Unified Framework for Image to Sketch and Sketch to Image Conversions, with Image Retrieval Functionalities from CLIP and Deep Generative Models

Akshat Jain[1] *
Ankit Kumar[1] †
Devansh Panchal[1] ‡
Harshit Goyal[1] §
Pujit Jha[1] ¶

[1]Indian Institute of Technology, Jodhpur
{b22cs007, b22cs076, b22cs021, b22cs024, b22cs091}@iitj.ac.in

Abstract

This project implements a simple image-to-sketch and sketch-to-image conversion system, as well as BLIP-based caption generation and image retrieval from text queries. We created a special Pix2Pix model trained exclusively on faces from the CelebA-HQ dataset [10], which was supplemented by a custom CLIP-based model for non-face images. The technology automatically distributes photos through several processing stages based on facial recognition. For image search, we compare the results of HuggingFace CLIP and our proprietary model trained on the Flicker8k dataset, with ChromaDB optimised for efficient retrieval. The entire system is running on HuggingFace Spaces with a Streamlit UI.

*Roll Number: B22CS007
†Roll Number: B22CS076
‡Roll Number: B22CS021
§Roll Number: B22CS024
¶Roll Number: B22CS091

Contents

1 Overview

1.1 Problem

This project aims to build a unified system for sketch-to-image and image-to-sketch conversion, coupled with intelligent image captioning and a multimodal search engine. Given the gap between abstract sketch inputs and realistic image outputs, and the growing need for content-based retrieval in large image databases, this system proposes a pipeline that solves both generative and retrieval challenges using deep learning and computer vision techniques.

1.2 Pipeline Components

The system has four major components:

- **Image-to-Sketch Conversion:** Utilizes classical computer vision techniques including Sobel, Canny, Dodge, and Lattice filters to convert images into four distinct sketch styles.
- **Sketch-to-Image Generation:** Implements a Pix2Pix model trained on the CelebA-HQ dataset using a UNet-based generator for producing realistic face images. Non-face inputs are routed to a retrieval-based system using CLIP.
- **Image Captioning:** Integrates a BLIP-based vision-language model that generates natural language captions from images.
- **Image Search Engine:** Employs both HuggingFace CLIP and a custom-trained CLIP model for similarity search, supported by ChromaDB for efficient vector-based retrieval from the Flickr8k dataset.

1.3 Applications

This technology has potential applications in:

- Digital art assistance and design prototyping
- E-commerce product visualization from sketches or descriptions
- Law enforcement for refining forensic sketches
- AI-assisted creativity in entertainment and content generation

1.4 Structure of the Report

This report is organized as follows:

- **Section 1:** Overview of the problem and pipeline architecture.
- **Section 2:** Detailed explanation of the four image-to-sketch conversion techniques implemented.

- **Section 3:** Description of the image captioning module using BLIP.
- **Section 4:** Architecture and training methodology of the sketch-to-image Pix2Pix model along with face detection-based routing.
- **Section 5:** Image search framework and optimization using CLIP and ChromaDB.
- **Section 6:** System deployment on HuggingFace Spaces and implementation details.
- **Section 7:** Conclusion and future directions.

2 Image to Sketch Conversion

Our system generates four separate drawing styles from input photos, each built using specific computer vision techniques:

2.1 Sobel Sketch

Sobel edge detection [1] utilizes the Sobel operator and detects edges by calculating the gradient of picture intensity in both the horizontal and vertical axes. It highlights regions with high spatial frequency, which correspond to edges.

- Implementation Details:
  - Converts image to grayscale using OpenCV
  - Applies Sobel operators in X and Y directions (kernel size = 3)
  - Combines gradients using weighted addition
  - Inverts and denoises the result
- Parameters:
  - Sobel kernel size: 3
  - Denoising parameters: h=15, templateWindowSize=7, searchWindowSize=21
- Output: Technical drawing with clear outlines

2.2 Canny Sketch

Canny edge detection [12] is a multi-stage technique that detects a wide variety of edges in photos with great accuracy. It minimises noise, detects gradients, and traces edges using non-maximum suppression, Double thresholding and hysteresis.

- Custom Implementation (not using OpenCV's Canny [5]):
  - Gaussian blur with the parameters (kernel size=5, σ=1.4)
  - Sobel gradient calculation using filter
  - Non-maximum suppression: suppresses the non-maximum values after comparing each pixel with the neighbours in gradient direction
  - Double thresholding using the values (low=5%, high=15% of max gradient)
  - Edge tracking by hysteresis (as explained in the class)
- Output: Clean line drawings with well-defined edges

2.3 Dodge Sketch

The dodge blending technique imitates pencil shading by blending a blurred inverted image with the original greyscale. It emphasises light areas while suppressing dark parts, resulting in a delicate, hand-drawn impression.

- Process Flow:
  1. Convert to grayscale
  2. Invert and apply Gaussian blur (21x21 kernel)
  3. Perform dodge blending: $dodge = \frac{gray}{255-blur} \times 256$
  4. Enhance with Laplacian edge detection (5x5 kernel)
  5. Apply non-local means denoising
- Output: Pencil-like sketch with soft shading

2.4 Lattice Sketch

The Lattice Sketch [7] approach improves mid-frequency details by sharpening and contrast equalisation. It highlights significant textures while minimising noise, resulting in a powerful sketch impression.

- Specialized Processing:
  - Sharpening with custom kernel:

$$\begin{bmatrix} 0 & -1 & 0 \\ -1 & 5 & -1 \\ 0 & -1 & 0 \end{bmatrix}$$

  - Histogram equalization
  - Aggressive denoising (h=75)
- Output: High-contrast sketch preserving mid-frequency details

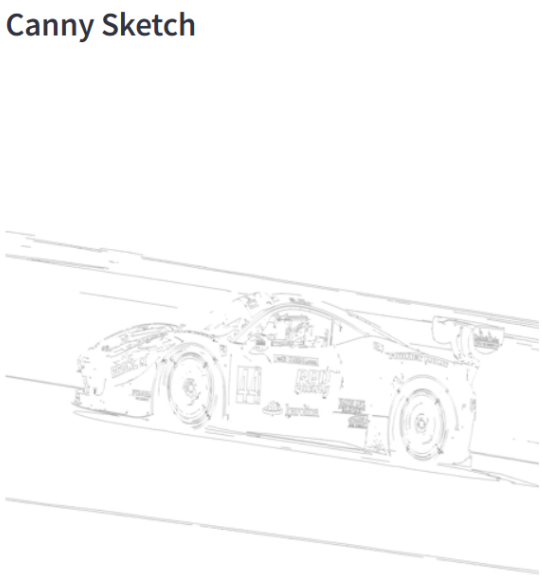
Comparison of Sketch Outputs

Each sketching technique produces a distinct visual effect:

- **Sobel Sketch** produces a technical, blueprint-like drawing with bold contours and crisp edges.
- **Canny Sketch** produces clear and precise line work with minimum noise, making it perfect for edge-focused depictions.
- **Dodge Sketch** creates soft, pencil-like shading with subtle tonal transitions that resemble hand-drawn sketches.

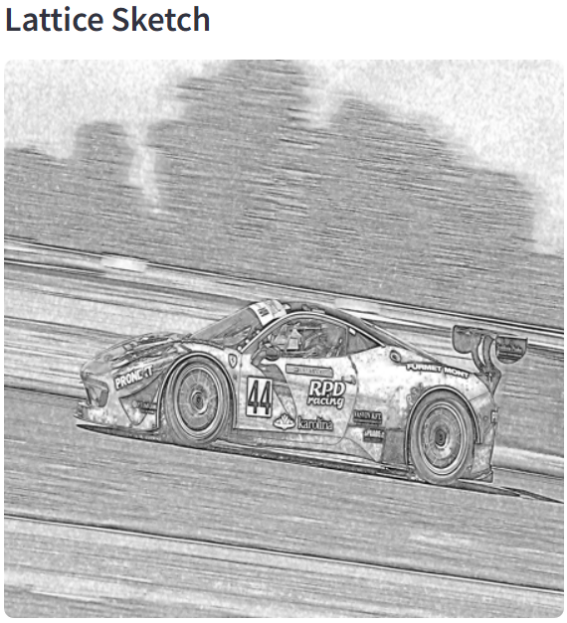
- **Lattice Sketch** emphasises contrast and texture, resulting in vivid, detailed illustrations of mid-frequency patterns.

(a) Sobel sketch



(b) Canny edge detection

(c) Dodge sketch



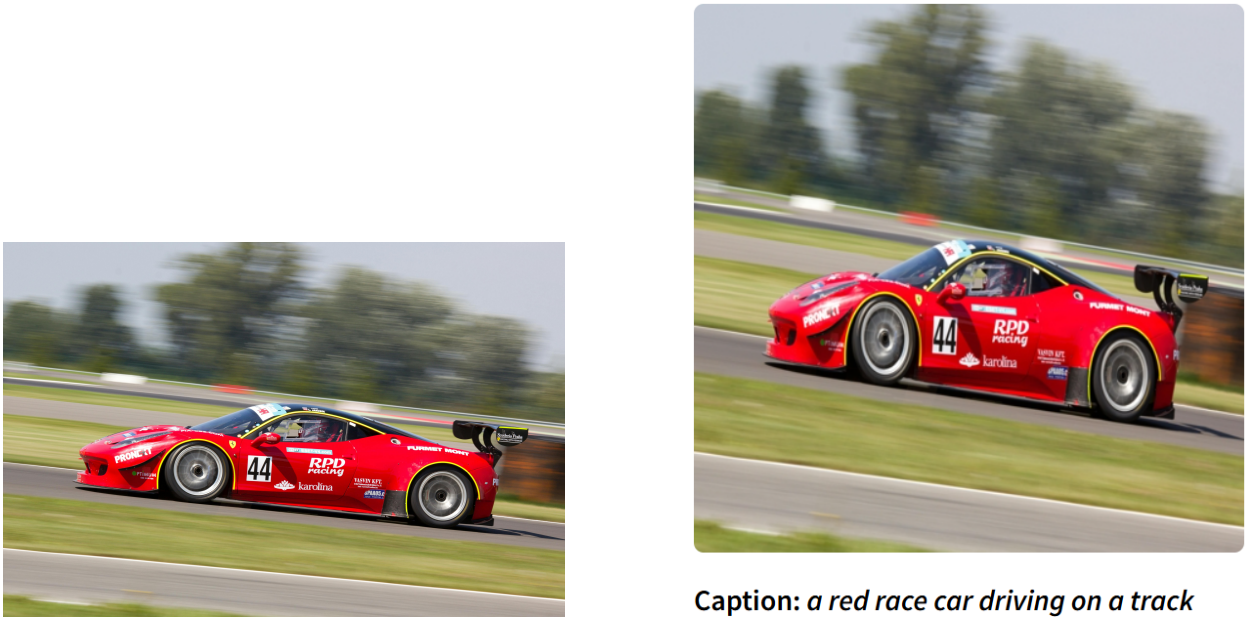
(d) Lattice Sketch

Figure 1: Side-by-side comparison of different sketch generation techniques

3 Image Captioning with BLIP

BLIP (Bootstrapped Language-Image Pretraining) [9] is a vision-language model that can interpret both images and text. It uses visual transformers and text decoders to create correct image descriptions. Our approach makes use of a fine-tuned BLIP model designed specifically for captioning tasks.

- Implementation Details:
  - Uses Salesforce/blip-image-captioning-base model
  - Automatic device detection (CUDA if available)
  - Input: RGB image (automatically converted if needed)
  - Output: Natural language caption
- Processing Flow:
  1. Image converted to RGB
  2. Processed through BLIP processor `inputs = blip_processor(images=image, return_tensors="pt")`
     `inputs = inputs.to(device)`
  3. Caption generated via model.generate()
  4. Decoded with special tokens removed



(a) Original image (b) Image with the caption using BLIP

Figure 2: Working of BLIP

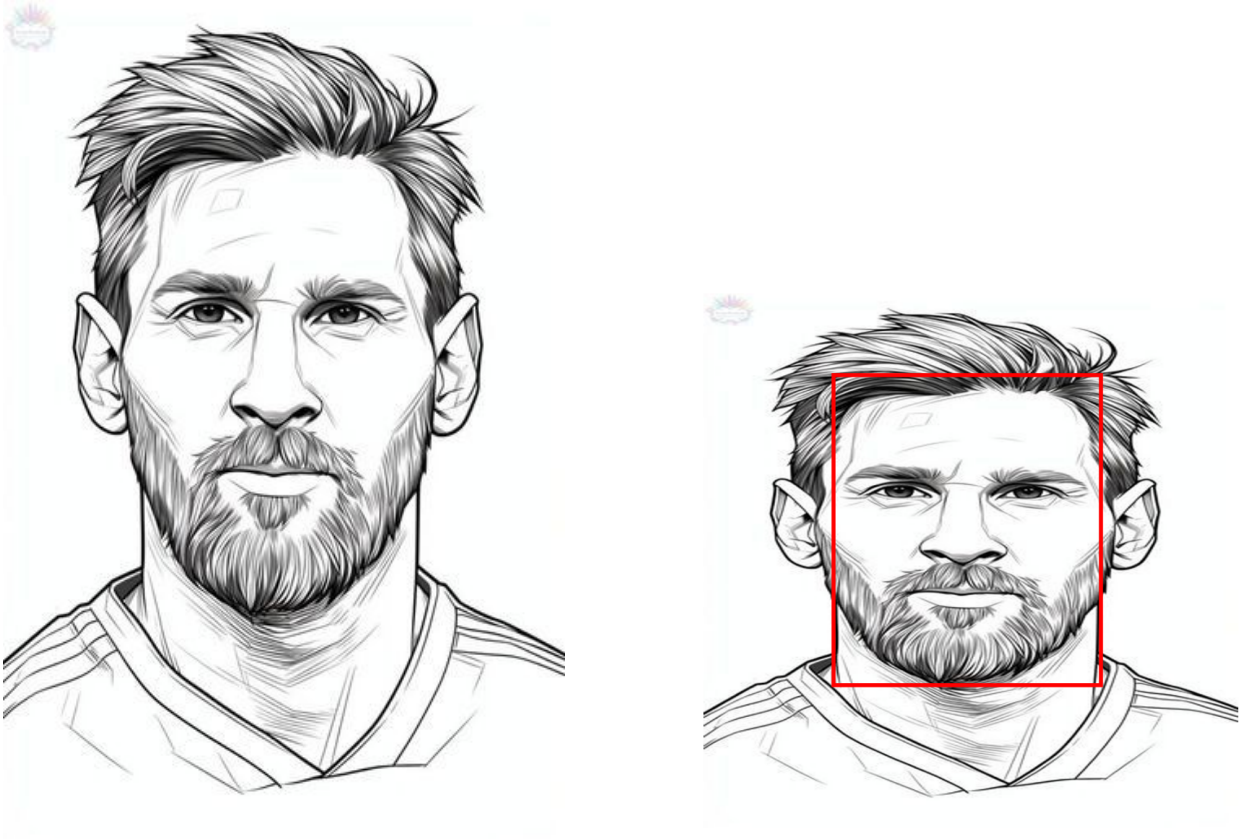
4 Sketch to Image Generation

4.1 Active Research Topic

Sketch-to-image translation remains an active research area in computer vision due to several inherent challenges. Sketches are abstract, sparse, and often incomplete, making the domain gap between sketches and real images difficult to bridge. The problem is also inherently one-to-many, as a single sketch can map to multiple realistic outputs, complicating generation. Additionally, the lack of large-scale paired sketch-photo datasets limits supervised learning. Disentangling structural content from stylistic features like color and texture remains difficult. Finally, evaluating the realism and semantic accuracy of generated images is non-trivial, often requiring subjective or embedding-based metrics.

4.2 Face Detection Routing

- Uses MTCNN for face detection: MTCNN (Multi-task Cascaded Convolutional Networks) [4] is a deep learning-based model for facial recognition and alignment. It uses three steps to recognise faces, refine their locations, and offer essential facial landmarks.
- Routing logic:
  - If face area $\geq 25\%$ of image: Use Pix2Pix (UNetGenerator)
  - Else: Use custom retrieval model



(a) Original Sketch (b) Face detected successfully($\geq 25\%$)

Figure 3

4.3 UNetGenerator Architecture

The UNet-based generator [11] is intended to convert sketches into realistic images that capture both global structure and fine details. It employs an encoder-decoder architecture [6] with skip links to maintain spatial information during reconstruction.

- **Encoder**:

  The encoder converts the input sketch to a compact feature representation by gradually reducing spatial dimensions while increasing the number of channels. This helps to extract high-level semantic information from the input.

  - 7 downsampling blocks (Conv2d + BatchNorm + LeakyReLU)
  - Final bottleneck layer (Conv2d + ReLU)
  - Channel progression: 3→64→128→256→512→512→512→512→512

- **Decoder**: The decoder reconstructs the image from the encoded characteristics by gradually increasing the spatial resolution. Skip connections from the encoder can assist restore fine-grained features that were lost during downsampling.

- 7 upsampling blocks (ConvTranspose2d + BatchNorm + ReLU)
- Skip connections from encoder
- Dropout (p=0.5) in first 3 decoder blocks
- Final layer: ConvTranspose2d + Tanh

4.4 Training Parameters

- Loss: $\mathcal{L} = \mathcal{L}_{GAN} + 100 \times \mathcal{L}_{L1}$
- Optimizer: Adam ($lr = 2e-4$, $\beta_1 = 0.5$)
- Normalization: Input [-1,1], Output Tanh [-1,1]

Figure 4: Pipeline of UNetGenerator Model Working

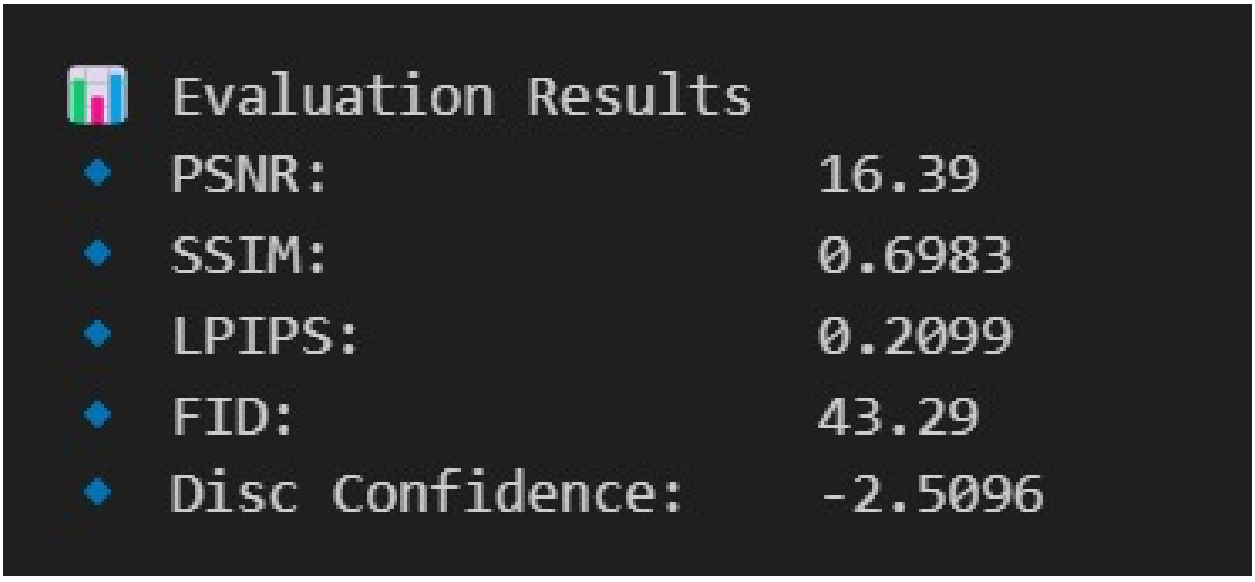
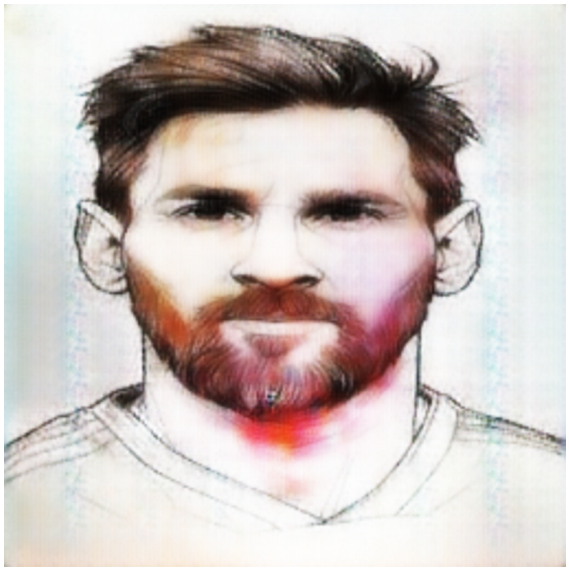


Figure 5: Model Results

(a) Good conversion



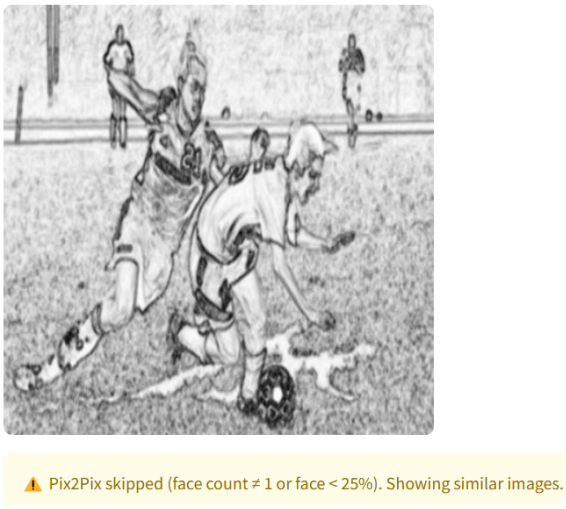
(b) Not a good conversion

Figure 6: Sketch converted to image using UNet Generator

4.5 Custom Retrieval Model

- Used when detected face percentage in input image is below threshold
- Retrieves similar images from Flicker8k Dataset [3]
- Trained on all 8,000 images of the aforementioned dataset using OpenAI CLIP [9]
- Autoencoder architecture with CLIP embeddings
- Uses KNN for finding similar sketch-image pairs

(a) Original Sketch



(b) Face NOT detected($\leq$ 25%)

Figure 7

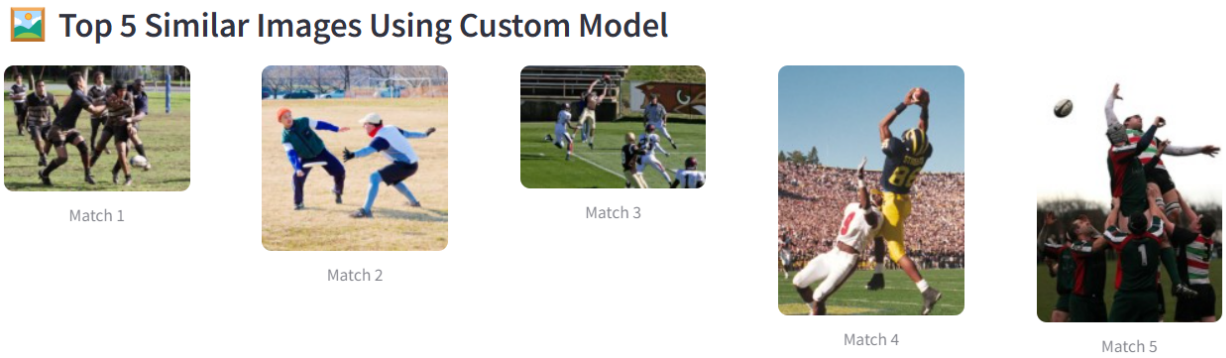


Figure 8: Custom model retrieves similar images

5 Image Search System

5.1 General Approach

- Accepts both text and image queries
- Returns visually/semantically similar images
- Two parallel systems:
  - HuggingFace CLIP implementation
  - Our custom trained model

5.2 HuggingFace CLIP

- Pretrained vision-language model
- Encodes images and text into shared embedding space
- Efficient but sometimes lacks domain specificity

5.3 Custom Model

- Trained on larger dataset than HF CLIP
- More robust to sketch queries
- Uses KNN(k=4) in embedding space

5.4 ChromaDB Optimization

- Stores embeddings of 4,000 Flicker8k images
- Precomputed for fast retrieval
- Eliminates need for per-query training

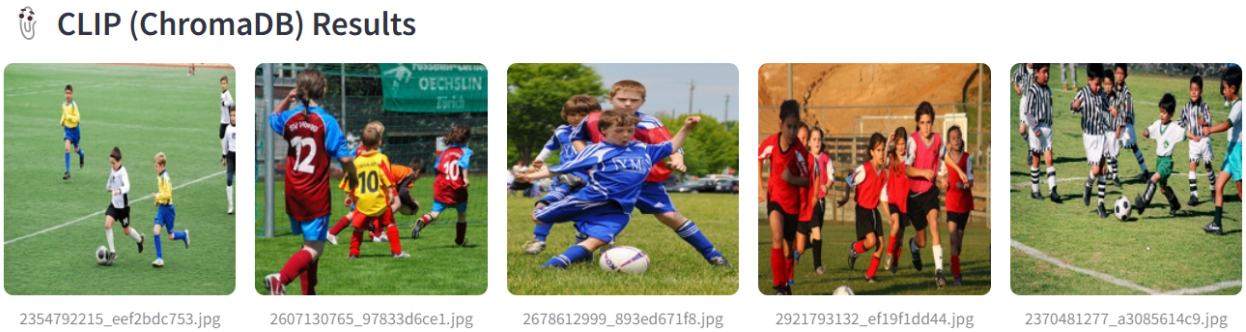


Figure 9: Image search using CLIP when the first image is given as input

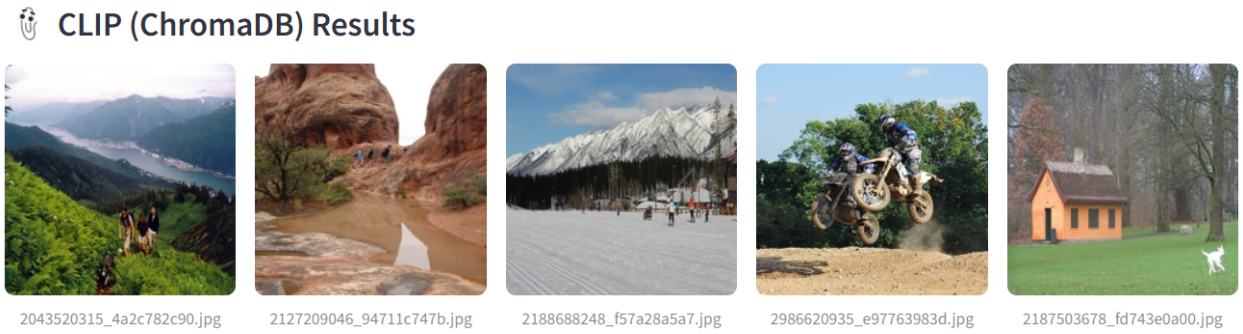


Figure 10: Image search using CLIP when the text entered is ”landscape”

6 Deployment

- At first, the project was deployed using Streamlit [8]. However, due to restricted resources (RAM and runtime limits), the third component (sketch-based retrieval with custom-trained CLIP model) could not be fully deployed on the free Streamlit plan.
- As a result, the deployment was moved to Hugging Face Spaces [13], which better supports machine learning applications and delivers additional processing power, particularly when combined with 'Gradio' or 'Streamlit' interfaces.
- **Application Architecture:**
  - `app.py`: The primary entry point for the web app, coordinating routing across several capabilities (image-to-sketch, sketch-to-image, image search, etc.).
  - `home.py`: Defines the landing page interface, which includes navigating to the various components.
  - `utils.py`: Contains utility functions for model loading, preprocessing, postprocessing, and visualization.
  - `download.py`: Manages on-the-fly dataset downloading and unzipping during app startup to reduce storage usage.
- The custom CLIP-based retrieval system is loaded using pre-trained models serialized with `pickle`, including:
  - `nn_model.pkl`: Trained Nearest Neighbors model for retrieval
  - `image_embeddings.npy`, `sketch_embeddings.npy`: Embedding files used for search
  - `image_paths.pkl`: Stores the paths to image files corresponding to embeddings
- ChromaDB [2] is used for fast similarity search with Hugging Face's CLIP model and runs in the background during the app session. This allows for scalable and fast image/text retrieval using a precomputed index.
- The full project is successfully deployed on Hugging Face Spaces and is accessible at: `https://huggingface.co/spaces/ankitkr1499/Image_To_Sketch_With_Content_Based_Feature_Extraction`

Summary

This project presents a unified system for bi-directional sketch and image transformation, as well as robust image search capabilities. The pipeline begins with converting images to sketches using four classical computer vision techniques: Sobel, Canny, Dodge, and Lattice, each producing distinct artistic outputs. It also supports sketch-to-image generation via a custom-trained Pix2Pix model based on UNet architecture, tailored specifically for face images from the CelebA-HQ dataset. A face detection mechanism using MTCNN intelligently routes input through either the generative model or a custom image retrieval system. Additionally, the system integrates BLIP for image captioning and leverages both HuggingFace CLIP and a proprietary CLIP-based model trained on Flickr8k for semantic search. ChromaDB is employed to ensure scalable and efficient retrieval. The entire application is deployed on HuggingFace Spaces with a Streamlit-based UI, providing a real-time, user-friendly platform for experimentation and visual exploration.

Conclusion

The developed system delivers a comprehensive and modular framework that unifies traditional image processing techniques with modern deep learning architectures for creative visual tasks. From flexible sketch generation to realistic image synthesis and intelligent image search, the pipeline encapsulates both artistic and utilitarian aspects of vision-based systems. By dynamically routing inputs based on content and deploying optimized models for search and generation, the project achieves a balance between accuracy, speed, and versatility. Future improvements could focus on expanding the dataset, refining the sketch-image pairing process, enhancing the routing criteria, and exploring transformer-based generation methods for more generalized and photorealistic outputs.

References

[1] Erhan Arslan. Exploring edge detection in python: 2–sobel edge detector: A closer look. https://medium.com/@erhan_arslan/exploring-edge-detection-in-python-2-sobel-edge-detector-a-closer-look-de051a7b56df 2023. Accessed: 2025-04-12.

[2] Chroma. Chroma documentation. https://docs.trychroma.com/docs/overview/getting-started, 2025.

[3] Aditya Jain. Flickr8k dataset. https://www.kaggle.com/datasets/adityajn105/flickr8k, 2025.

[4] Ahmad Jawabreh. Multi-task cascaded convolutional neural network (mtcnn). https://medium.com/the-modern-scientist/multi-task-cascaded-convolutional-neural-network-mtcnn-a31d88f501c8, 2023. Accessed: 2025-04-12.

[5] OpenCV. Canny edge detection tutorial. https://docs.opencv.org/4.x/da/d22/tutorial_py_canny.html, 2025.

[6] Ahmad Sabry. A perfect guide to understand encoder-decoder models in-depth with visuals. https://medium.com/@ahmadsabry678/a-perfect-guide-to-understand-encoder-decoders-in-depth-with-visuals-30805c23659b, 2025.

[7] Anish Sachdeva. image2sketch: Converts a 2d color image into a hand-drawn sketch using a novel technique. https://github.com/anishLearnsToCode/image2sketch, 2025. Accessed: 2025-04-12.

[8] Streamlit. Streamlit documentation. https://docs.streamlit.io/, 2025.

[9] Az. Tayyebi. Bridging vision and language: Exploring clip, blip, and owl-vit. https://medium.com/@az.tayyebi/bridging-vision-and-language-exploring-clip-blip-and-owl-vit-f8f6a99a263e, 2025.

[10] Badass Techie. Celeba-hq resized 256x256 dataset. https://www.kaggle.com/datasets/badasstechie/celebahq-resized-256x256, 2025.

[11] TensorFlow. Pix2pix tutorial. https://www.tensorflow.org/tutorials/generative/pix2pix, 2025.

[12] Author Unknown. Cv 2025 lecture 4. https://drive.google.com/file/d/1cbOtQGFtZAWpuHcOjikitekHnT46GO6J/view, 2025. Accessed: 2025-04-12.

[13] YouTube. Video tutorial. https://www.youtube.com/watch?v=dOJNrlPdlW4, 2025.

Contributions

- **Akshat Jain**: Led the integration of Pix2Pix for sketch-to-image conversion, implemented the UNet Generator architecture, and managed system deployment
- **Ankit Kumar**: Developed and fine-tuned the Pix2Pix model; contributed significantly to the drafting and refinement of the report
- **Devansh Panchal**: Implemented Sobel and Canny edge detection algorithms; integrated BLIP for automated image captioning
- **Harshit Goyal**: Engineered lattice and dodge sketch generation techniques; implemented face detection using MTCNN and assisted with report composition
- **Pujit Jha**: Designed and trained custom retrieval models; developed the image search pipeline, coordinated system deployment, and produced the demonstration video