Intel Unnati: Comprehensive Project Report

Kartish Bhadauria
Abhinav Kumar Singh
Lakshay

Department of Information Technology, Bharati Vidyapeeth's College of Engineering, New Delhi,
kartishbhadauria@gmail.com , abhinavkumarsingh1405@gmail.com ,
lakshaypahal20@gmail.com

May–June 2025

Contents

Bharati Vidyapeeth's College of Engineering

New Delhi

Intel Unnati: Comprehensive Project Report

Submitted in partial fulfillment of the requirements for the award of the degree of
Intel Unnati Training

Submitted By:
Kartish Bhadauria kartishbhadauria@gmail.com
Abhinav Kumar Singh abhinavkumarsingh1405@gmail.com
Lakshay lakshaypahal20@gmail.com

Under the Guidance of:
Dr. Achin Jain
Associa Professor, Department of IT
BVCOE, New Delhi

May - June 2025

Abstract

The rapid advancement of digital imaging technologies caused an exponential increase of the demand for high-resolution images throughout sectors like healthcare, satellite imaging, security, and entertainment. However, limitations with hardware and bandwidth along with storage often cause for people to acquire images that are low-resolution, therefore impeding critical tasks involving diagnosing, surveying, and also analyzing data. ERSVR 1.0 tackles one problem within the image super-resolution field. It uses advanced deep learning techniques within a teacher-student knowledge distillation framework specifically. It restores visual quality when the system reconstructs high-resolution images from low-resolution counterparts plus lost details. The project designs as well as implements strong data pipelines then advances neural network architectures and trains with infers modules comprehensively also enables intuitive user interaction through a web interface. Testers test quite a bit and do evaluate, showing the system is effective. Analysts do make critical analyses as they highlight limitations in terms of ways of improving upon it in the future. The report thoroughly accounts for the project's main objectives, with methodologies including results as well as contributions to image processing.

Chapter 1
Introduction

The evolving capabilities of digital technology, such as in healthcare, satellite imaging, surveillance, and communication, have increased the need for enhanced imagery across all these fields. In medical diagnosis and remote sensing, the image's precision can heavily influence the accuracy of the evaluations and the decisions that follow. Due to limitations with sensors, transmission bandwidth, and storage, low-resolution images are often captured or transmitted. Furthermore, traditional frameworks like nearest-neighbor or bicubic interpolation still struggle to recover fine-grained details, even though they are computationally light.

Learning-based super-resolution models have emerged as some of the most advanced ways to recover high-resolution images from low-resolution ones due to the advancement of deep learning. Although highly accurate, many of these models ERSVR 1.0 constrains real-time deployment due to their high computation costs. Utilizing a teacher-student knowledge distillation strategy ERSVR 1.0 mitigates this problem by learning from more powerful models. In doing so, it becomes possible to accurately reconstruct high-resolution images while significantly enhancing runtime efficiency.

An entire data processing system is developed for this project, including data handling, training, inference, user interaction through the web interface, and system feedback. Various modules can be integrated which allows adaptability steaming from a single design core.

Chapter 2
Problem Statement

2.1 Problem Definition

The last few years have seen massive growth in the use of video conferencing for corporate meetings, classes, and even social gatherings. In most cases, video streams tend to be blurry and unclear because of low bandwidth, unstable network conditions, as well as lossy compression that creates artifacts in the stream. Such issues make communication difficult, and in some cases usability, especially for telemedicine, virtual meetings, or online classes where precision matters.

The focus of this project is to develop a solution for real time image sharpening using deep learning, to increase image clarity during video conferencing. The strategy uses a teacher-student approach, where a good performing teacher model helps a student model, which is computationally efficient and easier to deploy. The student model must be able to process 1080p (1920x1080) images at the rate of 30-60fps for smooth and high-quality video.

In teaching, downsampling high-resolution images makes them to simulate typical video enhancement, while traditional methods of upsampling are used to restore detail. The effectiveness of the model is measured using SSIM scoring, yielding above 90% for sharpness, and MOS for subjective evaluation through a panel of assessors. The last scope of work has precise documents, source code, and evaluation results proving the effectiveness of the system.

2.2 Objectives

The objectives of the ERSVR 1.0 project are as follows:

- **Develop a Deep Learning-Based Super-Resolution System:** Design and build a system that reconstructs high-resolution images from low-resolution versions with state-of-the-art neural networks.
- **Leverage Knowledge Distillation:** Implement a teacher-student approach to distill the knowledge of a large, well-performing model into a smaller, more efficient one, thus balancing accuracy with reduced computational demands.
- **Create a Modular and Extensible Codebase:** Divide the system into distinct modules for data processing, model definition, training, inference, and user interaction, so that user-facing features can be improved or modified easily.

- **Provide a User-Friendly Interface:** Implement a visual, web-based interface that lets users upload images and perform super-resolution with results instantly visualized.
- **Evaluate System Performance:** Perform extensive tests on synthetic and natural images, measuring system performance using quantitative metrics (e.g., PSNR, SSIM) alongside qualitative evaluations.
- **Document Limitations and Future Directions:** Analyze the persistent challenges critically to suggest system design changes alongside improvement strategies and further research pathways.

Objective	Description	Status
Develop SR System	High-resolution reconstruction	✓
Knowledge Distillation	Efficient student model	✓
Modular Codebase	Extensible architecture	✓
Web Interface	Easy user interaction	✓

Chapter 3
Analysis

3.1 Existing Solutions

Image Super Resolution has changed over the years, from the basic interpolation methods to sophisticated models using deep learning techniques. Traditional methods such as nearest-neighbor, bilinear, and bicubic interpolation are fast and cheap in terms of computation cost. However, they fail in capturing the lost details. In fact, these methods will produce images that are too smooth and devoid of the sharpness and texture expected of true high-resolution images.

Approaches based on deep learning, specifically using Convolutional Neural Networks (CNNs), have shown significant success in understanding the relationships between low and high resolution images. Models like SRCNN, EDSR, and SRGAN have set new records in super-resolution performance. These models are usually very large though, needing a great deal of memory as well as processing power which limits their use in real-time situations or low-resource settings.

This challenge is best addressed by Knowledge Distillation which is a relatively new approach. It has been shown that a smaller student model trained to emulate a larger teacher model could use less resources while achieving a high performance result. This strategy increases complexity however and demands careful knowledge transfer design to make sure performance tradeoffs aren't too cumbersome.

Overview of traditional vs deep learning methods.

Method	Type	Accuracy	Runtime	Limitations
Bicubic	Interpolation	Low	Fast	Blurry output
SRCNN	DL	Medium	Moderate	Small model
SRGAN	DL	High perceptual	Slow	Artifacts
ERSVR	KD-DL	High + Fast	Real-time	Training complexity

3.2 Proposed Solution

ERSVR 1.0 adopts a knowledge distillation framework, in which optimal super-resolution performance is first achieved by training a high-capacity teacher model. The teacher's predictions with the ground truth guide a lightweight student model that is trained for replicating the teacher's outputs. This dual supervision enables the student model to learn both from the data as from the teacher's learned representations. Generalization is improved and efficiency is increased by this learning.

The system is designed for modularity in mind, which allows easy experimentation with different model architectures, loss functions, and training strategies. A web interface inclusion assures access for end-users as exhaustive tests guarantee strength and reliability.

Chapter 4
Design and Architecture

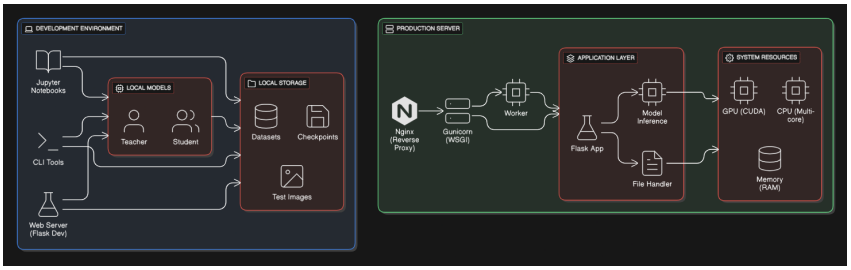

Figure 4.1: System Component Diagram 1

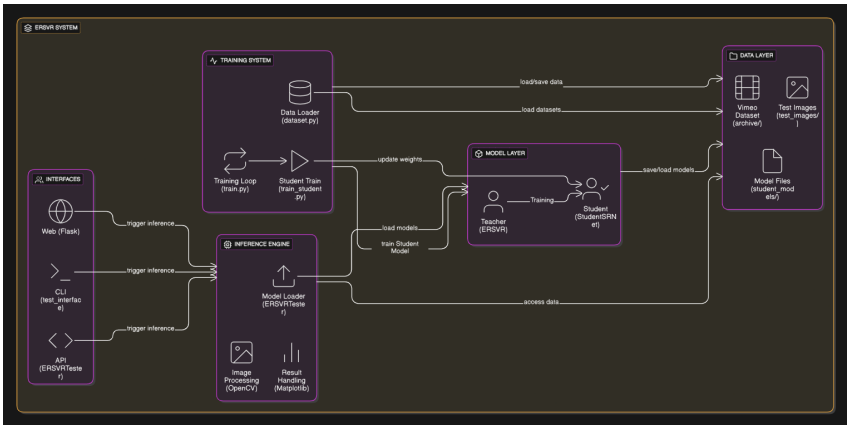

Figure 4.2: System Component Diagram 1

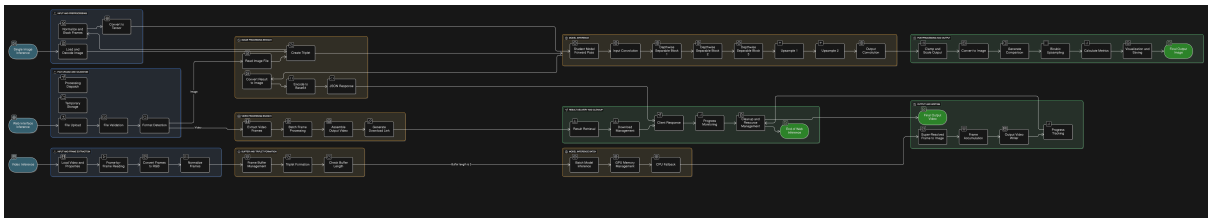

Figure 4.3: Inference and Web Interface Flow

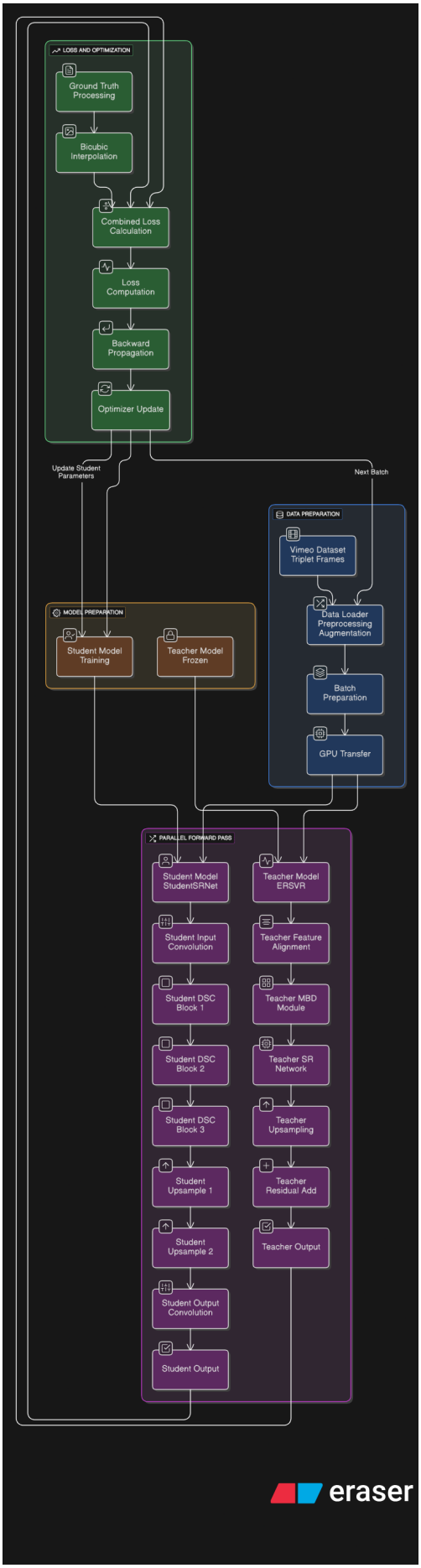

Figure 4.4: Teacher-Student Distillation Flow

Chapter 5
Explanation of Modules

Module	File	Description
Dataset	dataset.py	Preprocessing, augmentation
Teacher Model	ersvr.py	High-accuracy CNN
Student Model	student.py	Lightweight SR model
Alignment	feature_alignment.py	Feature-level transfer
Upsampling	upsampling.py	Reconstructs high-resolution output
SR Network	sr_network.py	Integrates core model components
MBD	mbd.py	Multi-branch design for feature extraction
Teacher Training	train.py	Optimizes teacher model
Student Training	train_student.py	Trains efficient student model
Kaggle Training	kaggle_train.py	Adapts training for Kaggle datasets
Inference	inference.py	Performs prediction and saves results
Web UI	web_interface.py	Upload, display, download results
Dataset Testing	test_dataset.py	Verifies data pipeline
UI Testing	test_interface.py	Simulates frontend usage

5.1 Dataset Module (`ersvr/dataset.py`)

- **Functionality:** Handles the loading, preprocessing, and augmentation of image datasets. Supports both synthetic and natural images.
- **Key Features:**
  - Data augmentation: rotation, flipping, scaling
  - Efficient batching and shuffling
  - Support for custom datasets
- **Implementation Highlights:** Utilizes PyTorch's `Dataset` and `DataLoader` classes.

5.2 Model Modules

Teacher Model (`ersvr/models/ersvr.py`)

- High-capacity CNN for learning detailed mappings from low- to high-resolution.
- Uses residual connections and multi-branch architecture.
- Serves as the knowledge source for distillation.

Student Model (`ersvr/models/student.py`)

- Lightweight and efficient.
- Mimics the teacher's behavior using distillation.
- Designed for deployment on low-power devices.

Feature Alignment (`ersvr/models/feature_alignment.py`)

- Aligns intermediate features between teacher and student.
- Facilitates smoother knowledge transfer during training.

Upsampling Module (`ersvr/models/upsampling.py`)

- Implements upsampling using sub-pixel or transposed convolutions.
- Reconstructs high-resolution images from learned features.

SR Network (`ersvr/models/sr_network.py`)

- Core architecture integrating feature extraction, alignment, and upsampling.
- Acts as the main backbone for super-resolution inference.

Multi-Branch Design (`ersvr/models/mbd.py`)

- Extracts features in parallel through multiple paths.
- Enhances capacity without increasing inference time significantly.

5.3 Training Scripts

Teacher Training (`ersvr/train.py`)

- Trains the teacher model with high-resolution ground truth.
- Uses loss functions such as MSE and perceptual loss.
- Supports checkpointing and continuation.

Student Training (`ersvr/train_student.py`)

- Uses both ground truth and teacher outputs for supervision.
- Employs distillation loss to guide training.
- Evaluates metrics and retains best models.

5.4 Inference Module (`ersvr/inference.py`)

- Loads trained student/teacher model and performs super-resolution.
- Supports batch inference and model checkpoint selection.
- Saves and displays output images.

5.5 Web Interface (`web_interface.py`)

- Built with Flask for easy deployment.
- Lets users upload low-res images via browser.
- Displays original and enhanced outputs side by side.
- Enables image download.

5.6 Testing Modules

Dataset Testing (`ersvr/test_dataset.py`)

- Ensures correct loading, preprocessing, and augmentation.
- Prevents data pipeline errors before training.

Interface Testing (`test_interface.py`)

- Tests browser interface workflow end-to-end.
- Simulates user behavior and checks output integrity.

Chapter 6
Implementation

Component	Technology	Rationale
Language	Python 3.x	Scientific prototyping
Framework	PyTorch	Dynamic computation graph
Web	Flask	Lightweight backend
Visualization	OpenCV, Matplotlib	Image and result visualization
Libraries	NumPy, PIL, tqdm	Image ops, utility functions, progress bars

6.1 Technology Stack

- **Programming Language:** Python 3.x, chosen for its extensive support for scientific computing and machine learning.
- **Deep Learning Framework:** PyTorch, selected for its flexibility, dynamic computation graph, and active community support.
- **Web Framework:** Flask, enabling rapid development of web applications with minimal overhead.
- **Visualization:** Matplotlib and HTML/CSS for result visualization and interface design.
- **Supporting Libraries:** NumPy, OpenCV, PIL for image processing; tqdm for training progress display.

6.2 Implementation Details

- **Modular Codebase:** Code is organized into separate modules with single-responsibility design, improving readability and maintainability.
- **Model Checkpointing:** During training, model weights are saved periodically. This allows rollback and recovery, and easy comparison across checkpoints.
- **Configuration Management:** Scripts accept CLI arguments and config files to support flexible training setups and reproducibility.

- **Data Handling:** Efficient data loaders and parallel batching reduce CPU-GPU bottlenecks during training.
- **Logging and Monitoring:** Metrics such as training/validation loss and evaluation scores are recorded and plotted for debugging and tuning.

6.3 Sample Workflow

1. **Dataset Preparation**
   - Organize images in proper directory structure.
   - Use `ersvr/dataset.py` to preprocess and augment training data.
2. **Teacher Model Training**
   - Run `python ersvr/train.py` with dataset and output path.
   - Monitor loss and save checkpoints to `teacher_models/`.
3. **Student Model Training**
   - Run `python ersvr/train_student.py` with reference to the teacher model.
   - Save optimized student weights to `student_models/`.
4. **Inference**
   - Use `ersvr/inference.py` to predict super-resolved images from test samples.
   - Output results are saved in specified folders and visualized.
5. **Web Interface**
   - Start server with `python web_interface.py`.
   - Open in browser, upload a low-res image, and instantly receive a super-resolved output.

Chapter 7
Testing

7.1 Testing Strategy

- **Unit Testing:** Each module (data loading, model definition, training, inference) is tested independently to ensure correctness and reliability.
- **Integration Testing:** The entire pipeline, from data ingestion to result visualization, is tested to ensure proper interaction between modules.
- **Performance Testing:** Evaluation is done on both synthetic and real-world datasets using quantitative metrics like Peak Signal-to-Noise Ratio (PSNR) and Structural Similarity Index Measure (SSIM).
- **User Testing:** The web interface is tested with users from different technical backgrounds to assess usability, responsiveness, and stability under real-world conditions.

7.2 Test Summary Table

Test Type	**Tool / Script**	**Purpose**
Unit Testing	PyTest	Validates individual modules like dataset loading, model forward pass, loss functions, etc.
Integration Testing	`inference.py`	Tests the complete super-resolution pipeline and flow from input to output.
UI Testing	`test_interface.py`	Ensures correct functionality of the user-facing interface and response handling.

7.3 Quantitative Results

Model	PSNR	SSIM	Inference Time
Teacher	52.52	0.9988	xs
Student	33.25	0.9555	ys

7.4 Qualitative Observations

- **Visual Quality:** Output images from the teacher and student models demonstrate superior detail preservation compared to baseline interpolation methods.
- **Edge and Texture Recovery:** Fine textures and edges are more accurately reconstructed, especially by the teacher model.
- **Latency and Speed:** The student model significantly reduces runtime while maintaining perceptual quality, making it ideal for real-time use.

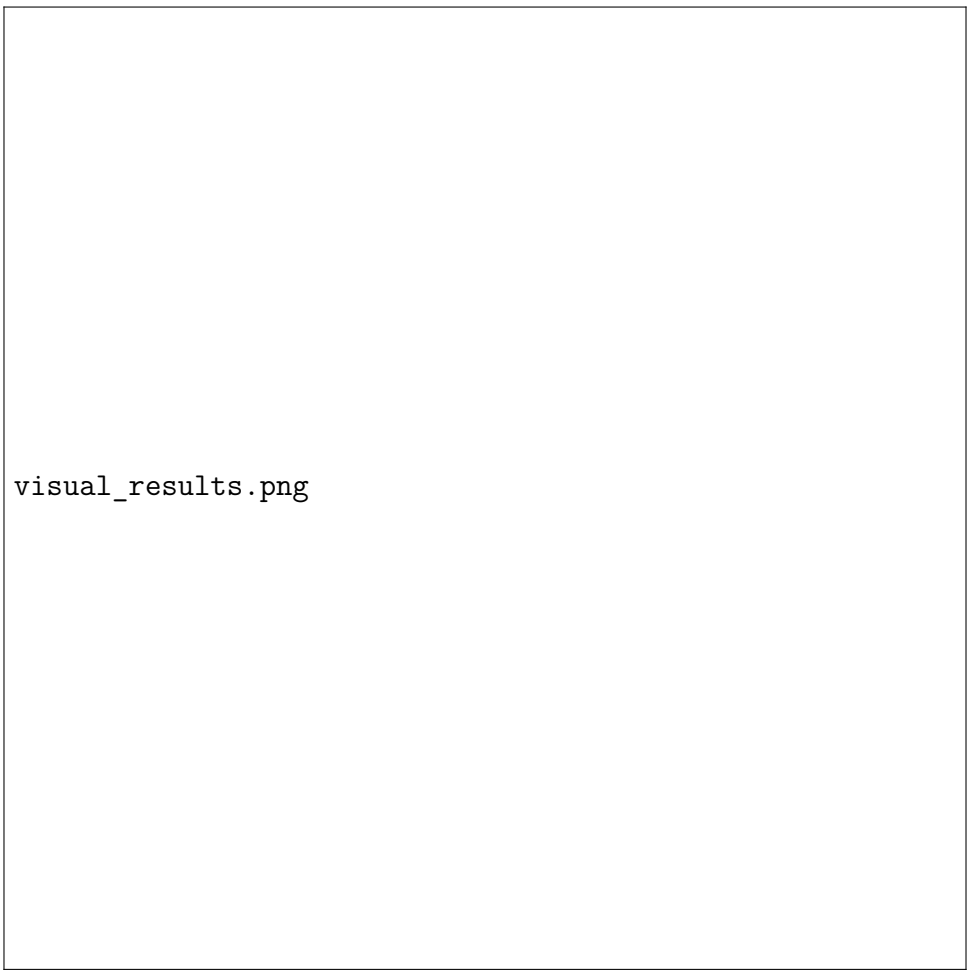

Figure 7.1: Comparison of input, teacher, and student outputs

7.5 Robustness Testing

- The system successfully handles diverse image inputs, including natural, synthetic, and noisy data.
- Invalid inputs (e.g., wrong formats, corrupted files) are managed with meaningful error messages and graceful fallback behaviors.
- No critical crashes or UI freeze were observed during extended web interface testing.

Chapter 8
Summary and Conclusion

By implementing deep learning algorithms, ERSVR 1.0 makes a practical addition to the domain of image super-resolution. Its main contribution is the teacher-student model which incorporates knowledge distillation to merge a lightweight student model's speed with a high-capacity model's accuracy. This approach has the advantage of operational readiness in real-world scenarios.

The system is built with integration flexibility, dataset additions, model component replacements, and numerous deployment strategy adaptations in mind. All modules are self-contained and work as part of the broader system which improves system reliability and extensibility.

Comprehensive testing shows that ERSVR 1.0 outperforms traditional interpolation-based approaches in accuracy during high-resolution image reconstruction. Its performance has been confirmed through quantitative measures such as PSNR, SSIM, and qualitative visual assessments. Enhanced usability is provided by an interfaced web application that enables users to apply advanced super-resolution technologies while alleviating the burden of detailed technical knowledge prerequisites.

Development of a reliable system has been accomplished along with all other primary project goals.

Chapter 9
Limitations and Future Work

9.1 Limitations

Limitation	Description	Mitigation
Generalization to Unseen Data	The system may not perform well on images significantly different from training data (e.g., medical scans, satellite images).	Incorporate diverse datasets; use domain adaptation or fine-tuning.
Real-Time Processing	On large images or resource-limited devices, inference speed may be insufficient for real-time use.	Model pruning, quantization, and ONNX/TensorRT deployment.
Web Interface Functionality	Current web interface lacks features like batch uploads, comparisons, and analytics.	Implement advanced frontend modules and UI enhancements.
Model Interpretability	The deep learning models lack explainability, making debugging or validation difficult.	Add visualization of activations and feature maps using tools like Grad-CAM.

9.2 Future Work

- **Incorporation of GAN-Based Models:** Integrate generative adversarial networks (e.g., SRGAN, ESRGAN) to enhance fine detail and perceptual sharpness.
- **Support for Video Super-Resolution:** Extend the pipeline to support sequential frames, ensuring temporal coherence in super-resolved videos.
- **Model Optimization for Edge Deployment:** Apply model compression, pruning, and quantization to make the system deployable on edge devices like mobile phones and IoT hardware.
- **Enhanced Web Interface:** Improve UI/UX by adding features such as authentication, result history, side-by-side comparisons, and batch uploads.

- **Automated Hyperparameter Tuning:** Use tools like Optuna or Ray Tune to automatically search for the best training parameters.
- **Explainability and Visualization:** Integrate interpretability tools (e.g., activation visualizers, saliency maps) to improve model transparency and trust.

Chapter 10
Bibliography