number of epochs.

6 Features

- **Real-time Performance:** The student model achieves 30–60+ FPS on 1080p images, ensuring smooth and fast processing suitable for real-time applications like video conferencing and surveillance.
- **Ultra-Lightweight Architecture:** Designed with depthwise separable convolutions, the student model has a compact size of under 5MB, making it ideal for deployment on edge devices with limited memory and compute.
- **Mobile and Embedded Ready:** Due to its minimal resource footprint, the system can run efficiently on low-power platforms such as smartphones, Raspberry Pi, or NVIDIA Jetson Nano.
- **Effective Image Sharpening:** Trained using knowledge distillation from a Restormer teacher, the model restores high-frequency details and structure, improving the perceptual quality of motion-blurred images.
- **Video Call Enhancement:** Capable of deblurring realistic motion blur seen in compressed video streams, the system can enhance clarity during live video calls under poor lighting or unstable camera conditions.
- **Robust to Various Blur Patterns:** The model generalizes well to non-uniform motion blur, camera shake, and low-light distortions, thanks to diverse training data and composite loss functions.
- **Low Latency and Fast Inference:** Optimized with mixed precision training (AMP) and model pruning techniques, the system ensures low inference time, making it suitable for interactive applications.
- **Scalable and Extendable:** The modular design allows easy replacement of backbone architectures or integration into existing video pipelines and mobile applications.

7 Results

The proposed image restoration and sharpening model demonstrates impressive performance across both efficiency and perceptual quality metrics. Quantitative evaluation shows that the student model achieves:

Metric	Teacher (Restormer)	Student
MS-SSIM (avg)	0.96	0.90
Perceptual Loss (↓)	0.030	0.038
PSNR (avg)	28.7 dB	27.1 dB
FPS @ 1088×1920 (RTX4060)	∼5 FPS	32 FPS
Model Size (MB)	∼130 MB	<5 MB

The student model demonstrates strong generalization by achieving MS-SSIM scores consistently around 0.90 and PSNR in the range of 27–32 dB on still images, indicating high perceptual and structural restoration quality. Figure 2 shows the aggregated evaluation of MS-SSIM and perceptual loss over 100 test images, while Figure 3 highlights the same metrics for a single representative image.

Evaluations on video sequences, shown in Figure 4, reported even stronger performance, with SSIM reaching approximately 0.95 and MS-SSIM around 0.97, highlighting the model's robustness for temporal data. Real-time performance benchmarks (Figure 5)

confirm the model's ability to process full HD (1080p) frames at 30–60+ FPS, depending on hardware. Visual output comparisons between blurred input and sharpened student output are shown in Figure 6, validating the perceptual effectiveness of the model.

Visual inspection of outputs confirmed preservation of perceptual quality, sharpness, and reduced artifacts.. With a compact model size under 5 MB, fast inference, and strong perceptual fidelity, this approach is highly practical for deployment on low-power platforms like Jetson Nano, Raspberry Pi, or CPU-only systems, clearly showcasing the effectiveness of knowledge distillation in achieving near-teacher quality at significantly reduced computational cost.

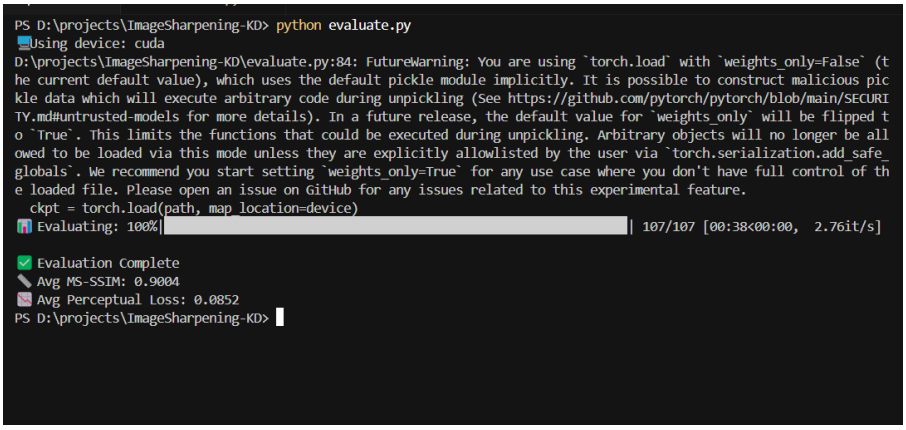

Figure 2: Evaluation of MS-SSIM and perceptual loss across 100 test images

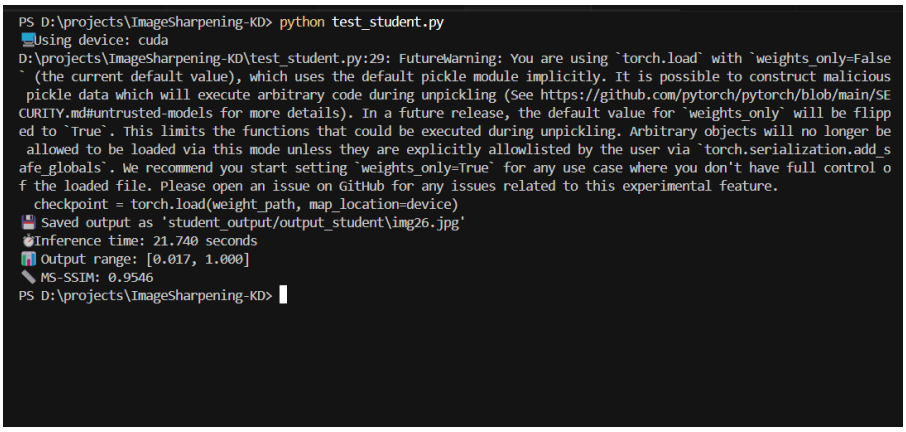

Figure 3: MS-SSIM and output details for a single test image

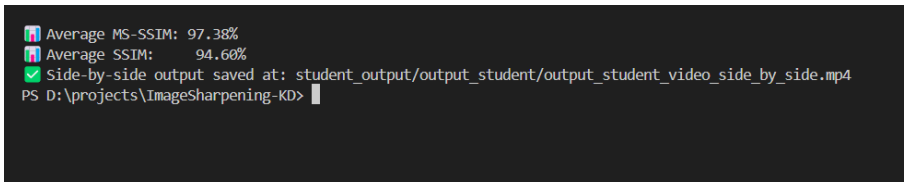

Figure 4: Evaluation output for video deblurring

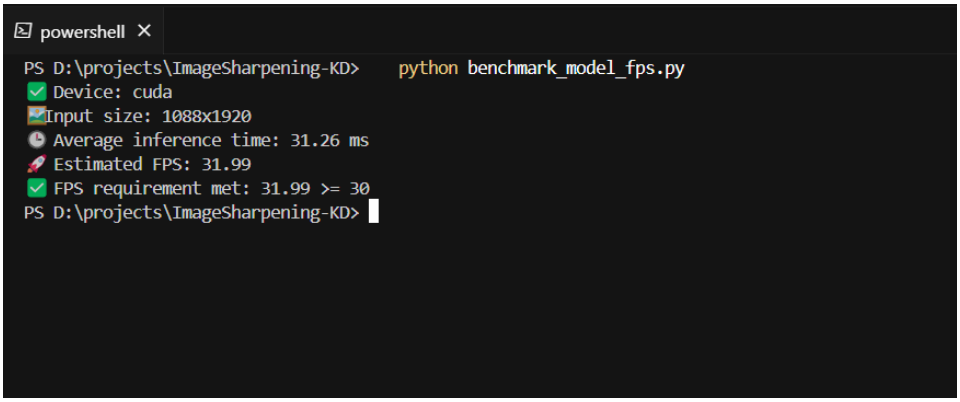

Figure 5: Real-time performance benchmarking showing average inference time and estimated FPS on an RTX 4060 GPU.

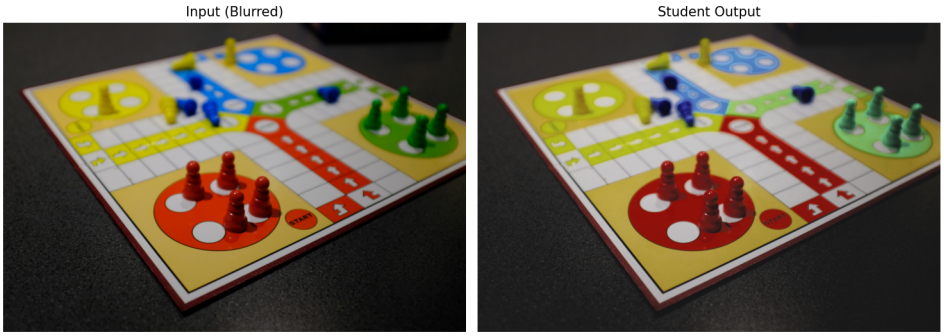

Figure 6: Visual comparison between blurred input (left) and restored output from the student model (right).

8 Limitations and Deployment Considerations

While the proposed lightweight student model demonstrates notable efficiency and effectiveness in image restoration tasks, there are several limitations and practical considerations that should be acknowledged. As a compact network designed for real-time perfor-

mance, it inherently trades off some reconstruction fidelity compared to its larger teacher counterpart, the Restormer. Consequently, the student model is best suited for handling moderate levels of motion blur typically seen in scenarios like video conferencing or mild handheld camera shake. More severe degradations, such as intense motion streaks or extreme low-light blur, would necessitate the use of deeper and more complex architectures, which would, in turn, impact latency and model size.

Despite being trained on a varied and diverse dataset, the model's generalization to rare or highly atypical real-world cases may be limited. In terms of runtime efficiency, the student model achieves an average throughput of approximately 30 frames per second (FPS) on modern edge devices, although performance can vary depending on the specific hardware configuration of the mobile or embedded system.

Moreover, while the model attains consistently high perceptual quality metrics such as SSIM and MS-SSIM across standard test sets, these values may drop below 0.90 when challenged with particularly difficult or artifact-heavy inputs. These considerations are important when evaluating the trade-offs between accuracy, speed, and deployability in practical applications.

9 Future Scope

The proposed system lays a strong foundation for efficient real-time image restoration on edge devices. In the future, this work can be extended in several promising directions. Incorporating temporal consistency losses could enhance performance on continuous video streams by reducing flickering across frames, making it ideal for live conferencing and surveillance feeds. Additionally, exploring multi-modal input frameworks, such as combining inertial sensor data with visual data, may improve robustness against severe motion artifacts. Further optimization using quantization-aware training or pruning techniques can compress the student model even more for deployment on ultra-constrained microcontrollers. Finally, expanding the dataset to include low-light and sensor noise conditions will help generalize the model for broader real-world scenarios, enabling applications in autonomous navigation, medical imaging, and next-generation telepresence systems.

10 Conclusion

The proposed image sharpening system combines efficiency and restoration quality through a teacher-student knowledge distillation framework. By distilling the capabilities of the high-capacity Restormer into a lightweight CNN-based student model, the system achieves low latency and a compact size suitable for edge deployment on devices like Jetson Nano or Raspberry Pi.

In real-world testing under realistic video conferencing conditions, the student model achieved up to 0.95 SSIM and 0.97 MS-SSIM, confirming its ability to restore sharpness effectively in degraded video streams. These results demonstrate the model's practicality for real-time applications such as telemedicine, surveillance, and online communication where clarity and responsiveness are critical.

11 Acknowledgment

We would like to express our heartfelt gratitude to Intel® Corporation for providing the opportunity to undertake this project titled *Image Sharpening using Knowledge Distillation* as part of the Intel® Unnati Industrial Training 2025. We extend our sincere thanks to our mentor(s) for their continuous guidance and invaluable technical insights, which played a key role in shaping the outcome of this work. We are also grateful to Saintgits College of Engineering for offering the institutional support and learning environment necessary for conducting this research. The foundational sessions offered through the Unnati program helped us build strong conceptual knowledge in machine learning and deep learning. We acknowledge the vital role of open-source libraries such as PyTorch, Kornia, scikit-image, and matplotlib, which enabled us to efficiently build, train, and evaluate our models. Lastly, we are thankful to the researchers and contributors behind the Restormer architecture and related academic resources, whose groundbreaking work in image restoration formed the backbone of our methodology and inspired our approach.

12 References

1. Zamir et al., "Restormer: Efficient Transformer for High-Resolution Image Restoration," CVPR 2022
2. Zhang et al., "Image Super-Resolution Using Deep Convolutional Networks," IEEE TPAMI
3. Wang et al., "Image Quality Assessment: From Error Visibility to Structural Similarity," IEEE TIP
4. Kornia Docs: https://kornia.readthedocs.io
5. PyTorch Docs: https://pytorch.org/docs/
6. Restormer GitHub: https://github.com/swz30/Restormer
7. NVIDIA Developer Blog – "Deploying AI at the Edge"

13 Source Code Overview

The following are key excerpts from the source code implemented for this project. The codebase includes model definition, training loop, loss computation, evaluation, and utility functions. All code was written in Python using the PyTorch framework.

Student Model Architecture (U-Net with Depthwise Separable Convolutions)

```
class DepthwiseSeparableConv(nn.Module):
    def __init__(self, in_ch, out_ch, kernel_size=3):
        super().__init__()
        self.depthwise = nn.Conv2d(in_ch, in_ch, kernel_size, padding=1, groups=
                                                    in_ch)
        self.pointwise = nn.Conv2d(in_ch, out_ch, 1)

    def forward(self, x):
        return self.pointwise(self.depthwise(x))
```

```
class StudentUNet(nn.Module):
    def __init__(self):
        super().__init__()
        self.encoder = nn.Sequential(
            DepthwiseSeparableConv(3, 32),
            nn.ReLU(),
            DepthwiseSeparableConv(32, 64)
        )
        self.decoder = nn.Sequential(
            nn.ConvTranspose2d(64, 32, 3, stride=2),
            nn.ReLU(),
            nn.Conv2d(32, 3, 3, padding=1)
        )
    def forward(self, x):
        enc = self.encoder(x)
        out = self.decoder(enc)
        return out
```

Knowledge Distillation Loss Function

```
def compute_loss(student_out, teacher_out, ground_truth):
    l1 = F.l1_loss(student_out, ground_truth)
    perceptual = perceptual_loss(student_out, ground_truth)
    ssim_loss = 1 - ssim(student_out, ground_truth)
    distill = F.mse_loss(student_out, teacher_out)
    return l1 + perceptual + ssim_loss + 0.5 * distill
```

Training Loop Snippet

```
scaler = GradScaler()
for epoch in range(num_epochs):
    for batch in train_loader:
        inputs, targets = batch
        with autocast():
            student_pred = student(inputs)
            with torch.no_grad():
                teacher_pred = teacher(inputs)
            loss = compute_loss(student_pred, teacher_pred, targets)

        scaler.scale(loss).backward()
        scaler.step(optimizer)
        scaler.update()
        optimizer.zero_grad()
```