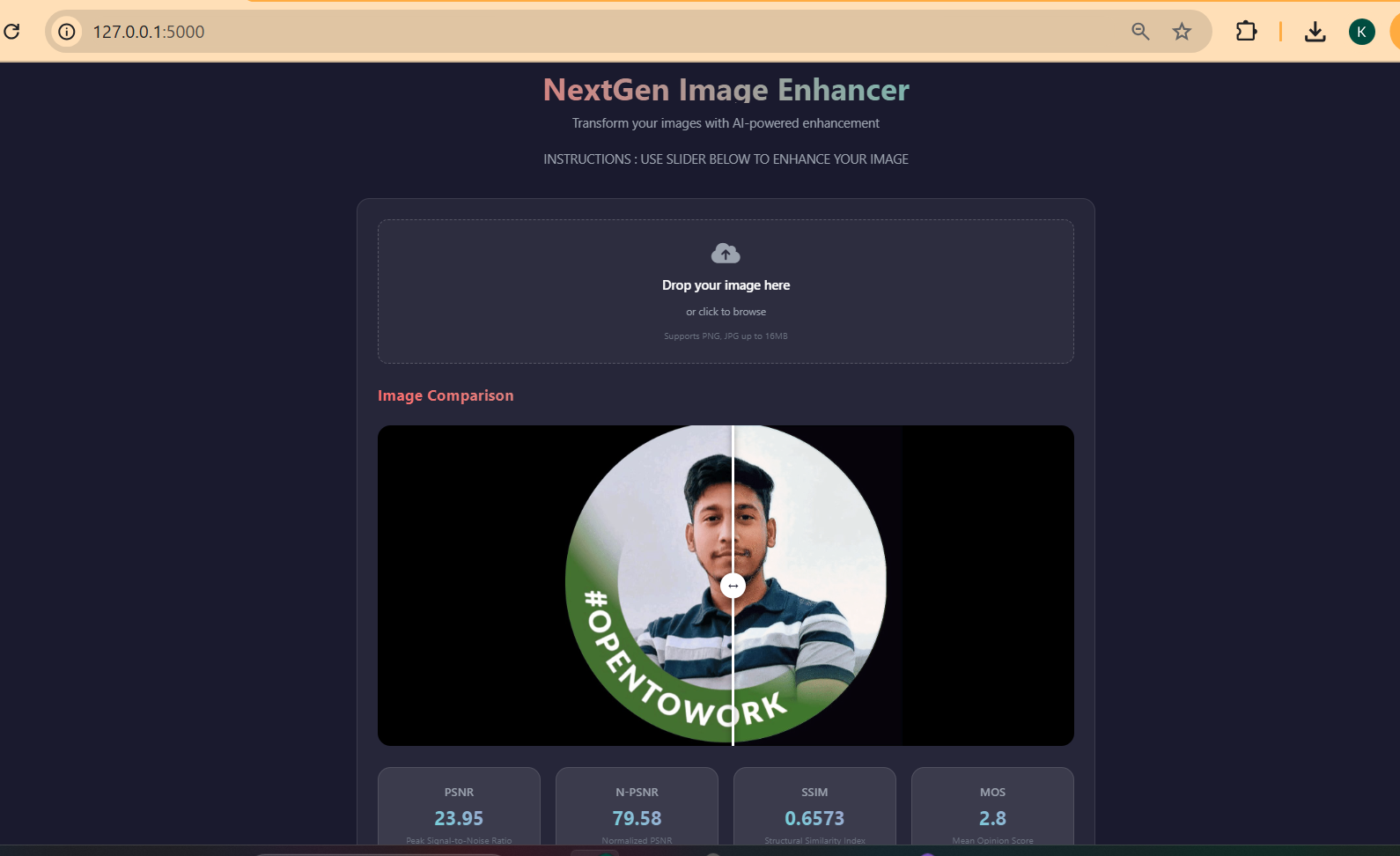
**Image Enhancer Using GANs**

Nexus – PS-1 (Computer Vision )



Materials Engineering Dept. IIT JAMMU

**An advanced deep learning project , implements image super-resolution using Generative Adversarial Networks (GANs). This model enhances low-resolution images to create high-quality, detailed high-resolution outputs using state-of-the-art deep learning techniques.**

**Application Interface** 

**Prerequisites**

** Python 3.8+**

** CUDA-capable GPU (recommended)**

** 16GB+ RAM**

***# Required Python packages***

**torch>=1.8.0**

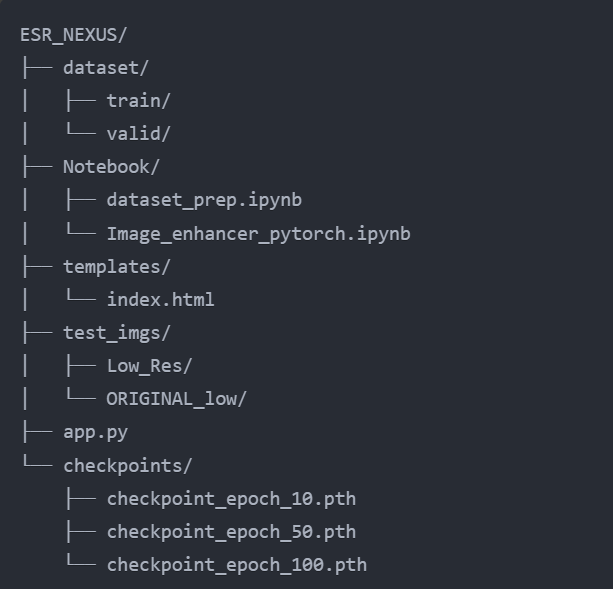
**torchvision>=0.9.0**

**numpy>=1.19.2**

**Pillow>=8.0.0**

**jupyter>=1.0.0**

**Project Structure :**



**Running the Application :**

1. Download ESR\_nexus directory to your system and open in IDEs like VS Code .
2. Create and activate a virtual environment (recommended): python -m venv venv

source venv/bin/activate *# On Windows: venv\Scripts\activate*

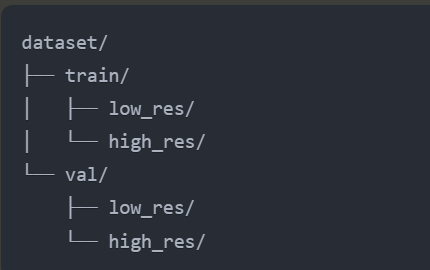
1. Install dependencies:

pip install -r requirements.txt

**: Using the Web Interface**

1. Start the web application:
   * python app.py
2. Open your web browser and navigate to: http://localhost:5000
3. Upload your low-resolution image and proceed with enhancement.

**Dataset Structure**

****

**Features**

* **GAN-based Architecture**: Utilizes advanced Generator-Discriminator architecture for realistic image enhancement
* **Multiple Loss Functions**: Combines adversarial, content, and perceptual (VGG) losses for optimal results
* **Robust Image Processing**: Handles various image formats and sizes with automatic preprocessing
* **Quality Metrics**: Comprehensive evaluation using PSNR, SSIM, and normalized PSNR
* **Web Interface**: Flask-based web application for easy image enhancement
* **Model Checkpointing**: Regular saving of model states for training continuity
* **GPU Acceleration**: Full CUDA support for faster processing

**Architecture**

**Generator Network**

The generator employs a deep residual architecture designed for optimal image enhancement:

* **Initial Layer**: 9x9 convolutional layer with PReLU activation
* **Residual Blocks**: 16 residual blocks with:
  + Dual 3x3 convolutional layers
  + Batch normalization
  + PReLU activation
  + Skip connections
* **Upsampling**: Custom blocks using pixel shuffle for 2x resolution increase
* **Final Layer**: 9x9 convolutional layer with Tanh activation

**Discriminator Network**

A sophisticated convolutional network for real/fake image classification:

* **Convolutional Layers**: Multiple layers with increasing channels (64 to 512)
* **Activation**: LeakyReLU with 0.2 negative slope
* **Batch Normalization**: After each convolutional layer
* **Dense Layers**: Final classification layers (512 → 1024 → 1)

**Loss Functions**

Multiple loss components for balanced training:

1. **Adversarial Loss**: BCE loss for GAN training
2. **Content Loss**: MSE loss for pixel-level accuracy
3. **Perceptual Loss**: VGG19-based loss for natural features
4. **Combined Loss**: Weighted sum of above components

**Challenges and Solutions**

While implementing the Image Super-Resolution model, several challenges were encountered and addressed:

1. **Training Stability**: GANs are notoriously difficult to train, with issues such as mode collapse or the generator producing unrealistic images.
   * **Solution**: Careful tuning of hyperparameters, especially learning rates and batch sizes, helped mitigate these issues. Additionally, using a pre-trained VGG network for perceptual loss improved training stability.
2. **Quality of Output**: Initially, the model generated blurry images due to insufficient learning capacity in the Generator.
   * **Solution**: Increased the depth of the Generator and used residual blocks to improve feature learning. Enhanced image quality with the inclusion of perceptual loss based on VGG19.
3. **Training Time**: Training the model from scratch took significant time, especially when the dataset size increased.
   * **Solution**: Utilizing a powerful GPU for training significantly reduced the time. Checkpoints were saved regularly to resume training without losing progress.
4. **Balancing Generator and Discriminator**: A common issue with GANs is the imbalance between the generator and discriminator during training, where one network becomes too powerful compared to the other.
   * **Solution**: This was handled by adjusting the loss weights for the generator and discriminator, ensuring they both improved at a similar rate.
5. **Evaluation Metrics**: Evaluating the quality of generated high-resolution images can be challenging, especially for perceptual quality.
   * **Solution**: Using both PSNR and SSIM along with perceptual loss gave a more comprehensive measure of image quality.
6. **Overfitting**: With a smaller dataset, there was a risk of the model overfitting, especially on the training data.
   * **Solution**: Data augmentation techniques such as flipping, rotating, and cropping were used to introduce more variety into the training dataset.

**Quality Metrics**

The system provides comprehensive quality assessment:

* **PSNR (Peak Signal-to-Noise Ratio)**
* **SSIM (Structural Similarity Index)**
* **Normalized PSNR**
* **Estimated Mean Opinion Score (MOS)**

**Image Processing**

* Automatic image format conversion
* Resolution constraints handling
* Memory-efficient processing
* CUDA memory management
* Batch processing capabilities

**METRICS ACHIEVED**

**Final Model Performance AFTER 100 EPOCHS**

* **Average Structural Similarity Index Measure (SSIM):** 0.9003
* **Validation Peak Signal-to-Noise Ratio (PSNR):** 28.99
* **Validation SSIM:** 0.9009

