Assignment-1

December 13, 2025

1 Problem statement

Tou are given a natural image (Mario image provided). Your objective is to detect and visualize the edges present in the image using only basic convolution techniques.
You are not allowed to use any built-in edge detection functions such as `cv2.Canny`, `cv2.Sobel`, or `cv2.Laplacian` as single-line APIs. The goal is to understand how edge detection actually works at a low level.
Your final output should be a binary edge map (edges in white, background in black).

Constriants

- You must implement edge detection only using convolution.
  You may use:
- Grayscale conversion
- Smoothing filters
- Hand-defined convolution kernels
- `cv2.filter2D` for performing convolution
- Basic NumPy operations
- You may not call ready-made edge detector functions.

Hints A stable edge detection pipeline typically begins by converting the input image to grayscale, followed by applying a smoothing (low-pass) filter to reduce noise before computing derivatives. Common smoothing filters include simple averaging (box) filters or Gaussian filters. Once the image is smoothed, edges can be detected by convolving the image with derivative kernels that capture intensity changes, such as first-order gradient operators (e.g., Sobel or Prewitt) or second-order operators like the Laplacian. These kernels must be defined manually and applied using a convolution function such as cv2.filter2D. The resulting response represents edge strength and usually lies in an arbitrary numeric range, so it is helpful to normalize the output and apply a suitable threshold to obtain a clear binary edge map. You may use online coding assistants to help with implementation, but you are expected to understand and be able to explain the purpose of every line of code you write

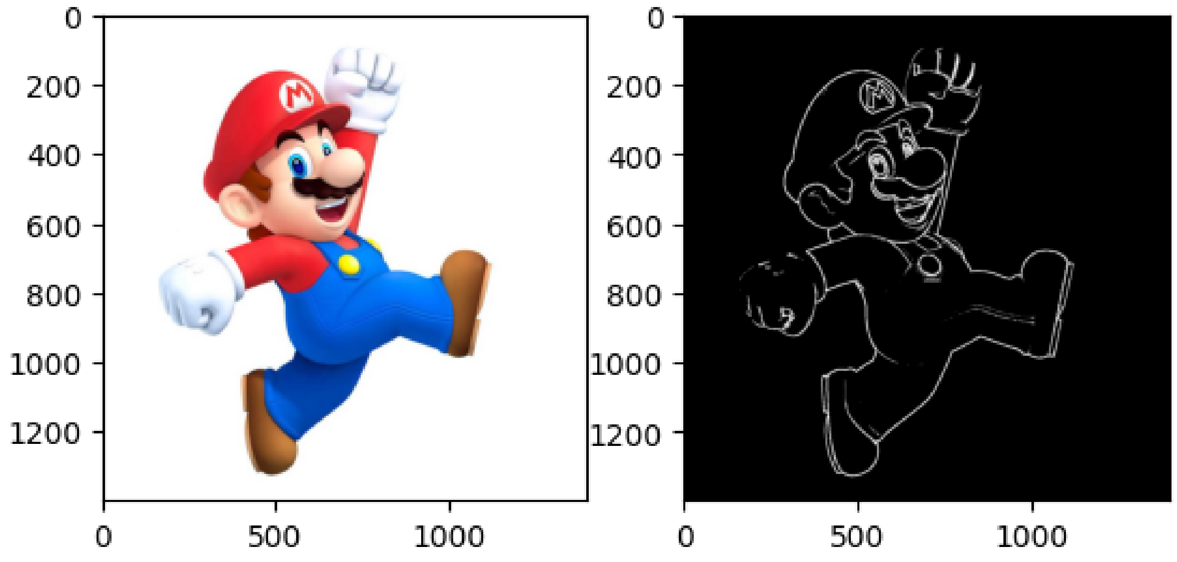

Figure 1: Input and sample output