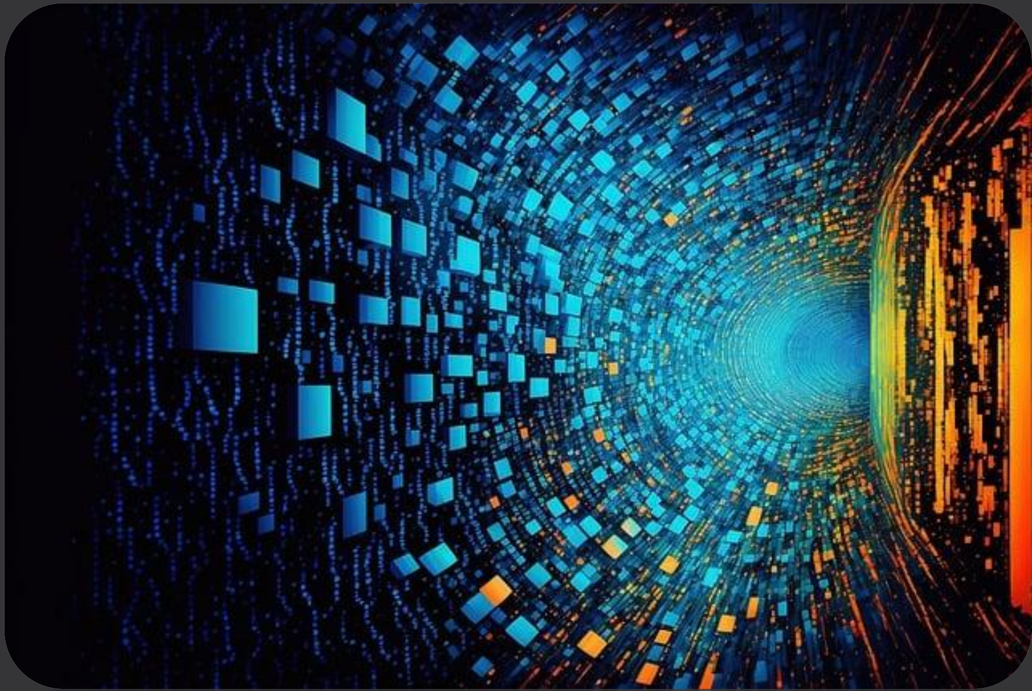
Image Data Visualization

Data Visualization Course – Lecture 6



Dr. Muhammad Sajjad

R.A: Imran Nawar

December 2024

Overview

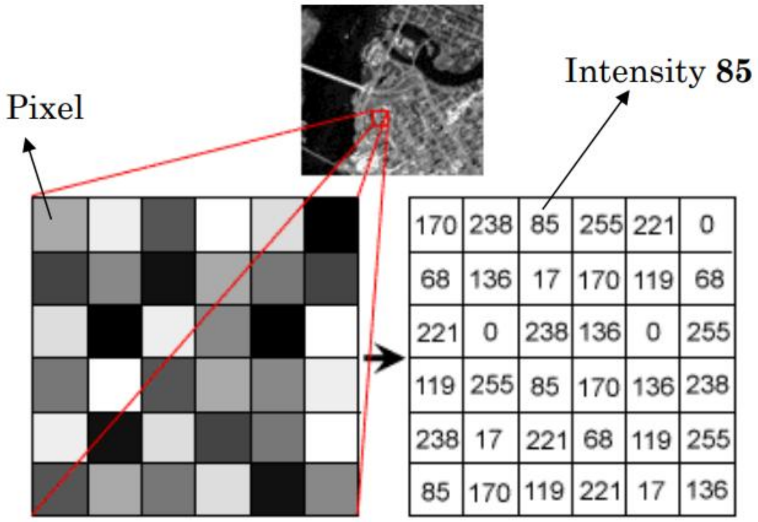
- **Introduction to Image Data**
  - What is Image Data?
  - Digital Image Representation (pixels, channels, bit depth)
  - Common Image Formats (RGB, grayscale, binary)
- **Role of Image Data in Data Science**
- **Image Processing Tools in Python**
  - Overview of Key Libraries
    - OpenCV
    - Pillow/PIL
  - When to use each library
- **Basic Image Operations**
  - Loading and reading images
  - Color Spaces and conversions
  - Image properties, metadata and Basic transformations (resize, rotate, flip)
- **Advanced Visualization Techniques**
  - Histograms and Color Distribution
  - Image Enhancement Techniques (Filtering techniques, noise reduction, edge detection)
  - Multi-image Visualization (Subplots and layouts, Side-by-side comparisons)
- **Practical Applications**
  - Visualizing image classification, object detection and segmentation results
- **Coding:**
  - Practical Coding examples for the topics

Introduction to Image Data

Image Data

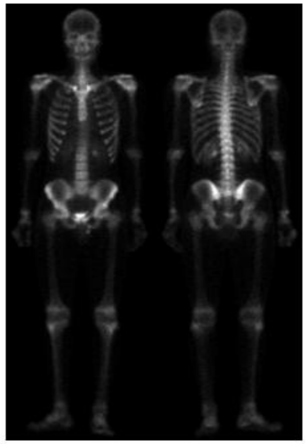
- Image data consists of pixel grids where each pixel encodes intensity or color values.
- Images stored as structured numerical arrays that computers can process.

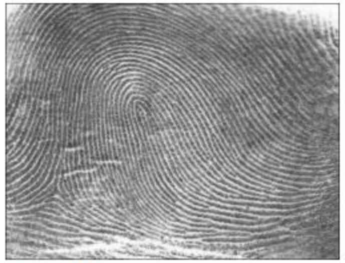
- **Key Characteristics**:
  - **Pixel Grid**: A 2D/3D numerical matrix.
  - **Representation**: Grayscale (single channel) or RGB (three channels).
  - **Range**: Pixel values typically range from 0–255 (8-bit images).
- **Core Components**:
  - **Pixels**: Smallest unit of an image grid.
  - **Channels**: Layers of pixel data (e.g., Red, Green, Blue in RGB).
  - **Dimensions**: 2D (grayscale), 3D (color), or more (multispectral).
- **Example**: A grayscale X-ray represented as a 2D matrix with pixel values ranging 0–255
- **Applications**:
  - Foundational to fields like computer vision, medical imaging, and satellite analysis.
  - Powers applications in AI, robotics, and automation.



170	238	85	255	221	0
68	136	17	170	119	68
221	0	238	136	0	255
119	255	85	170	136	238
238	17	221	68	119	255
85	170	119	221	17	136

X-ray imaging


gamma-ray imaging

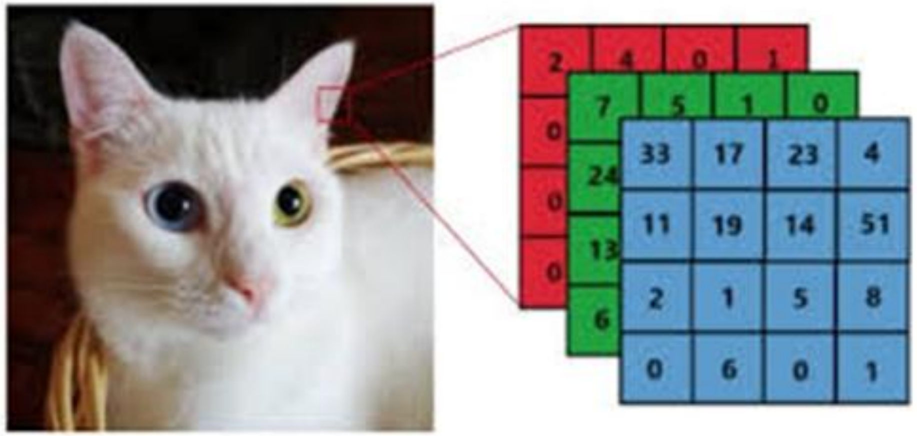

Visible spectrum
Security, Biometrics

Introduction to Image Data

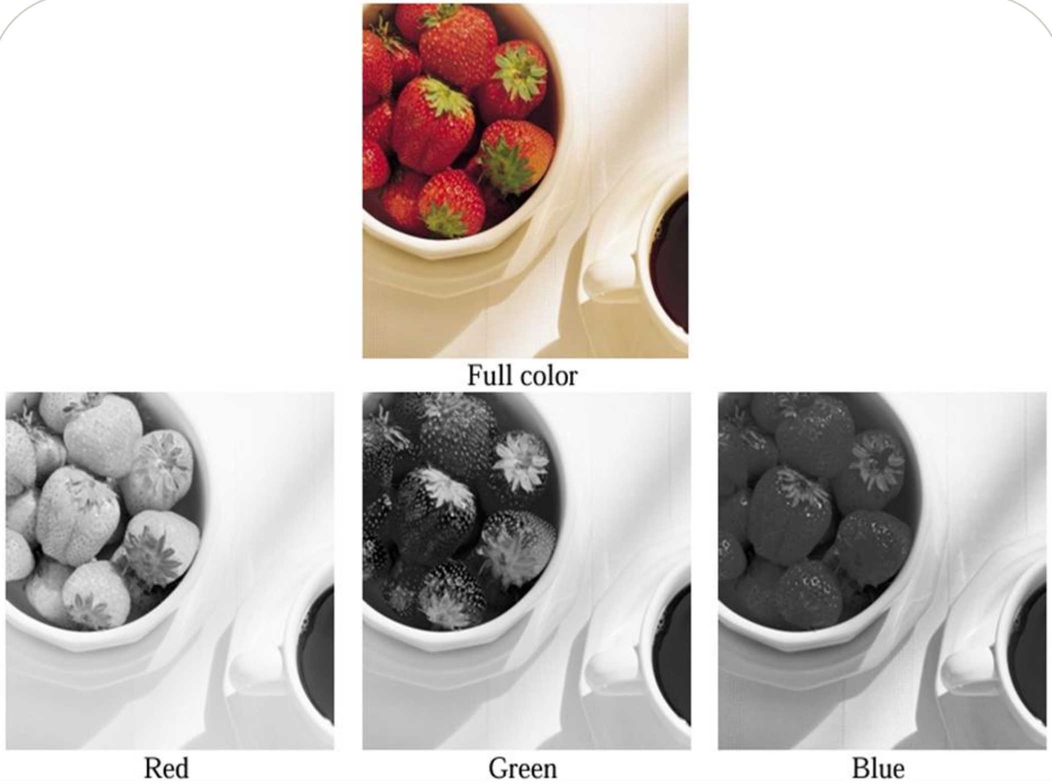
Digital Image Representation

- **Pixels**: Basic building blocks of an image.
  - Intensity in grayscale; color in RGB/multispectral formats.
- **Channels**: Number of data layers in an image.
  - **Single-channel**: Grayscale.
  - **Three-channel**: RGB (color images).
  - **Multispectral**: Beyond visible light (e.g., infrared).
- **Bit Depth**: Determines pixel value range.
  - **8-bit**: 0–255.
  - **16-bit**: 0–65535.
  - **24-bit:** 0–16,777,216 (color images)
- **Visualization**:
  - **Grayscale image:** Single matrix.
  - **RGB image:** Three overlapping grids (R, G, B).

Color Image – Three Channels



33	17	23	4
11	19	14	51
2	1	5	8
0	6	0	1


Full color

Red

Green

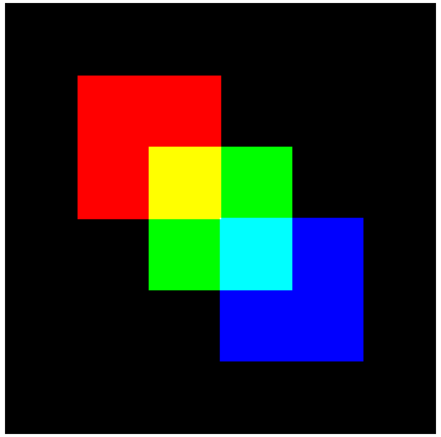
Blue

RGB Color Channels of a Full-Color Image

Introduction to Image Data

Common Image Formats

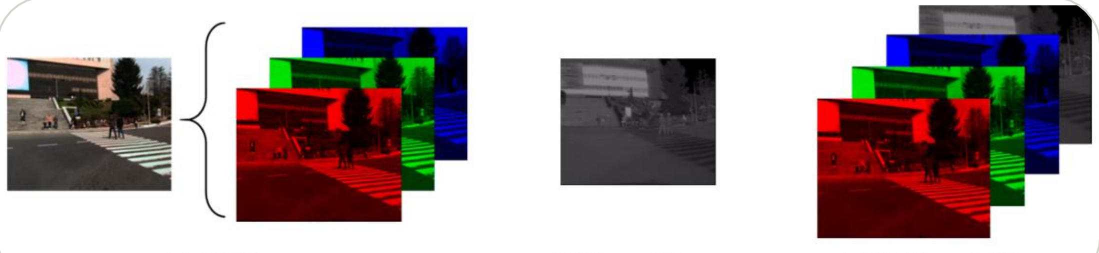
- **Grayscale Images**
  - Single-channel; values 0 (black) to 255 (white).
  - Applications: Medical imaging (X-rays), document scanning
  - Example: Scanned documents
- **RGB Images**
  - Three channel color representation (Red, Green, Blue)
  - Each pixel represented by three values (0-255)
  - Common format for digital photography and computer graphics
  - Applications: Photography, web graphics.
  - Example: Digital photographs
- **Binary Images**
  - Two-states: 0 (black) or 1 (white).
  - Result of image thresholding operations.
  - Applications: OCR, edge Detection.
  - Example: Thresholded images.
- **Multispectral Images**
  - Multiple channels beyond visible spectrum
  - Applications: Satellite imagery, remote sensing, climate analysis.
  - Example: Infrared maps.



RGB Image

Gray Scale Image

Binary Image



(a) RGB Image

(b) Thermal Image

(c) Multispectral Image

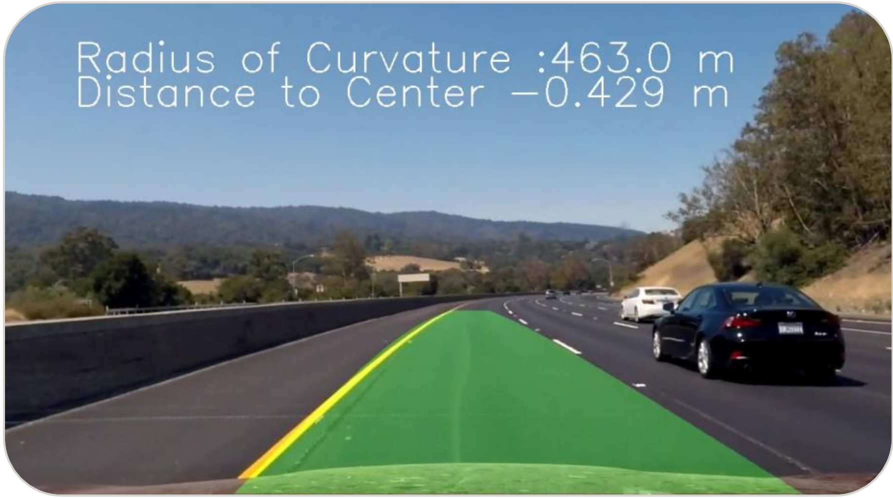
Role of Image Data in Data Science

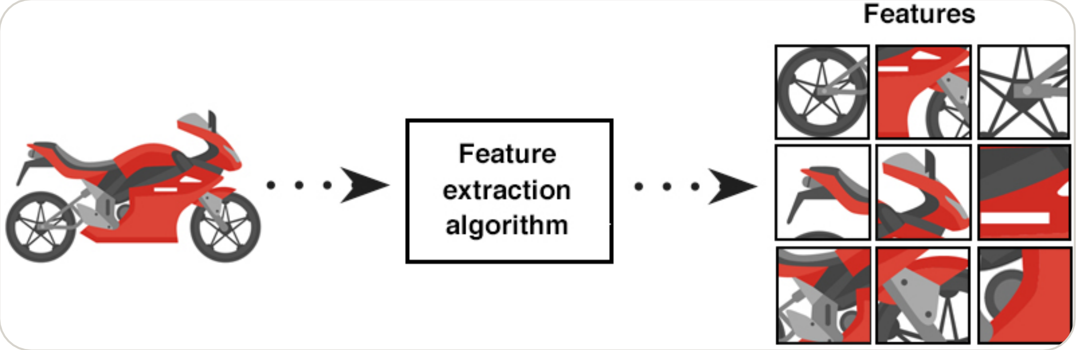
Applications in Data Science

- **Computer Vision:**
  - **Object Detection:** (e.g., pneumonia detection in X-rays).
  - **Image Segmentation** (e.g., lane detection for autonomous vehicles).
  - **Feature Extraction** for ML models.
- **Medical Image Analysis:**
  - Disease detection (e.g., tumors) and diagnosis support.
- **Remote Sensing:**
  - Environmental monitoring, urban planning.

- **Benefits of Visualizing Image Data:**
  - Explore data distributions (e.g., intensity histograms).
  - Detect anomalies (e.g., noise, missing data).
  - Communicate insights effectively.

Car Lane Detection for Self-Driving Cars





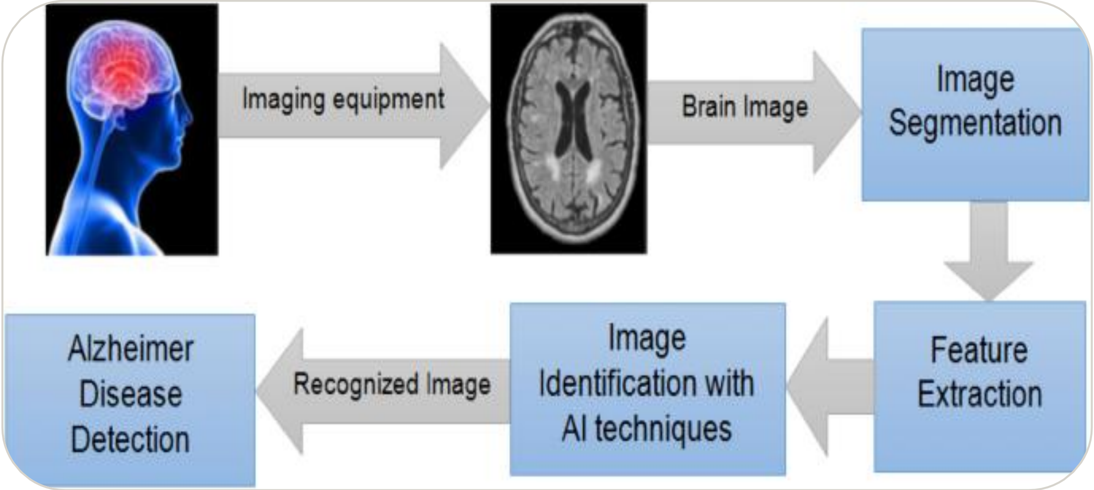


Image Processing Tools in Python

Overview of Key Libraries

1) OpenCV – Advanced Image Processing in Python

- A comprehensive library for advanced image processing and computer vision tasks.

Key Features:

- **Speed**: Optimized for large-scale operations using matrix manipulation.
- **Color Formats**: Supports BGR (default) and conversions to other formats.
- **Advanced Tools**: Edge detection, face detection, object tracking.

Common Functions:

- `cv2.imread()`: Read images.
- `cv2.imshow()`: Display images.
- `cv2.cvtColor()`: Convert between color spaces.

Use Cases:

- Real-time video analysis.
- Object detection and recognition.
- Computer vision applications.

Why to Learn OpenCV

Image Processing
Feature Detection & Matching
Object Detection
Geometry
Machine Learning
Video Analysis
GUI Tools
Integration
NumPy
TensorFlow

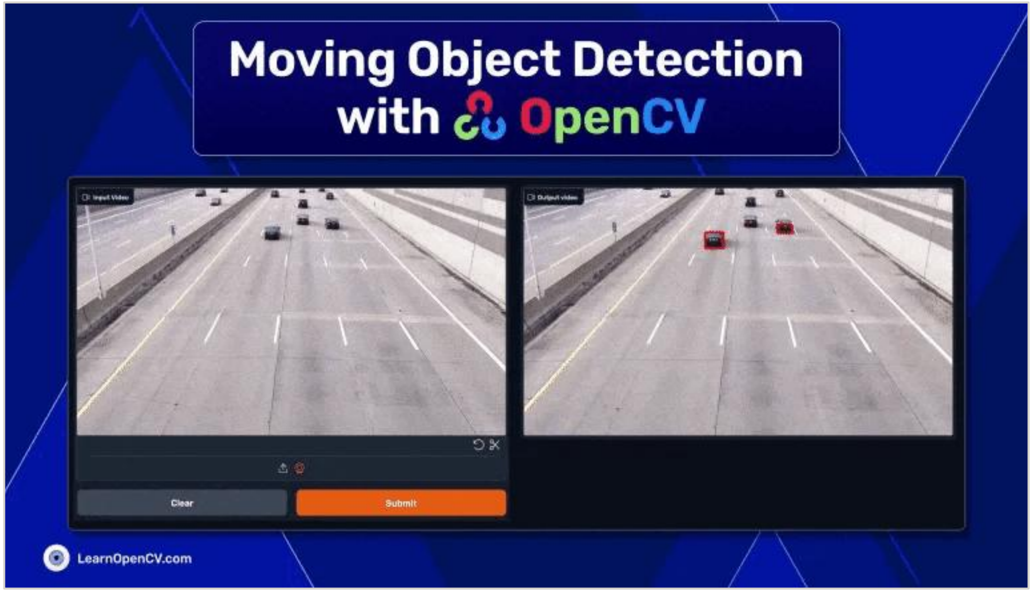


Image Processing Tools in Python

2) Pillow – Simplified Image Manipulation

- A user-friendly library for basic image manipulation and editing.

Key Features:

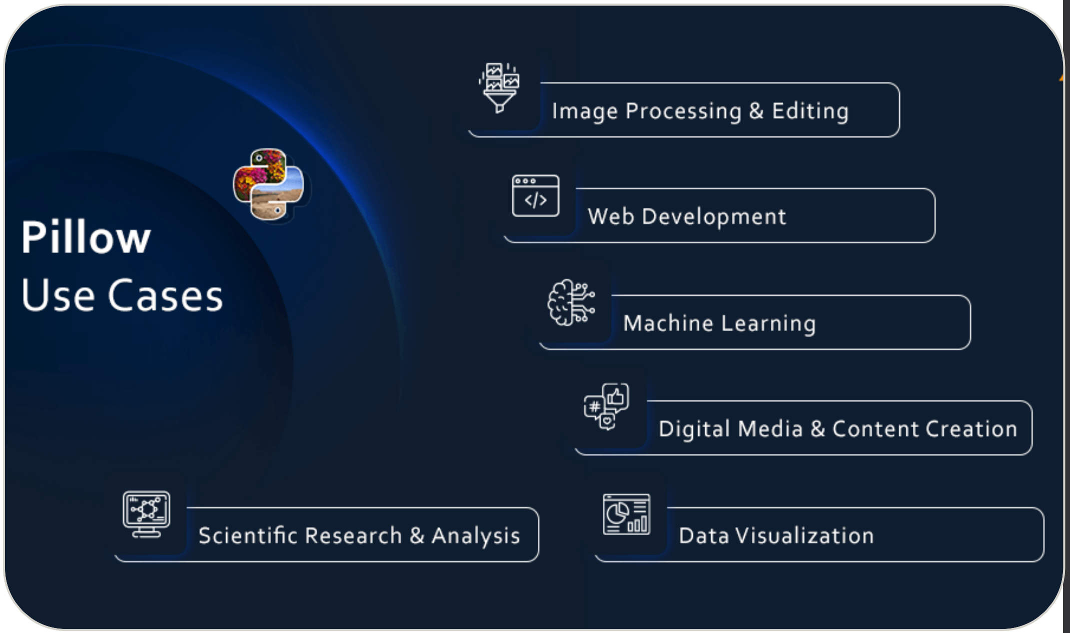
- **Ease of Use:** Ideal for quick, simple operations.
- **Format Support:** Handles various formats (e.g., PNG, JPEG, BMP).
- **Color Handling:** Uses standard RGB color format.

Common Functions:

- `Image.open()`: Open image files.
- `Image.show()`: Display images.
- `Image.resize()`: Resize images.
- `Image.convert()`: Convert color modes (e.g., RGB to grayscale).

Use Cases:

- Basic editing: Resizing, cropping, and filtering.
- Preparing images for visualization.
- Converting image formats.



Pillow
Use Cases

Image Processing & Editing

Web Development

Machine Learning

Digital Media & Content Creation

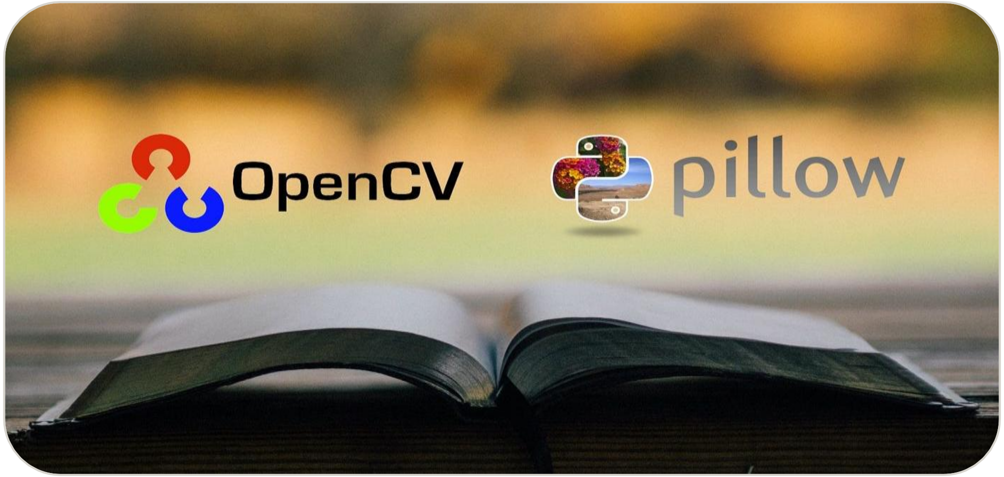
Scientific Research & Analysis

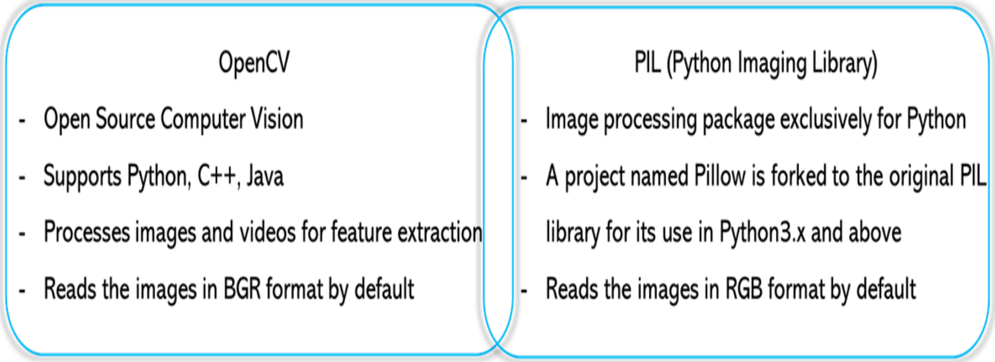
Data Visualization

Image Processing Tools in Python

When to Use OpenCV vs. Pillow

- **OpenCV**:
  - Best for **complex tasks** like real-time processing, edge detection, and object recognition.
  - Suitable for **large-scale applications** with performance-critical needs.
  - **Examples:** Autonomous driving, facial recognition, AR/VR development.
- **Pillow**:
  - Ideal for **simple tasks** like resizing, cropping, or format conversion.
  - Useful for quick prototyping and lightweight processing.
  - **Examples:** Generating thumbnails, basic photo editing, integrating images into web apps.
- **Key Comparison**:
  - **Performance**: OpenCV is faster and better for advanced tasks.
  - **Ease of Use**: Pillow is simpler for beginners and basic needs.





OpenCV

- Open Source Computer Vision
- Supports Python, C++, Java
- Processes images and videos for feature extraction
- Reads the images in BGR format by default

PIL (Python Imaging Library)

- Image processing package exclusively for Python
- A project named Pillow is forked to the original PIL library for its use in Python3.x and above
- Reads the images in RGB format by default

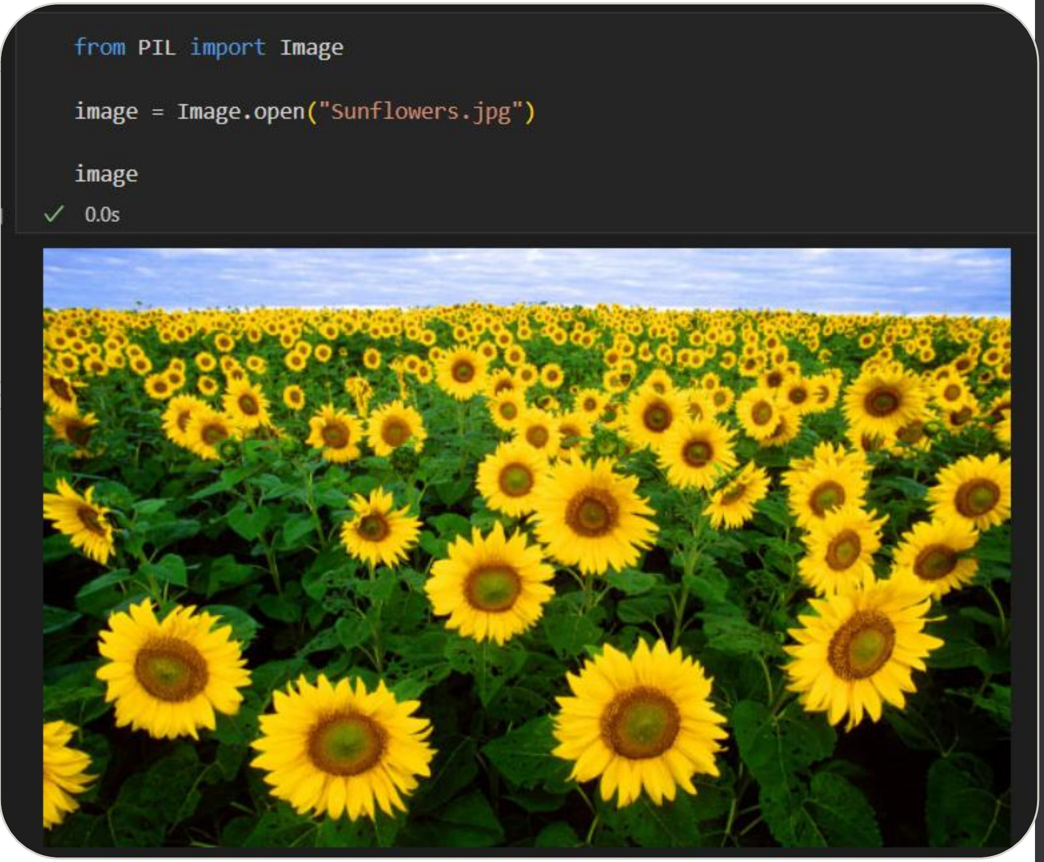
Basic Image Operations

Loading and Reading Images

The foundational step in image processing, involves loading a displaying images for analysis and manipulation.

- **Key Functions**:
  - **OpenCV**:
    - `cv2.imread()`: Read an image from a file.
    - `cv2.imshow()`: Displays the image in a pop-up wind for inspection.
  - **Pillow**:
    - `Image.open()`: Opens image files for processing.
- **Supported Formats**: JPEG, PNG, BMP, TIFF, and more.
- **Applications**:
  - Display image datasets for ages for quality inspection.
  - Load training machine learning models.
- **Example Code:**

```
import cv2
image = cv2.imread('example.jpg')
cv2.imshow('Loaded Image', image)
cv2.waitKey(0)
cv2.destroyAllWindows()
```



```
from PIL import Image

image = Image.open("Sunflowers.jpg")

image
```

✓ 0.0s

Basic Image Operations

Color Spaces and Conversions

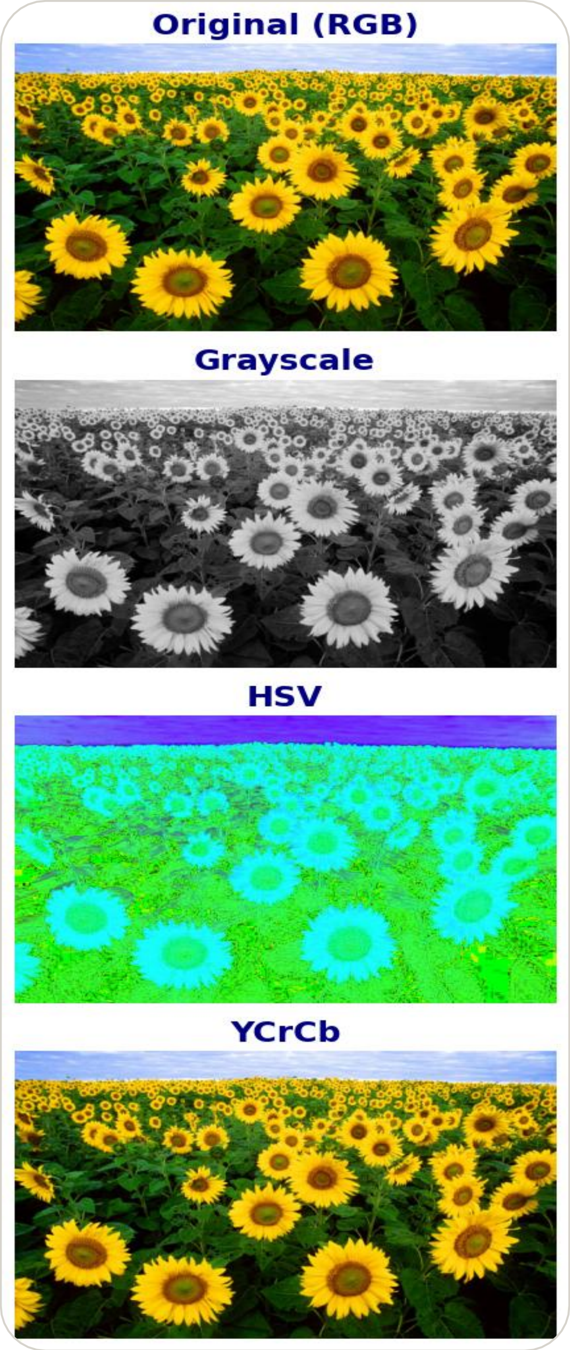
Color spaces define how colors are represented and interpreted in an image (e.g., RGB, BGR, Grayscale, HSV).

➢Key Functions:

- **OpenCV**:
  - `cv2.cvtColor()`: Convert between color spaces.
    - Example: RGB ↔ Grayscale, RGB ↔ HSV.
- **Pillow**:
  - `Image.convert()`: Change color mode (e.g., RGB to Grayscale).

➢Applications:

- Convert to Grayscale for edge detection.
- Use HSV for color-based object segmentation.
- Prepare images for algorithms requiring specific color formats.



Original (RGB)

Grayscale

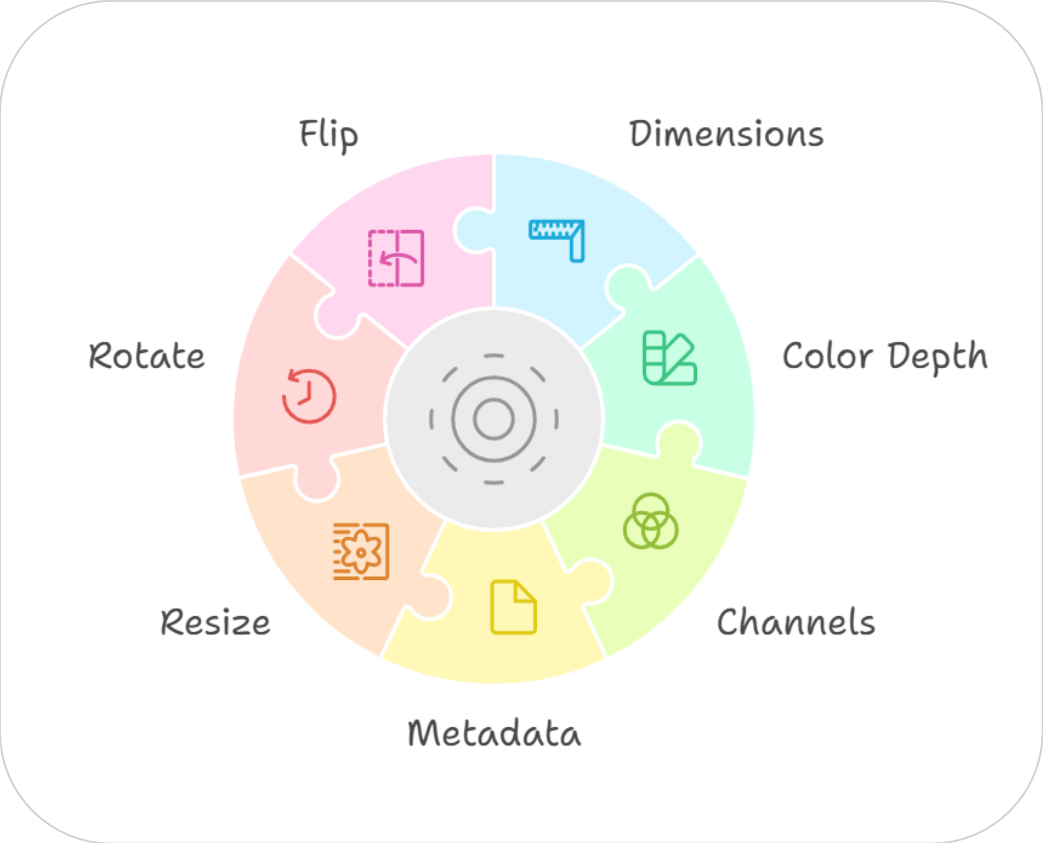
HSV

YCrCb

Basic Image Operations

Image Properties, Metadata, and Basic Transformations

- **Image Properties**:
  - **Properties**: Dimensions (height, width), color depth, and channels.
    - OpenCV: Use `image.shape`.
    - Pillow: Use `image.size` and `image.mode`.
  - **Metadata**: Includes resolution, file format, EXIF data (e.g., camera info, GPS).
- **Basic Transformations**:
  - **Resize**: Change image dimensions.
    - OpenCV: `cv2.resize()`.
    - Pillow: `Image.resize()`.
  - **Rotate**: Rotate the image by a specified angle.
    - OpenCV: `cv2.getRotationMatrix2D()` and `cv2.warpAffine()`.
    - Pillow: `Image.rotate()`.
  - **Flip**: Create mirrored images.
    - OpenCV: `cv2.flip()` for horizontal/vertical flipping.
    - Pillow: `Image.transpose()` with options like FLIP_LEFT_RIGHT.
- **Examples**:
  - **Resize:** Downscale images for faster processing.
  - **Rotate:** Align images for better visualization.
  - **Flip:** Data augmentation for ML models.



Advanced Visualization Techniques

Histograms and Color Distribution

Purpose:

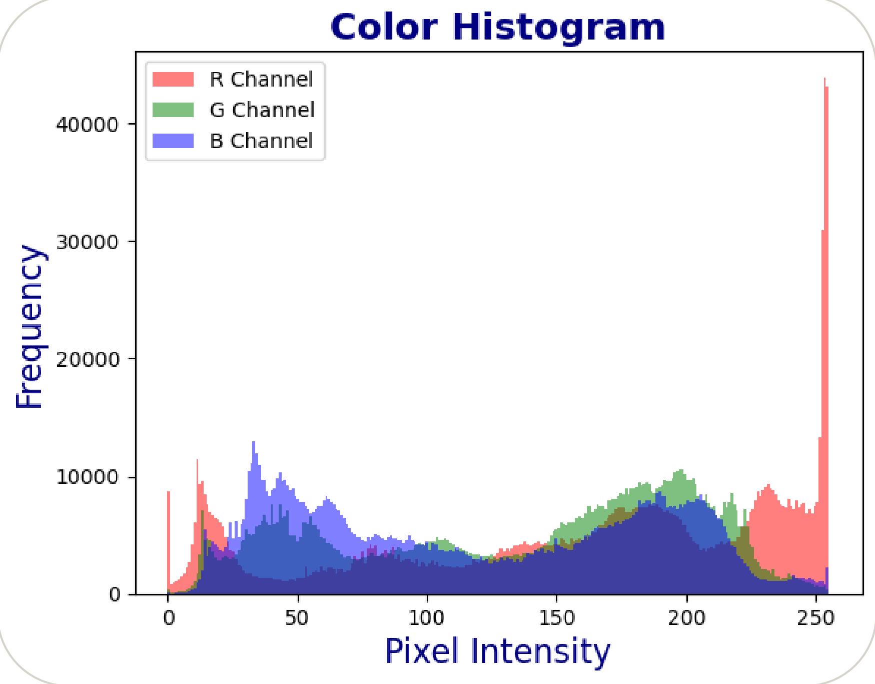
- Visualize the pixel intensity or color distribution in an image.
- Analyze image properties such as brightness, contrast, and color balance.

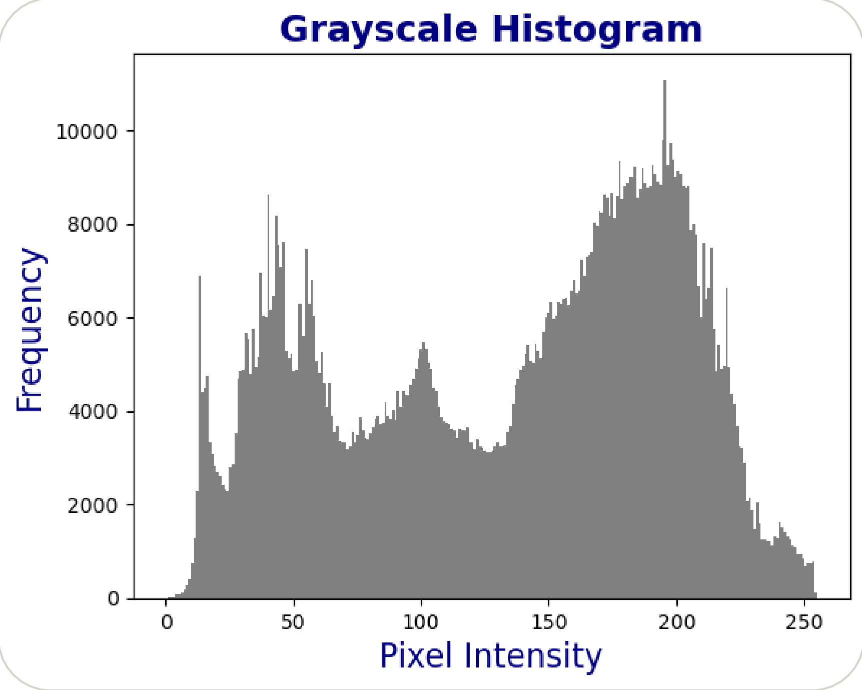
Key Points:

- **Grayscale Histogram:** Shows the frequency distribution of pixel intensities in a grayscale image.
- **Color Histogram:** Displays the intensity distributions for each channel (Red, Green, Blue) in an RGB image.
- **Applications:**
  - Identify overexposed or underexposed regions in an image.
  - Understand the balance of colors for editing or feature extraction.

Code Snippet (Overview):

```
# Generate histograms for RGB image
colors = ['r', 'g', 'b']
for i, color in enumerate(colors):
    channel = image[:, :, i]
    plt.hist(channel.ravel(), bins=256, color=color, alpha=0.5)
```





Advanced Visualization Techniques

Image Enhancement Techniques

➢ Filtering Techniques

1) **Smoothing**: Reduces noise and minor intensity variations.
    - Example: Gaussian blur.
2) **Sharpening:** Highlights edges and improves clarity
    - Example: Unsharp masking.

➢ Noise Reduction

- Techniques like median and Gaussian filtering to minimize random noise.

➢ Edge Detection

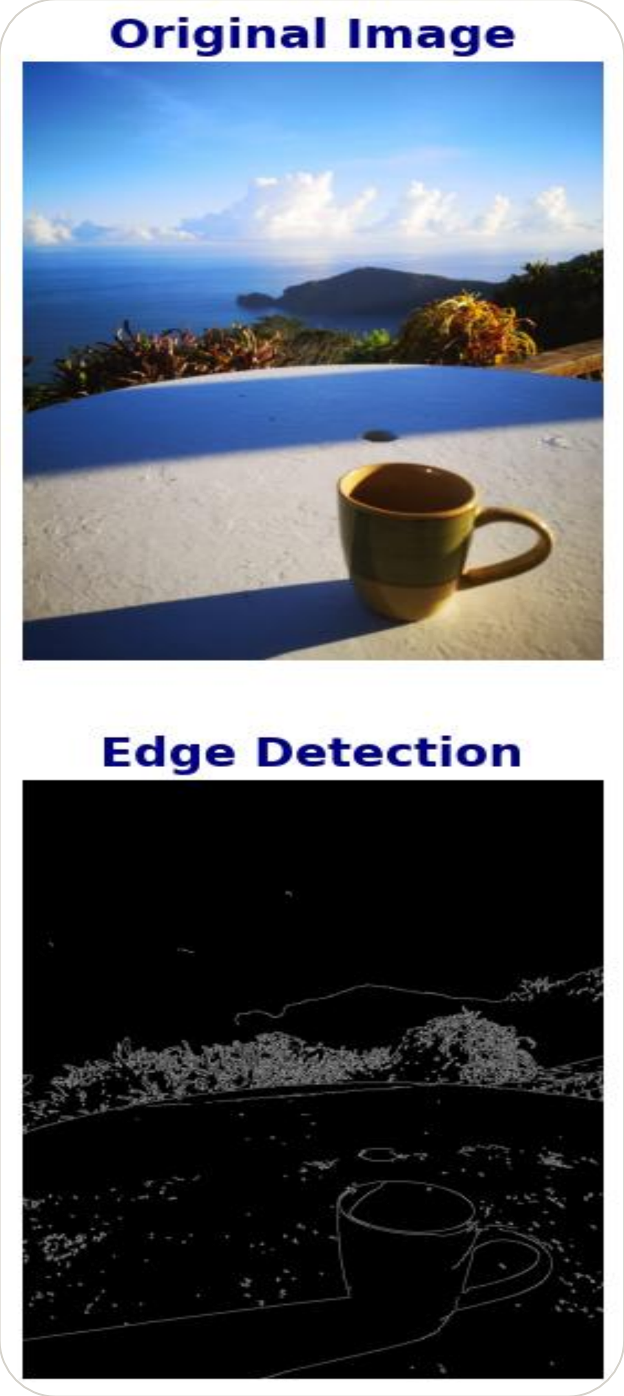
- Algorithms such as Sobel and Canny detect significant intensity transitions (edges).

➢ Applications:

- Preprocessing images for computer vision models.
- Highlighting critical features in medical or satellite imagery.

Code Snippet:

```
# Canny edge detection
edges = cv2.Canny(image, 100, 200)
```



Advanced Visualization Techniques

Multi-Image Visualization

- **Purpose:** Compare and analyze multiple images or transformations in a single view.

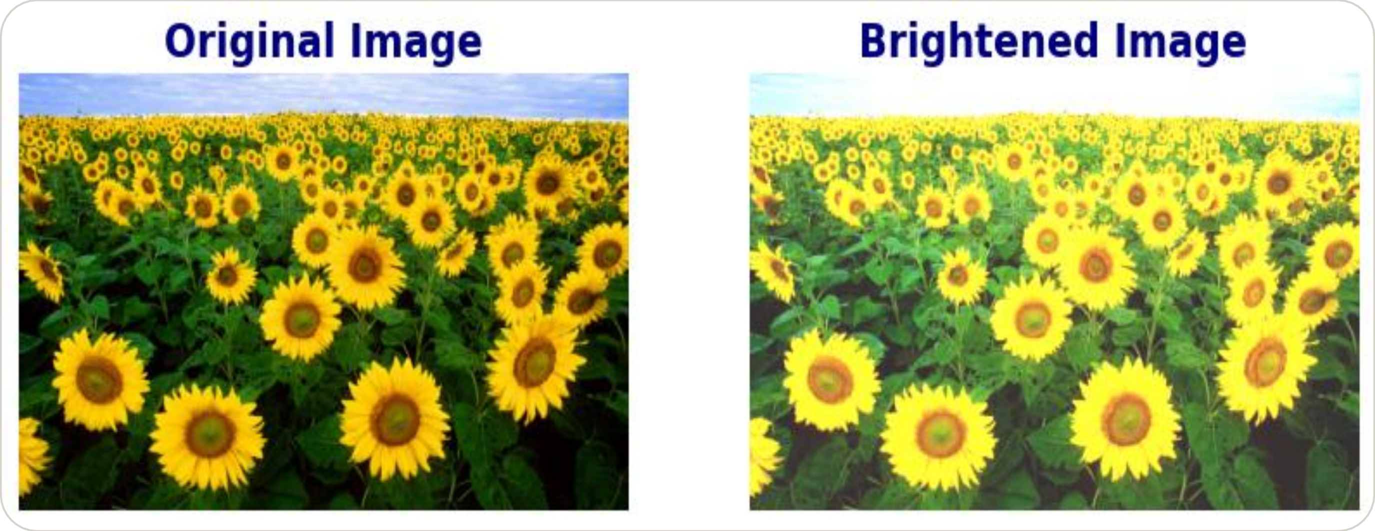
- **Key Points:**

1) **Subplots and Layouts:**
   - Arrange multiple images in grids for structured visualization.
2) **Side-by-Side Comparisons:**
   - Compare variations, such as before and after transformations.
3) **Before-After Visualizations**
   - Directly show the impact of a processing techniques.

Code Snippet (Subplots):

```
plt.subplot(1, 2, 1)
plt.imshow(original_image)
plt.title("Original Image")

plt.subplot(1, 2, 2)
plt.imshow(brightened_image)
plt.title(“Brightened Image")
plt.show()
```



Original Image

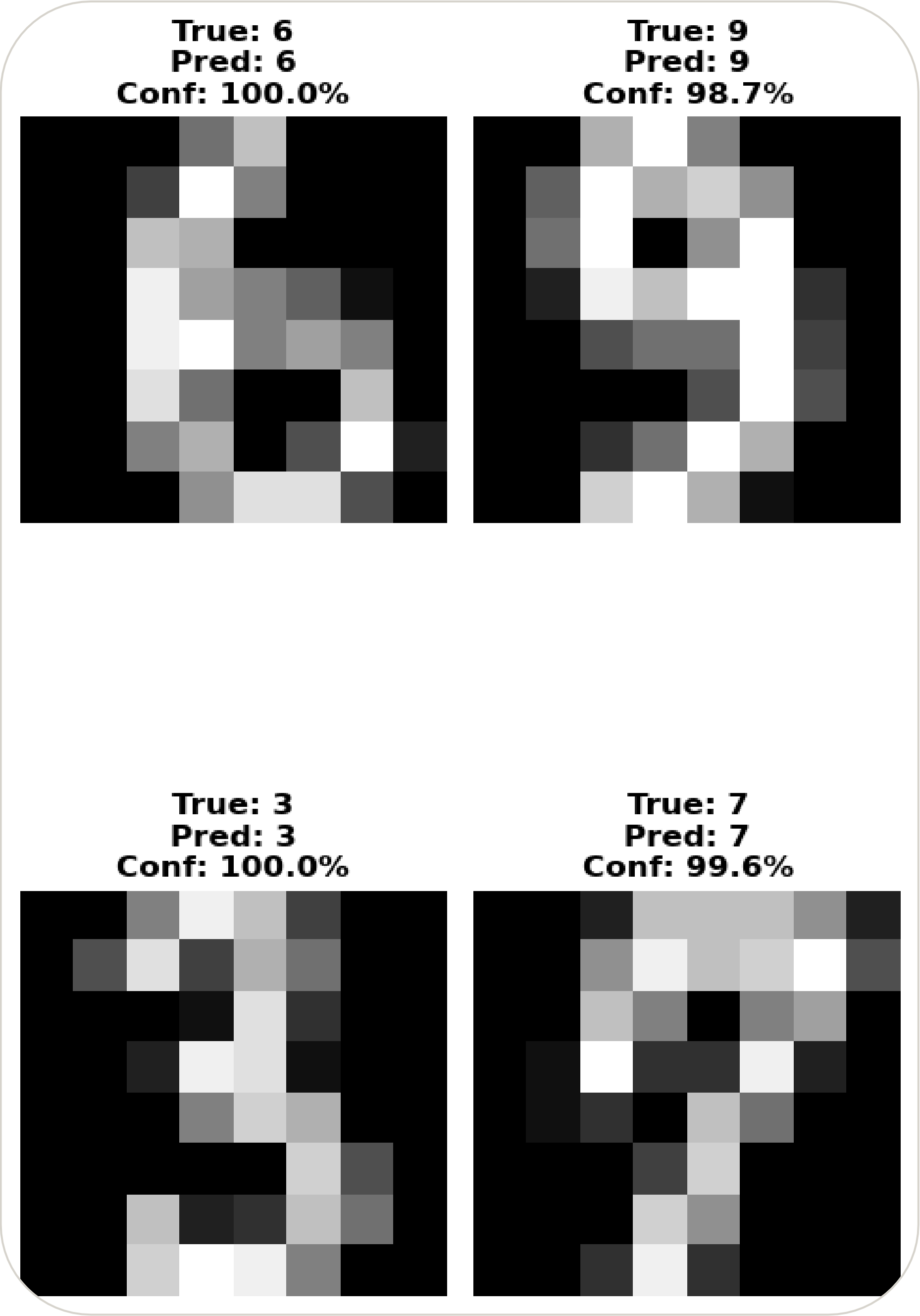
Brightened Image

Practical Applications

1) Visualizing Image Classification Results

- **Purpose:**
  - Understand model predictions for classification tasks.
  - Evaluate correct and incorrect predictions.
- **Techniques:**
  - Display images alongside predicted and true labels.
  - Include confidence scores for predictions.
- **Applications:**
  - Classifying handwritten digits (e.g., MNIST).
  - Identifying diseases in medical imaging.
- **Example Use Cases:**
  - Classifying handwritten digits (MNIST).
  - Identifying diseases in X-ray images.

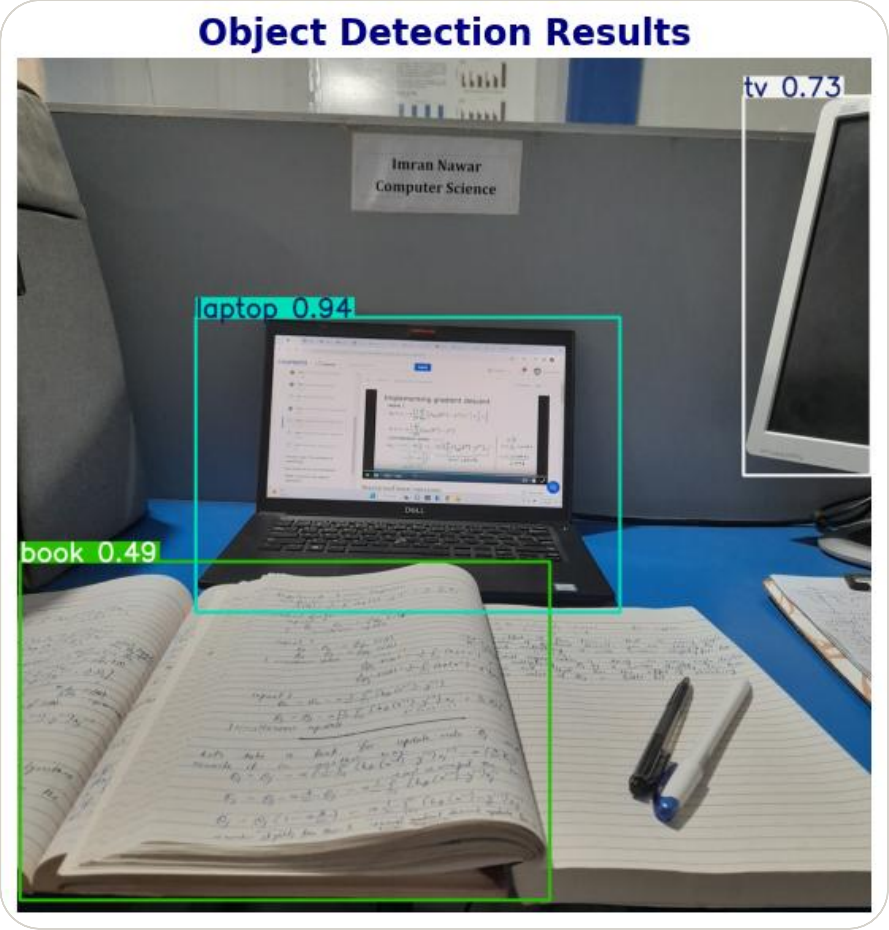
Image Classification Results

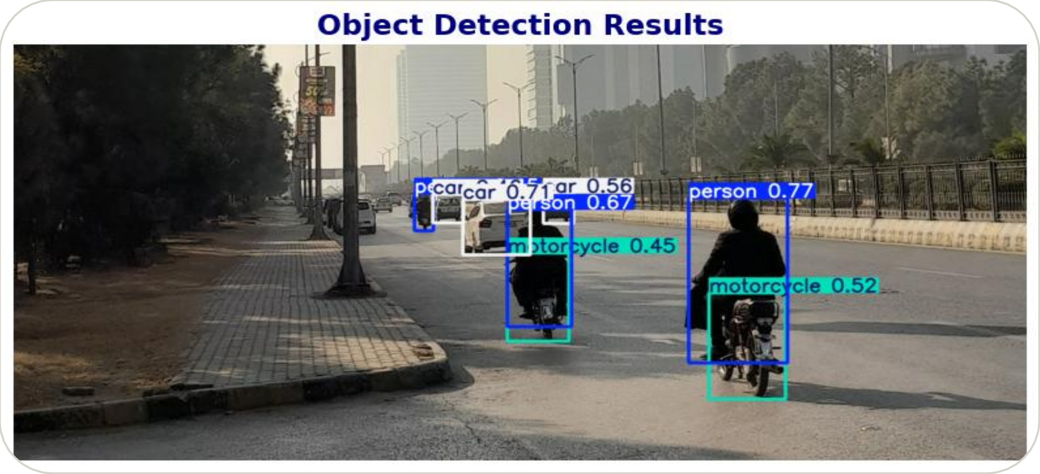


Practical Applications

2) Visualizing Object Detection Results

- **Purpose:**
  - Visualize detected objects in image.
  - Analyze object locations and categories.
- **Techniques:**
  - Overlay bounding boxes with class labels and confidence scores.
  - Use color-coded boxes for categories.
- **Applications:**
  - Detecting pedestrians and vehicles for autonomous driving.
  - Tracking animals in wildlife monitoring.
- **Example Use Cases:**
  - YOLO outputs with bounding boxes.

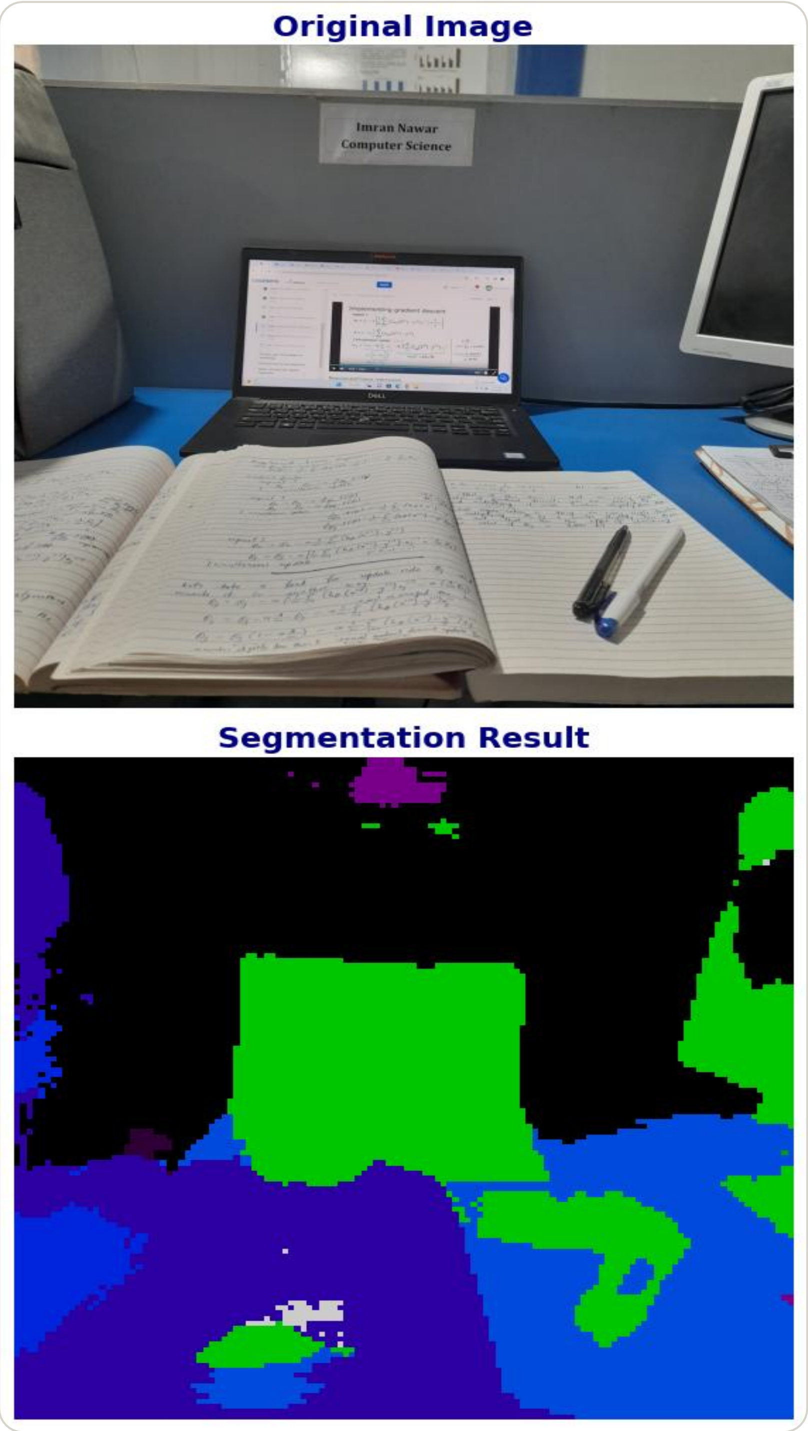




Practical Applications

3) Visualizing Segmentation Results

- **Purpose:**
  - Visualize pixel-level classifications.
  - Interpret overlays on original images.
- **Techniques:**
  - Overlay segmentation masks with distinct colors for classes.
- **Applications:**
  - Medical imaging for tumor segmentation.
  - Satellite imagery analysis.
- **Example Use Cases:**
  - Semantic segmentation with SegFormer models.



Summary and Next Steps

- **Key Takeaways:**
  - **Image Data**:
    - A numerical representation using pixels, channels, and bit depth.
    - Common formats include RGB, grayscale, and binary, vital for modern data science and AI.
  - **Image Processing Tools**
    - **OpenCV**: Advanced library for real-time applications and extensive functionality.
    - **Pillow**: Easy to use for basic manipulations like resizing, cropping, and saving.
    - Use OpenCV for complex tasks and real-time processing; Pillow for simple and intuitive workflows
  - **Basic Operations**:
    - Tasks like loading images, converting between color spaces, extracting metadata, and performing basic transformations (resize, rotate. flip).
  - Advanced Visualization Techniques:
    - Histograms, color distributions, and image enhancements improve clarity and understanding.
    - Multi-image visualizations (subplots, comparisons) provide deeper insights
  - **Next Steps:**
    - Apply image processing techniques on sample datasets like MNIST, CIFAR-10, or custom images.
    - Experiment with combining **OpenCV** and **Pillow** for diverse tasks in workflows.
- **Next Lecture:** 3D Data Visualization.

Thank You