IMAGE SHARPENING USING KNOWLEDGE DISTILLATION

1. Introduction

Video conferencing platforms often face a significant drop in visual quality due to poor internet bandwidth and compression artifacts. To address this, the project focuses on developing a deep learning-based image sharpening model that enhances image clarity in real-time. The method involves using Knowledge Distillation (KD) to train a compact student model under the supervision of a high-performing teacher model. The goal is to balance high accuracy and low computational cost for real-time deployment.

2. Objectives

- Develop a compact deep learning model that sharpens video frames in real-time (30–60 FPS).
- Employ knowledge distillation to train a student model using a larger, more accurate teacher model.
- Simulate degraded images using bicubic downscaling and upscaling.
- Evaluate performance using SSIM and visual quality (MOS).
- Demonstrate the effectiveness through real-time webcam sharpening.

3. Tools and Technologies

- Programming Language: Python
- Frameworks & Libraries: PyTorch, TorchVision, OpenCV, scikit-image
- Development Environment: Visual Studio Code
- Dataset Used: CIFAR-10 (for training), webcam input (for inference)

4. Data Sources

- CIFAR-10 Dataset: Used as a proxy for low-resolution image sharpening tasks. Each 32x32 image was degraded and restored.
- Live Webcam Feed: Used to test the model in real-world scenarios.

5. Model Architectures

5.1 Teacher Model

The TeacherSharpenNet is a moderately deep convolutional network:

```
nn.Sequential(
  nn.Conv2d(3, 64, kernel_size=3, padding=1),
  nn.ReLU(),
  nn.Conv2d(64, 64, kernel_size=3, padding=1),
  nn.ReLU(),
  nn.Conv2d(64, 32, kernel_size=3, padding=1),
  nn.ReLU(),
  nn.Conv2d(32, 3, kernel_size=1)
)
```

It is trained using MSE loss between the sharpened output and the ground truth high-resolution image.

5.2 Student Model

The student model is a lightweight CNN optimized for real-time processing:

```
nn.Sequential(
  nn.Conv2d(3, 16, kernel_size=3, padding=1),
  nn.ReLU(),
  nn.Conv2d(16, 3, kernel_size=1)
)
```

The student is trained using a combination of:

Loss = α * MSE(Student, Teacher) + β * MSE(Student, GroundTruth)

6. Performance Analysis Process

6.1 Image Degradation Pipeline

- Downscale each input image to 16x16
- Upscale it back to 32x32 using bilinear interpolation
- Use the upscaled image as degraded input to the model

6.2 Metrics Used

- SSIM (Structural Similarity Index): To compare restored vs. ground truth images

- FPS (Frames per Second): Measured real-time processing speed
- MOS (Optional): Subjective user feedback (future scope)

6.3 Evaluation Method

- Test on 10,000 CIFAR-10 test images
- Real-time evaluation using OpenCV webcam pipeline

7. Results

- SSIM Performance:
    - Student model: ~91.3%
    - Teacher model: ~93.6%
- FPS Performance:
    - Student model: 35–40 FPS (real-time)
    - Teacher model: 10–15 FPS
- Qualitative Output:
    - Significant improvement in edge definition and clarity compared to degraded input

8. Source Code Summary

File: model/teacher_model.py

Defines the architecture for the teacher model.

File: model/student_model.py

Defines the student model for fast inference.

File: train.py

Handles the training of both teacher and student models with MSE and KD losses.

File: evaluate.py

Computes SSIM scores on the test dataset.

File: realtime_sharpen.py

Uses webcam to capture live video, degrades it, applies the student model, and displays results.

File: realtime_teacher_sharpen.py

Same as above but uses the teacher model for output.

9. Conclusion

The developed system demonstrates that knowledge distillation enables a compact model to approximate a complex model's performance while running efficiently in real time. With strong SSIM scores and frame rates above 30 FPS, the student model is suitable for enhancing video feeds in conferencing systems.

10. Future Work

- Replace CIFAR-10 with high-res datasets like TinyImageNet or DIV2K
- Export student model to ONNX for deployment
- Add GUI-based interface for user-controlled sharpening
- Integrate MOS collection through human feedback forms

11. References

- PyTorch Documentation: https://pytorch.org/
- scikit-image SSIM: https://scikit-image.org/docs/stable/
- OpenCV: https://opencv.org/
- Hinton et al., Knowledge Distillation Paper (2015)
- CIFAR-10 Dataset: https://www.cs.toronto.edu/~kriz/cifar.html

TEAM MEMBERS:

1.JAINA SHAJIL 2363035

2.DRISHYA HARISH 2363025

3.MOHAMMED FAJIS CK 2363071