PROJECT REPORT ON

“IMAGE SUPER RESOLUTION USING CNN”

Submitted by:

D Sreeram: 245323733076

G Bhaargavv Jaswonthh: 245323733082

T Haaswith Sai: 245323733125

Avyukth Nunna: 245323748009

G Amisha: 245323753017

Ashrita Rachuri: 245323733132

Under the Guidance Of:

Anil Prasad Surampudi

NEIL GOGTE INSTITUTE OF TECHNOLOGY

Kachavanisingaram Village, Hyderabad, Telangana 500058.

February, 2025

CERTIFICATE

This is to Certify that ***D Sreeram (245323733076), G Bhaargavv Jaswonthh (245323733082), T Haaswith Sai (245323733125), Avyukth Nunna (245323748009), G Amisha (245323753017), Ashrita Rachuri (245323733132)*** *have successfully completed the project entitled "****IMAGE SUPER RESOLUTION USING CNN****" as part of* ***Project School*** *conducted by* ***Neil Gogte Institute of Technology***

The project was carried out under the able guidance of ***Mr. Anil Prasad Surampudi,*** *Project Mentor*

Date: February, 2025

Project Mentor: Anil Prasad Surampudi

Signature:

ACKNOWLEDGMENT

We would like to express our sincere gratitude to Mr. Anil Prasad Surampudi, our1 project mentor, for his invaluable guidance, constant support, and expert advice throughout the course of this project. His insights and suggestions have been instrumental in the successful completion of this work.

We also wish to extend our heartfelt thanks to the faculty members of Neil Gogte Institute of Technology for providing the necessary resources, knowledge, and encouragement during the project.

Finally, we would like to thank the college management for conducting the Project School and for creating an environment that fosters innovation and learning.

ABSTRACT

The demand for high-resolution images has significantly increased across various industries, including medical imaging, satellite imaging, and digital media. However, many real-world images are captured at lower resolutions, leading to a loss of crucial details and features. Image Super-Resolution (ISR) is a technique aimed at enhancing the resolution of images, making them clearer and more informative.

In this project, we explore the application of Convolutional Neural Networks (CNNs) for image super-resolution. CNNs, with their ability to learn hierarchical patterns and spatial dependencies in data, are particularly effective for tasks involving images. By training a deep CNN model on a dataset of low-resolution images, the network learns to generate high-resolution outputs that restore lost details and improve the overall quality of the image.

The process begins with a dataset of low-resolution images, which the CNN model is trained on. Through the training process, the network learns how to map these low-resolution inputs to their high-resolution counterparts by identifying important features and textures. Once trained, the model can predict and generate high-resolution images from new, unseen low-resolution inputs.

The results of this project show substantial improvements in image quality, with sharper details, more defined textures, and overall better visual clarity. This demonstrates the potential of CNN-based methods in real-world applications, such as improving medical diagnostics, enhancing security footage, and refining satellite images where precision is vital.

By leveraging deep learning techniques such as CNNs, this project showcases how machine learning can play a significant role in advancing image enhancement and quality improvement, paving the way for more accurate visual analysis and decision-making in various domains.

PREFACE

Images are everywhere—whether we're using them in our phones, exploring medical scans, or analysing satellite data. But sometimes, the images we work with are of low resolution, which means we miss out on important details. This project, Image Super-Resolution Using Convolutional Neural Networks (CNNs), is all about improving those low-quality images and making them clearer, sharper, and more useful.

In this project, we've worked with Convolutional Neural Networks (CNNs), a type of deep learning model that's really good at recognizing patterns and details in images. By training the CNN on a set of low-resolution images, it learns how to predict and enhance them, producing high-resolution versions with a lot more detail. The idea behind this is simple but powerful—by teaching the model how to "fill in the gaps," we can make blurry or pixelated images much clearer.

The motivation behind this project came from seeing how much we rely on high-quality images in different fields, whether it's healthcare, security, or even entertainment. The ability to take low-resolution images and make them sharper can open up so many possibilities. It's been an exciting process, and along the way, we've learned a lot about both the power of deep learning and the impact it can have in making images clearer.

This project is just one step in exploring how technology can help improve the way we see and understand images, and I hope it sparks more curiosity and inspiration in the field of image enhancement and machine learning.

TABLE OF CONTENT:

1. Introduction..……...8

2. Literature Survey...……..9

2.1 Early Approaches: Example-Based and Patch-Reuse Methods……...9

2.2 Sparse Representation Techniques……10

2.3 Deep Learning–Based Approaches……11

2.4 Discussion and Comparative Insights……12

2.5 Conclusion……..13

3. Proposed Work & Implementation……..……14

3.1 Proposed Work……...14

3.2 Dataset Preparation and Preprocessing……..15

3.3 Model Implementation……...17

3.4 Training Process and Evaluation……18

3.5 Evaluation Metrics……18

4. Interfaces & Communication Details……..……19

4.1 User Interface (UI)……19

4.2 Backend Interface……...19

4.3 System Components and Communication……19

4.4 Communication……..20

5. Experiments……25

5.1 Optimizer variants……..25

6. Results & Discussion……..27

6.1 Quantitative Results……...27

6.2 Qualitative Results……28

6.3 Analysis of Results……29

6.4 Challenges and Limitations……29

6.5 Possible Improvements……..29

7. Conclusion……...31

8. References & Bibliography……32

1 Introduction

The demand for high-quality images has grown significantly across various fields, including medical imaging, satellite surveillance, and digital media. However, many images are captured at lower resolutions due to hardware limitations, compression artifacts, or environmental factors, leading to loss of important details. Image Super-Resolution (ISR) is a crucial technique aimed at enhancing image quality by reconstructing high-resolution images from low-resolution inputs.

This project explores the application of Convolutional Neural Networks (CNNs) for super-resolution, leveraging deep learning techniques to restore fine details and improve image clarity. The Super-Resolution Convolutional Neural Network (SRCNN) is employed to learn mappings between low-resolution and high-resolution images, enabling the model to generate enhanced outputs. By implementing a web-based interface, users can seamlessly upload images and obtain super-resolved results using a trained deep learning model. The outcomes of this project demonstrate the potential of CNN-based approaches in real-world applications such as medical diagnostics, security enhancement, and remote sensing, where image quality plays a vital role in decision-making.

2 Literature Survey

The evolution of SR methods can be broadly categorized into three distinct eras. Early approaches, such as example-based and patch-reuse methods, exploited the redundancy inherent in natural images. Seminal works in this period demonstrated that by leveraging a database of paired low- and high-resolution patches or by searching within the image for self-similar patterns, it is possible to infer missing details and enhance image resolution.

Building on these foundations, the early 2000s witnessed the rise of sparse representation techniques. These methods model image patches as sparse linear combinations of elements drawn from over-complete dictionaries. By learning these dictionaries directly from data, researchers established a formal framework for capturing the essential structures and textures of images. Notable contributions in this area include strategies that not only generalize across various image types but also adapt to domain-specific challenges, such as face hallucination, where preserving identity-specific features is crucial.

More recently, the advent of deep learning has revolutionized image super-resolution. Neural network–based approaches, particularly convolutional neural networks (CNNs) and generative adversarial networks (GANs), have been employed to learn end-to-end mappings from low-resolution to high-resolution images. These methods benefit from large datasets and advanced training techniques, yielding reconstructions with significantly improved visual fidelity and perceptual realism. The integration of concepts from earlier methods, such as sparse representation, into deep architectures illustrates the field's progression towards more robust and adaptive solutions.

This survey synthesizes contributions from these varied approaches, tracing the journey from early example-based methods through the rigorous frameworks of sparse coding to the modern deep learning techniques that now dominate the field. By comparing these methods, we highlight not only their individual innovations and limitations but also the overall trajectory of research that continues to push the boundaries of what is achievable in image super-resolution.

2.1 Early Approaches: Example-Based and Patch-Reuse Methods

Example-Based Super-Resolution (2002) Freeman, Jones, and Pasztor’s seminal work laid the foundation for example-based super-resolution. By building a database of low- and high-resolution patch pairs, the method infers high-frequency details for an input low-resolution

image through nearest-neighbour search and patch synthesis. This approach underscored the value of learning from example data and demonstrated that rich detail can be recovered by leveraging internal image redundancies.

Super-Resolution from a Single Image (2003) Avidan and Hel-Or further explored the idea of exploiting internal image redundancies. Their technique involves searching within the image itself for self-similar patches at various scales and orientations, thereby reconstructing higher-resolution details without external training data. This work highlighted that even a single image often contains sufficient information for enhancing resolution when effectively harnessed.

Hallucinating Faces: TensorPatch Super-Resolution and Coupled Residue Compensation (2005) Wang and Tang focused on the domain of face images—a category where perceptual quality and identity preservation are paramount. Their method used tensor analysis combined with patch-based super-resolution and residue compensation to generate realistic high-resolution facial images. This early work in domain-specific SR demonstrated that incorporating structural constraints can yield improvements in both fidelity and visual quality.

2.2 Sparse Representation Techniques

The early 2000s methods paved the way for techniques that explicitly modelled the underlying structure of image patches. Sparse representation emerged as a powerful framework in which patches are represented as a linear combination of a small number of basis elements from an over-complete dictionary.

Image Super-Resolution as Sparse Representation of Raw Image Patches (2008) Jian and Wu introduced a method that directly employs raw image patches for sparse representation, foregoing intermediate feature extraction. Their approach demonstrated that preserving the original patch information could lead to more faithful reconstructions, as the learned dictionary captures essential image details that standard interpolation methods miss.

Image Super-Resolution via Sparse Representation (2010) Building on the principles of sparse coding, Yang and colleagues proposed a method in which two coupled dictionaries are learned—one for low-resolution patches and one for their high-resolution counterparts. For a given low-resolution patch, the sparse coefficients (obtained with respect to the low-resolution dictionary) are used to reconstruct the corresponding high-resolution patch from the high-resolution dictionary. This method demonstrated significant improvements in preserving sharp edges and textures and became one of the most-cited approaches in the field.

Face Hallucination via Sparse Coding (2010) Also from Yang and colleagues, this work extended sparse representation techniques specifically to facial images. By combining sparse coding with a global parametric model and incorporating non-negative matrix factorization, the method effectively captured localized facial features. This domain-specific adaptation—often referred to as face hallucination—yielded reconstructions that were both perceptually convincing and faithful to the underlying identity.

2.3 Deep Learning–Based Approaches

The advent of deep learning has transformed image super-resolution research, with neural network–based methods now dominating the field. While early deep learning models often drew inspiration from earlier patch-based and sparse representation techniques, they introduced new architectures that allowed for end-to-end learning and more sophisticated loss functions.

Image Super-Resolution Using Deep Convolutional Networks (2014) Dong et al. pioneered the use of convolutional neural networks (CNNs) for super-resolution. Their work demonstrated that a deep network could directly learn the mapping from low-resolution to high-resolution images, offering improvements in both speed and quality over traditional methods. This marked a significant departure from patch-based methods, as the network could learn hierarchical representations and end-to-end optimizations.

Photo-Realistic Single Image Super-Resolution Using a Generative Adversarial Network (2016) Ledig et al. introduced SRGAN, which combined a generative adversarial network (GAN) with a perceptual loss function to produce super-resolved images that are not only high in resolution but also photorealistic. By using an adversarial loss to guide the reconstruction process, the model generated finer textures and more visually appealing images, moving beyond the limitations of mean squared error–based objectives.

Perceptual Image Super-Resolution with Progressive Adversarial Network (2020) Further advancing GAN-based approaches, Wong and colleagues proposed a progressive adversarial network that gradually enhances image resolution. By avoiding traditional reconstruction losses and focusing on perceptual quality, this method showcased the benefits of progressive training and U-Net architectures in recovering high-fidelity details while mitigating artifacts.

A Comprehensive Review of Deep Learning-based Single Image Super-Resolution (2021) Bashir et al. provided an extensive review of the progress in single-image super-resolution, categorizing methods into classical, supervised, unsupervised, and domain-specific

approaches. This survey contextualized the evolution from sparse representation and example-based methods to modern deep learning approaches, offering insights into the strengths and limitations of each category and outlining challenges for future research.

StarSRGAN: Improving Real-World Blind Super-Resolution (2023) Vo and Bui's StarSRGAN represents the latest evolution in GAN-based methods for blind super-resolution—where the degradation process is unknown. This approach integrates novel architectures to improve real-world performance, with significant improvements in both reconstruction speed and perceptual quality, paving the way for real-time applications in SR.

2.4 Discussion and Comparative Insights

The evolution of image super-resolution methods can be viewed as a progression from leveraging explicit examples and self-similarity in images to more complex representations:

2.4.1 Early Methods:

The example-based and patch-reuse techniques of Freeman et al. (2002) and Avidan & Hel-Or (2003) laid the conceptual groundwork by illustrating that image redundancy can be exploited for SR.

2.4.2 Sparse Representation:

The sparse representation techniques introduced by Jian & Wu (2008) and further refined by Yang et al. (2010) provided a formal mechanism for capturing the essential structure of image patches. These methods effectively balanced fidelity and computational complexity through learned dictionaries and sparse coding strategies, with specialized adaptations for domains like face hallucination.

2.4.3 Deep Learning Era:

Starting with Dong et al. (2014), deep learning models began to dominate SR research by learning hierarchical representations and end-to-end mappings. Subsequent advances, notably with SRGAN (2016) and progressive adversarial networks (2020), shifted the focus toward perceptual quality and realism. Comprehensive reviews and state-of-the-art systems like StarSRGAN (2023) demonstrate that modern methods continue to build on and integrate principles from earlier work, including aspects of sparse representation, even as the primary modelling paradigm has shifted.

Each phase brought its own innovations, and together they outline a trajectory of increasing complexity, performance, and practical applicability. While early methods emphasized explicit modelling of image patches, later approaches leverage the power of large datasets and deep architectures to learn more abstract representations, ultimately leading to impressive improvements in both objective and perceptual quality.

2.5 Conclusion

The surveyed literature reflects a dynamic field marked by continuous innovation. Early methods set the stage by demonstrating the viability of learning from examples and exploiting internal redundancies. The introduction of sparse representation techniques provided a robust framework for modelling image details, which in turn influenced later deep learning methods. Today's state-of-the-art systems—ranging from CNN-based models to sophisticated GAN architectures—stand on the shoulders of these earlier contributions. Together, these works not only illustrate the evolution of image super-resolution but also inspire ongoing research that seeks to further bridge the gap between low-resolution inputs and high-fidelity, realistic outputs.

3 Proposed Work & Implementation

3.1 Proposed Work

The project, Image Super-Resolution using Convolutional Neural Networks (CNNs), focuses on transforming low-resolution (LR) images into high-resolution (HR) images by leveraging deep learning techniques. Super-resolution is a crucial technology in medical imaging, satellite imagery, security surveillance, forensics, and general photography, where image clarity and detail restoration play a significant role in decision-making and analysis.

3.1.1 Problem Statement & Motivation

In many real-world applications, images suffer from poor resolution due to hardware limitations, compression artifacts, and environmental factors. Traditional interpolation methods like bilinear and bicubic upscaling fail to restore fine details and often introduce blurriness and noise. To overcome these limitations, we utilize deep learning-based super-resolution techniques that learn feature hierarchies directly from data, ensuring superior image reconstruction.

The primary goal of this project is to develop a deep learning model that can enhance image resolution while preserving structural integrity and minimizing artifacts. The Super-Resolution Convolutional Neural Network (SRCNN) serves as the backbone of our implementation, offering an end-to-end learning approach that optimizes feature extraction, non-linear mapping, and image reconstruction.

3.1.2 Why CNNs for Image Super-Resolution?

Convolutional Neural Networks (CNNs) have demonstrated exceptional performance in image processing tasks due to their ability to:

- Extract spatial hierarchies and detect fine details in images.
- Preserve edge structures and textures while enhancing image quality.
- Reconstruct high-resolution details by learning complex mappings from low-resolution inputs.

By leveraging CNNs, we can achieve higher accuracy and realism in image super-resolution, making it a more effective approach than conventional methods.

3.1.3 Project Components

This project consists of three core components that work together to enable real-time image super-resolution:

- Backend (Deep Learning Model & Processing Engine):
    - Developed using Flask, a lightweight Python-based web framework.
    - Hosts the pre-trained SRCNN model, allowing it to handle image processing requests.
    - Accepts low-resolution image inputs and returns high-resolution outputs.
- Frontend (User Interface):
    - A web-based interactive interface where users can upload images for super-resolution enhancement.
    - Displays before-and-after comparisons of the uploaded image.
    - Built using HTML, CSS, and JavaScript, ensuring a smooth user experience.
- API Communication (Backend-Frontend Integration):
    - Implements a RESTful API to enable seamless data exchange between the frontend and backend.
    - API endpoints handle image uploads, processing requests, and results retrieval.
    - Uses Flask's request-response mechanism to ensure efficient communication.

The following sections provide an in-depth explanation of dataset preparation, model implementation, training methodologies, and API integration strategies.

3.2 Dataset Preparation and Preprocessing

To train and evaluate the SRCNN model, a dataset of paired high-resolution (HR) and low-resolution (LR) images was required. The dataset was sourced from Kaggle's Image Super-Resolution Dataset which consists of high-quality images that were down sampled to create corresponding low-resolution versions.

3.2.1 Dataset Characteristics:

- The dataset contains thousands of image pairs, ensuring diverse image types for better generalization.
- Each low-resolution image (input) is paired with its high-resolution counterpart (ground truth) to facilitate supervised learning.

3.2.2 Preprocessing Steps:

- Data Loading & Normalization:
    - The dataset was loaded using PyTorch's ImageDataset class, which efficiently manages HR and LR image pairs.
    - Images were normalized to pixel values between 0 and 1, improving model convergence and stability.
    - The image resolution was adjusted dynamically to ensure compatibility with the SRCNN model.
- Data Splitting Strategy:
    - The dataset was divided into 80% for training and 20% for testing to evaluate generalization performance.
    - The PyTorch DataLoader was utilized for batch processing and shuffling, ensuring balanced learning across all images.
- Batching and Augmentation:
    - Training images were batched in groups of 5, enabling efficient parallel computation on the GPU.
    - Data augmentation techniques such as random flipping, rotations, and contrast enhancements were considered to increase model robustness.

The dataset played a crucial role in training the SRCNN model, helping it minimize the Mean Squared Error (MSE) between predicted high-resolution images and their corresponding ground-truth images.

3.3 Model Implementation

The Super-Resolution Convolutional Neural Network (SRCNN) was implemented using PyTorch. It follows a three-layer convolutional architecture, optimized for feature extraction, non-linear mapping, and image reconstruction.

3.3.1 SRCNN Architecture Overview:

The SRCNN model enhances image resolution through a multi-step learning process:

- Layer 1 (Feature Extraction):
    - 64 convolutional filters with a 9×9 kernel size.
    - Extracts key structural information and essential features from the low-resolution input.
- Layer 2 (Non-Linear Mapping):
    - 32 convolutional filters with a 1×1 kernel size.
    - Transforms extracted features into a refined high-resolution representation.
- Layer 3 (Image Reconstruction):
    - 3 convolutional filters with a 5×5 kernel size.
    - Reconstructs the final high-resolution image output by refining details and eliminating noise.

3.3.2 Activation Functions:

- Leaky ReLU activation function was used for its ability to handle small gradient values, ensuring effective deep feature learning.

3.3.3 Model Optimization:

- Optimizer: Adam Optimizer, which dynamically adjusts learning rates for faster convergence.
- Loss Function: Mean Squared Error (MSE), minimizing pixel-wise differences between predicted and actual images.

3.3.4 GPU Utilization & Performance Optimization:

- The model was trained on a GPU (NVIDIA CUDA-enabled device), significantly accelerating computation.
- Challenges such as memory limitations and crashes due to large batch sizes were mitigated through batch size tuning and model optimization techniques.

3.4 Training Process and Evaluation

The SRCNN model was trained for 30 epochs, with the following workflow applied to each batch:

- Forward Propagation:
    - The model processes low-resolution images to generate high-resolution predictions.
- Loss Computation:
    - The MSE loss function calculates the error between predicted and ground-truth images.
- Backpropagation & Parameter Updates:
    - The Adam optimizer updates model parameters, improving accuracy with each epoch.

3.5 Evaluation Metrics:

- Peak Signal-to-Noise Ratio (PSNR):
    - Measures reconstruction quality—higher PSNR indicates better image restoration.
- Visual Inspection & User Evaluation:
    - Side-by-side comparisons of input, output, and ground-truth images ensure qualitative assessment.

4 Interfaces & Communication Details

4.1 User Interface (UI)

The frontend has three pages to ensure simplicity and usability:

Page 1: Users upload images using drag-and-drop or a button.

Page 2: Displays the selected image with options to change or enhance the image.

Page 3: Shows both the original and enhanced images, with a Download button for the result.

4.2 Backend Interface

The backend, powered by Flask, connects the frontend to the deep learning model (SRCNN).

Flask's role:

Handles image uploads from the frontend via HTTP requests.

Sends the image to the SRCNN model for processing.

Returns the enhanced image to the frontend for display.

API Endpoints:

/upload_img (POST): Receives the uploaded image from the user.

/display (GET): Sends back the enhanced image after processing.

4.3 System Components and Communication

Frontend: Allows users to upload images, view results, and download the enhanced output.

Backend: Prepares and processes the image using Flask and the SRCNN model.

Deep Learning Model (SRCNN): Enhances image quality.

Communication Workflow:

The frontend sends the uploaded image to the backend through the /upload_img endpoint.

Flask preprocesses the image and passes it to the SRCNN model.

The model generates the enhanced image, which is returned to the frontend through the /display endpoint.

The frontend displays both the input and enhanced images on Page 3.

4.4 Communication

4.4.1 API Design

The API connects your frontend and backend, allowing them to exchange data (like images). Here's a breakdown of the structure:

- POST Method (For File Uploads):
  - Endpoint: /upload_img
  - Purpose: This is the endpoint the frontend uses to send the selected image to the backend. The frontend will send the image as part of a POST request.
  - Details: The image is sent to the server using form tag, which is used for uploading files.
- GET Method (For Retrieving Results):
  - Endpoint: /display
  - Purpose: After the backend processes the image using your PyTorch model, the frontend will request the processed image. The GET method is used here because it's for retrieving data from the server.
  - Details: The backend will send the enhanced image to the frontend, in a format like base64 for embedding directly in the page.

4.4.2 Data Flow

The data exchanged between the frontend and backend must be in a format that both understand. Here's how it works:

From Frontend to Backend (POST request):

When the user uploads an image, the frontend sends it to the backend. The image is sent in the body of the request as a file (like JPEG, PNG).

The backend receives the file and processes it using your PyTorch model.

From Backend to Frontend (GET request):

After the image is processed, the backend sends the result (the enhanced image) back to the frontend.

This can be done in two ways:

Image file: The backend sends the processed image as a file to the frontend.

Base64 encoded image: If you want to display the image directly on the webpage, the image can be converted into a base64 string, which the frontend can embed

4.4.3 Error Handling

- Invalid Input Handling: If the user uploads an unsupported file type (e.g., a .txt file instead of an image), the backend should respond with a helpful error message, like "Invalid file type. Please upload a valid image."
- Backend Errors: If there's an issue during image processing (e.g., the model fails), the backend should return a response like "Error occurred while upscaling the image, please try again!"
- Bad Requests: If the user doesn't upload any file or sends an incomplete request, the backend should send a "400 Bad Request" response with a message like "No file received."
- Postman: You can use tools like Postman to test the API manually. For example, you can simulate uploading an image with a POST request to /upload_img and check if the server processes it correctly. You can also test the /display endpoint to ensure it returns the processed image.

In summary:

POST: Used for uploading images (to /upload_img).

GET: Used for retrieving processed images (from /display).

Data Flow: Images are sent from the frontend to the backend as files, and the backend sends back the processed image.

Error Handling: Handle issues like invalid file types and server errors gracefully.

Testing: Use tools like Postman and pytest to ensure the API works as expected.

4.4.4 Tools and Libraries

Frontend Tools:

- HTML:
  - Description: HTML (Hyper Text Markup Language) is used to structure the content of your website. It defines things like headings, paragraphs, images, and buttons.
  - Role: It's used to create the basic layout of your web pages, such as the image upload area, buttons, and display sections.
- CSS:
  - Description: CSS (Cascading Style Sheets) is used to style your web pages. It controls the look of the HTML elements, such as colors, fonts, spacing, and positioning.
  - Role: It ensures your web pages are visually appealing and responsive. For example, it makes sure your image upload area looks good and the buttons are easy to use.
- JavaScript:
  - Description: JavaScript is a programming language that makes web pages interactive. It can update content on the page without reloading, handle user events (like clicks), and communicate with the backend.
  - Role: It’s used for things like handling button clicks (e.g., uploading an image) and sending requests to the backend to process the image.

Backend Tools:

- Flask:
  - Description: Flask is a lightweight web framework for Python. It helps you create web servers that handle requests (like getting or sending images) and serve responses.
  - Role: Flask connects the frontend and backend. It receives the image from the frontend, passes it to your PyTorch model for processing, and sends the processed image back to the frontend. Flask handles routing, such as:
  - /upload_img: For receiving the image upload from the user.
  - /display: For returning the processed image.

Python libraries

- PyTorch:
  - Description: PyTorch is an open-source machine learning library used for building deep learning models. It's particularly good for creating neural networks such as SRCNN (Super-Resolution Convolutional Neural Network) model.
  - Role: PyTorch processes the image using the trained model to enhance its quality. It takes the input image, passes it through layers of the network, and outputs an enhanced image.
- os: Helps with file operations, like checking directories and paths.
- tqdm: Adds progress bars to loops, helpful for tracking the progress of long-running tasks like model training or image processing.
- matplotlib.pyplot: Used for visualizing images and creating plots to show results.
- torch.nn: Provides the building blocks for defining and constructing neural networks in PyTorch.
- torch.optim: Used to implement optimization algorithms (e.g., Adam, SGD) for training neural networks.
- torch.utils.data: Helps with creating datasets and data loaders, managing data during training and inference.
- torch.Tensor: Used to represent multi-dimensional data in PyTorch (similar to NumPy arrays).

4.4.5 Figures and Diagrams

1. Work flow

This flowchart will show how data flows from the frontend to the backend and through the PyTorch model. Here's how it could look:

Step 1: User uploads an image on the frontend (via drag-and-drop or file upload).

Step 2: Frontend sends the image to the backend using a POST request.

Step 3: Backend receives the image, loads it, and processes it through the PyTorch model (SRCNN).

Step 4: The processed image is returned to the frontend.

Step 5: The frontend displays both the original and the enhanced images. Users can download the output image.

This flowchart would clearly show the sequence of actions in the project.

2. System Architecture Diagram

This diagram would illustrate the components of the system and how they interact with each other. Components could include:

- Frontend:
  - Web page with file upload, buttons, and image display.
  - JavaScript handling user input and sending requests to the backend.
- Backend:
  - Flask server handling requests (/upload_img, /display).
  - PyTorch model for image processing.
- Data Exchange:
  - POST request for uploading images.
  - GET request for fetching results.
- Deep Learning:
  - PyTorch model (SRCNN) for image enhancement.

This diagram would give a high-level overview of the entire system and how each part connects.

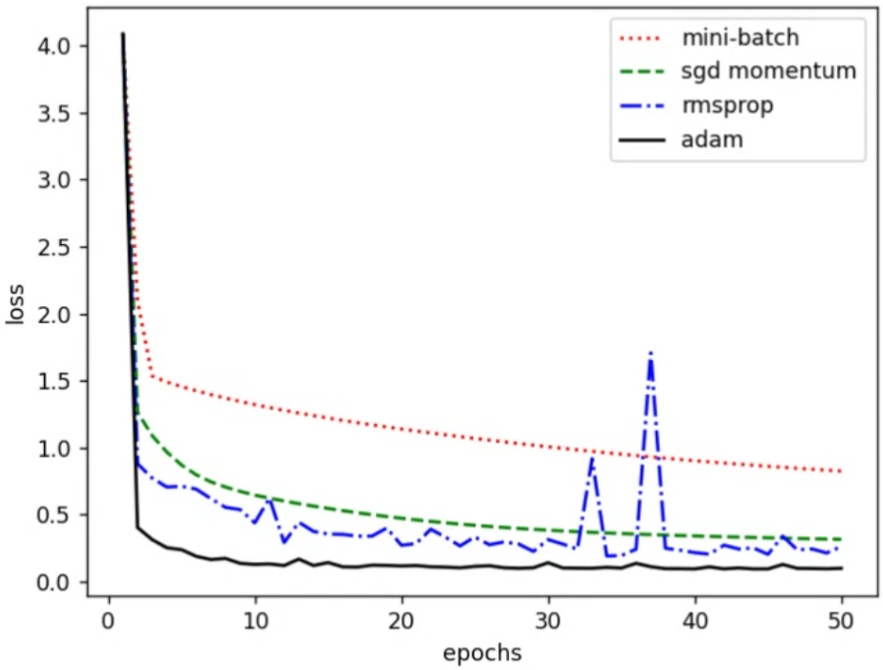
5. Experiments

5.1 Optimizer variants

To evaluate the performance of different optimization algorithms in training our SRCNN model for image super-resolution, we conducted experiments using the following optimizers:

1. Mini-Batch Gradient Descent (MBGD)
2. Stochastic Gradient Descent with Momentum (SGD Momentum)
3. RMSprop
4. Adam

The model was trained for 50 epochs, and the Mean Squared Error (MSE) loss was recorded at each epoch. The training loss was plotted against the number of epochs to analyse the convergence behaviour of each optimizer.



The loss vs. epochs graph shows the performance of the four optimizers:

- Mini-Batch Gradient Descent (Red, Dotted Line)
    - Converges slowly compared to other optimizers.
    - Exhibits a gradual decline in loss but remains relatively high throughout training.

    - Less suitable for fast convergence in deep learning models.
- SGD with Momentum (Green, Dashed Line)
    - Faster convergence than MBGD due to momentum, which helps overcome sharp gradients.
    - Smoother loss curve but still slower compared to Adam and RMSprop.
- RMSprop (Blue, Dash-Dot Line)
    - Converges quickly in the initial epochs but shows oscillations in later epochs.
    - This indicates instability in gradient updates, which might require fine-tuning the learning rate.
- Adam (Black, Solid Line)
    - Converges the fastest and reaches the lowest validation loss.
    - The smoothest curve with minimal oscillations, indicating stable learning.
    - The best-performing optimizer for our SRCNN model.

From our experiments, we observed that Adam outperforms other optimizers in terms of convergence speed and stability. RMSprop, while effective initially, showed instability in later epochs. SGD with Momentum improved convergence over basic Mini-Batch Gradient Descent, but was still slower than Adam.

Hence for training our SRCNN model, Adam was chosen.

6. Results & Discussion

6.1 Quantitative Results

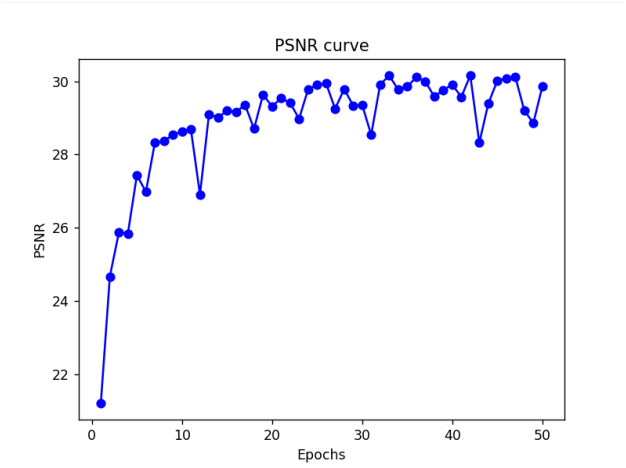
To evaluate the performance of our SRCNN model, we used Peak Signal-to-Noise Ratio (PSNR) as our key evaluation metric. It quantifies how close the super-resolved image (output of SRCNN) is to the original high-resolution image.

The results on the test dataset are as follows:

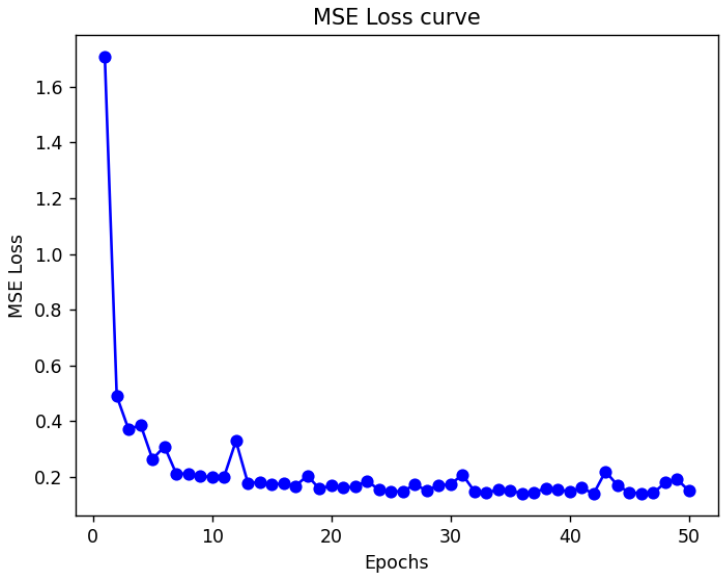
Method	**PSNR (dB)**
Bicubic Interpolation	28.13
SRCNN	31.04

The SRCNN model outputs showed a better PSNR value than Bicubic interpolation even when the model was trained for comparatively less number of epochs. This demonstrates the effectiveness of deep learning-based super-resolution over traditional upscaling techniques.

A graph was plotted to show the change in PSNR with increasing number of epochs. The PSNR value is observed to increase significantly indicating that the model is reconstruct high resolution images that are close to the ground truth.

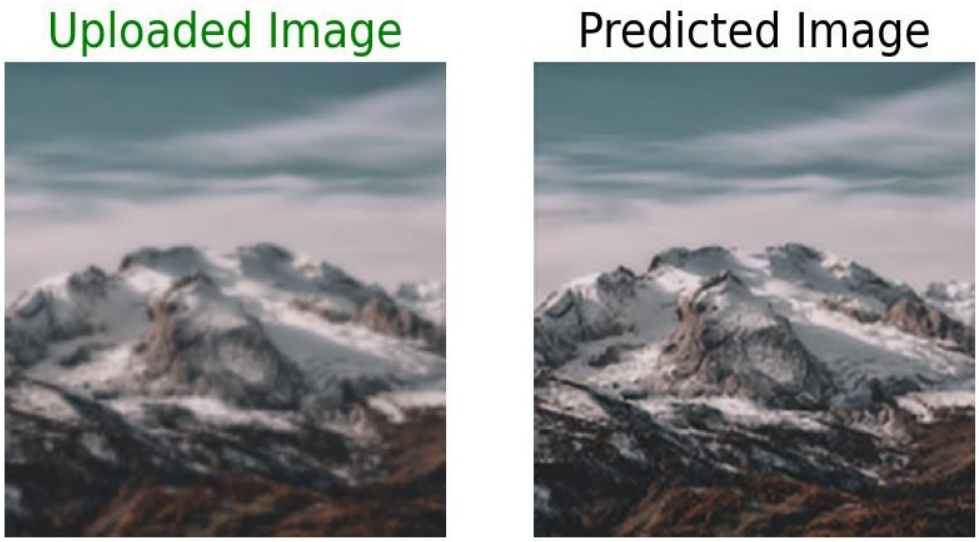


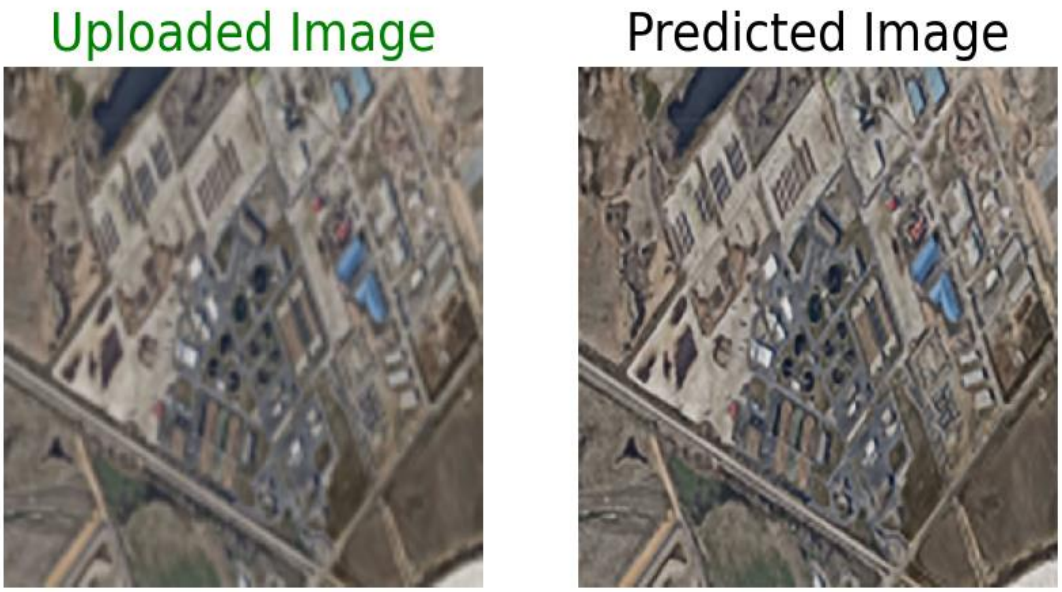
Additionally, the training loss curve over 50 epochs indicates that the model is effectively learning to map low resolution images to high resolution images.



6.2 Qualitative Results

Visual comparisons between the low-resolution inputs, SRCNN outputs, and ground truth images reveal sharpness improved quality. Below are sample results:





While SRCNN enhances details and reduces blurriness

6.3 Analysis of Results

The improvements in PSNR indicate that SRCNN effectively reconstructs finer image details. However, for even better results, the model has to be trained over a larger dataset and for a larger number of epochs which might be computationally expensive.

Comparing against bicubic interpolation, SRCNN performs better due to its ability to learn feature mappings rather than relying on simple mathematical upscaling.

6.4 Challenges and Limitations

Despite its success, SRCNN has some limitations:

- **Computational Cost**: The model has to be trained over a larger dataset and for a larger number of epochs to perform better which might be computationally expensive.
- **Poor performance for certain kind of images**: In case of images with darker background, the model seems to perform poorly compared to other kind of images

6.5 Possible Improvements

To enhance performance, future improvements could include:

- Using deeper networks to capture more complex image features.
- Training on a larger and more diverse dataset to improve generalization.
- Experimenting with different loss functions and hyper parameters for better texture preservation.

7. Conclusion

The implementation of Image Super-Resolution using CNNs has shown promising results in enhancing low-resolution images while preserving crucial details. Through the use of the SRCNN model, the project successfully demonstrates how deep learning can be utilized to improve image quality without relying on traditional interpolation methods. The web-based interface further provides a user-friendly platform for image enhancement, making this approach accessible for practical applications.

While the project has achieved significant improvements in image clarity, there are still areas for future enhancements, such as incorporating more advanced architectures like Generative Adversarial Networks (GANs) or Transformer-based models for even better perceptual quality. Additionally, optimizing the model for real-time processing and expanding the dataset for diverse image types can further enhance performance. Overall, this project highlights the transformative potential of deep learning in image processing and opens doors for further exploration in the field of super-resolution and computer vision.

8. References & Bibliography

[1] ***Chao Dong, Chen Change Loy, Kaiming He, and Xiaoou Tang.*** *Image Super-Resolution Using Deep Convolutional Networks. IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI), 2016.*

[2] ***Dataset:*** *The model is trained and tested using the Image Super-Resolution dataset from Kaggle: Kaggle Dataset - Image Super-Resolution.*

[3] ***W. Freeman, T. Jones, E. Pasztor****. Example-Based Super-Resolution (2002)*

[4] ***Wei Liu, Dahua Lin, Xiaoou Tang****. Hallucinating faces: TensorPatch super-resolution and coupled residue compensation (2005)*

[5] ***Jianchao Yang, John Wright, Thomas S. Huang, Yi Ma****. Image super-resolution as sparse representation of raw image patches (2008)*

[6] ***Jianchao Yang, John Wright, Thomas S. Huang, Yi Ma****. Image Super-Resolution Via Sparse Representation (2010)*

[7] ***Christian Ledig, Lucas Theis, Ferenc Huszar.*** *Photo-Realistic Single Image Super-Resolution Using a Generative Adversarial Network (2016)*

[8] ***Lone Wong, Deli Zhao, Shaohua Wan, Bo Zhang****. Perceptual Image Super-Resolution with Progressive Adversarial Network (2020)*