# Mini Project: Bidirectional Image Conversion: Sketch to Real and Real to Sketch

## Introduction:

This mini project aims to develop a system capable of:

1. Converting real images into pencil sketches using Gaussian blur and edge detection.
2. Converting pencil sketches back into realistic images using a trained DCGAN (Deep Convolutional Generative Adversarial Network).

The system will feature a user-friendly web-based interface, allowing seamless interaction with both functionalities and real-time adjustments to conversion parameters.

## Objectives:

### 1. Image to Sketch Conversion

❿ Utilize Gaussian blur and edge detection techniques to convert real images into pencil sketches.

❿ Provide adjustable parameters for sketch intensity, such as Gaussian blur levels and edge detection thresholds.

### 2. Sketch to Real Image Conversion

❿ Train a DCGAN model to generate realistic images from pencil sketches.

❿ Explore advanced architectures like Pix2Pix and CycleGAN for improved performance and quality of outputs.

### 3. User-Friendly Interaction

❿ Develop an intuitive web-based interface with features for:

  ❿ Uploading real images or sketches.

  ❿ Adjusting sketch conversion parameters with real-time previews.

  ❿ Converting sketches back into realistic images.

  ❿ Batch processing for handling multiple images at once.

## Technical Workflow:

### 1. Image to Sketch Conversion:

**Methodology:**

1. Convert the input image to grayscale.
2. Apply adjustable Gaussian blur to the grayscale image.
3. Detect edges using Canny edge detection or Sobel filters.
4. Invert the detected edges to create a pencil sketch effect.

**Features:**

- Adjustable sliders for Gaussian blur intensity and edge detection thresholds.
- Real-time preview of the pencil sketch effect.

**Output:**

A grayscale pencil sketch of the input image, with options to save the output.

## 2. Sketch to Real Image Conversion:

**Methodology:**

1. **Model Architecture:**
    - Use a DCGAN with the following components:
        - Generator: Converts pencil sketches into realistic images.
        - Discriminator: Distinguishes between real and generated images.
    - Explore Pix2Pix and CycleGAN architectures for improved performance and quality.
2. **Training Process:**
    - Dataset Preparation:
        - Use paired datasets like CUHK Sketch Dataset or Kaggle datasets.
        - Augment data with variations using Albumentations (e.g., noise, distortions).
    - Train the GAN in an adversarial manner:
        - The generator minimizes the difference between real and generated images.
        - The discriminator distinguishes between real and fake inputs.
    - Apply stabilization techniques, including gradient clipping and batch normalization.
3. **Evaluation Metrics:**
    - Fréchet Inception Distance (FID): Measures similarity between real and generated images.
    - Structural Similarity Index (SSIM): Quantifies the perceptual similarity.

**Output:**

A realistic image generated from the input pencil sketch, available for download.

## 3. Web Interface for Interaction:

**Features:**

1. **Image Upload:**
    - Upload real images or pencil sketches via a drag-and-drop interface.
2. **Image to Sketch Conversion:**
    - Adjust sketch parameters and view real-time previews.
3. **Sketch to Real Image Conversion:**
    - Upload or select a generated pencil sketch for realistic image generation.
4. **Batch Processing:**
    - Process multiple images in one session.
5. **Performance Optimization:**
    - Optimize conversion speed for real-time usability.
6. **Deployment:**

- Deploy the system using Flask for the back-end and React for an interactive front-end.
- Host the application on platforms like AWS or Heroku or any for public access.

## Tools and Libraries:

### 1. Python Libraries:

- **OpenCV:** For image processing (Gaussian blur, edge detection).
- **NumPy:** For array manipulation.
- **TensorFlow/Keras:** For GAN model training and deployment.
- **Albumentations:** For dataset augmentation.
- **Streamlit:** For rapid prototyping of the web interface.

### 2. Front-End Technologies:

- **HTML, CSS, JavaScript:** For UI/UX design.
- **React or Vue.js:** For a dynamic and responsive web interface.

### 3. Dataset:

- **Public Datasets:** Use datasets like CUHK Sketch Dataset or Kaggle's paired datasets.
- **Custom Dataset:** Create paired datasets by converting real images to sketches using the pipeline.

### 4. Training Hardware:

- **GPU:** Essential for faster GAN training (e.g., NVIDIA GPUs).
- Use platforms like Google Colab or Kaggle Kernels for free GPU access.

## Implementation Details:

### Step 1: Image to Sketch Conversion

- Implement the Gaussian blur and edge detection pipeline.
- Integrate adjustable parameters for user control.

### Step 2: DCGAN Model Training

- Train on paired datasets.
- Save the trained model for deployment.

### Step 3: Web Interface Development

- Prototype with Streamlit.
- Transition to Flask (back-end) and React (front-end) for production deployment.
- Integrate the trained DCGAN model for real-time inference.

## Expected Outcomes:

1. Users can upload real images and convert them into adjustable pencil sketches.

2. Users can upload pencil sketches and convert them into realistic images using the trained DCGAN model.
3. A functional web interface integrating both processes with options for batch processing and real-time previews.

## Challenges and Solutions:

### 1. Dataset Quality:

- ➓ **Challenge:** Ensuring paired sketches and realistic images are well-aligned.
- ➓ **Solution:** Use high-quality datasets or manually curate the dataset.

### 2. GAN Training Stability:

- ➓ **Challenge:** Training GANs can be unstable.
- ➓ **Solution:** Apply techniques like gradient clipping, batch normalization, and hyperparameter tuning.

### 3. Processing Speed:

- ➓ **Challenge:** Real-time conversion may be slow on low-end hardware.
- ➓ **Solution:** Optimize the model and explore lightweight architectures like MobileNet-based generators.

## Conclusion:

This mini project combines classical image processing techniques and advanced deep learning models to achieve bidirectional image conversion. By incorporating advanced GAN architectures such as Pix2Pix and CycleGAN, the system aims to deliver high-quality outputs. The inclusion of a user-friendly interface ensures accessibility, making it an excellent project for learning and showcasing skills in computer vision, machine learning, and web development.