

```
from IPython.display import Image, display  
  
# Display the image  
display(Image(filename='P1.png'))
```



Define Image

An image is a two-dimensional array of pixels (picture elements) that represent visual information. Each pixel contains color information or intensity values.

Common formats: JPEG, PNG.

Structure of Digital Images

- **Pixels:** The smallest unit of a digital image, arranged in a grid. Each pixel has a value representing its intensity and/or color.
- **Resolution:** The total number of pixels in an image, typically expressed as width × height (e.g., 1920 × 1080).
- All images consist of pixels in a grid. A 640 x 480 image has 640 rows and 480 columns. There are 640×480=307200 pixels.

Bit Depth:

The number of bits used to represent each pixel (e.g., 8-bit, 16-bit)

Color Channels:

- **Grayscale:** A single channel where each pixel intensity ranges from black (0) to white (255).

- Color: Multiple channels (e.g., RGB – Red, Green, Blue) combined to represent colors.).

What is Grayscale?

Grayscale refers to a type of image representation that uses only one channel, where each pixel has an intensity value ranging from black (0) to white (255). This means the image contains only shades of gray, without any color information.

Where is Grayscale Used?

- Image Processing & Analysis: Grayscale images simplify computations.
- Machine Learning & AI: Used in facial recognition, medical imaging (X-rays, MRI), and pattern detection.

What is RGB?

RGB (Red, Green, Blue) is a color model used in digital imaging where each pixel in an image is represented using a combination of three primary colors:

Red (R)
Green (G)
Blue (B)

Each of these colors has an intensity value ranging from 0 to 255, and by mixing them in different proportions, you can create millions of colors.

How RGB Works

If $R = 255$, $G = 0$, $B = 0$, the color is pure red.
If $R = 0$, $G = 255$, $B = 0$, the color is pure green.
If $R = 0$, $G = 0$, $B = 255$, the color is pure blue.
If $R = 255$, $G = 255$, $B = 255$, the color is white.
If $R = 0$, $G = 0$, $B = 0$, the color is black.

Why is RGB Used?

RGB is the standard color model used in:

Computer screens, TVs, and mobile devices
Digital cameras and image processing
Graphics design and video games

Difference Between Color and Grayscale Images

Image Processing

Image processing is a technique to perform operations on images to enhance them, extract useful information, or convert them into a desired format. It has applications in computer vision, medical imaging, surveillance, and more.

Image processing involves techniques to enhance, analyze, or transform images using mathematical operations. Examples include:

- Enhancement: Adjusting brightness, contrast, or sharpness.
- Analysis: Extracting features, edges, or objects.

What is OpenCV?

OpenCV (Open Source Computer Vision Library) is an open-source computer vision and machine learning software library. It provides a collection of tools and algorithms to process images and videos.

- `cv.imread( FileName, Flag )` to Read an image from a storage.
- `cv.imshow( window_Name, Image )` to display an image in a window.
- `cv.imwrite( File_Name, Image )` to write and save an image to the storage.
- A common color mode where images are represented using three channels (Red, Green, Blue).
- OpenCV reads images in BGR order by default, not RGB. To convert to RGB, use: `cv.cvtColor(image, cv.COLOR_BGR2RGB)`.

When reading images using `cv2.imread`, you can specify the following flags:

```
cv2.IMREAD_COLOR (Default): Reads a full-color image with three channels (BGR).  
cv2.IMREAD_GRAYSCALE: Reads the image as grayscale (1 channel).
```

How install main libraries

Install Anaconda, create a new environment, install Jupyter, open the command prompt, and enter the following commands

- `pip install opencv-python`
- `pip install tensorflow`
- `pip install keras`
- `pip install numpy`
- `pip install matplotlib`

Read and display image as RGB

```
import cv2
from matplotlib import pyplot as plt

# Load an image (replace 'image.jpg' with your image path)
image = cv2.imread('image.jpg')

# Convert BGR (OpenCV default) to RGB for proper display
image_rgb = cv2.cvtColor(image, cv2.COLOR_BGR2RGB)

# Display the image
plt.imshow(image_rgb)
plt.title("Loaded Image")
plt.axis('off')
plt.show()
```

Loaded Image



Convert Image to Grayscale

```
import cv2
import matplotlib.pyplot as plt

# Load the image
image = cv2.imread("image.jpg")

# Convert to grayscale
```

```
gray_image = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)

# Show the image
plt.imshow(gray_image, cmap="gray")
plt.axis("off")
plt.show()
```



How can read and write image?

```
import cv2

# Step 1: Read an image from storage
image = cv2.imread('image.jpg', cv2.IMREAD_COLOR)

# Step 2: Display the image in a window
cv2.imshow('Original Image', image)

# Step 3: Wait for a key press (0 means wait indefinitely)
key = cv2.waitKey(0)

# Step 4: Save the image to a new file if a specific key is pressed
(e.g., 's')
if key == ord('s'): # If 's' key is pressed
    cv2.imwrite('saved_image.jpg', image) # Save the image
    print("Image saved as 'saved_image.jpg'")
```

```
# Step 5: Close all OpenCV windows
cv2.destroyAllWindows()
```

Getting the width and height of an image

```
import cv2

# Load the image
image = cv2.imread("image.jpg")

# Get dimensions
height, width, channels = image.shape

print(f"Width: {width}, Height: {height}")

Width: 612, Height: 612
```

Image data augmentation

- Image data augmentation is used to expand the training dataset in order to improve the performance and ability of the model to generalize.
- Image data augmentation is supported in the Keras deep learning library via the ImageDataGenerator class. What are different types of data augmentation methods?
- Horizontal and Vertical Shift Augmentation: This augmentation shifts the image along the x-axis (horizontal) and y-axis (vertical).
- Horizontal and Vertical Flip Augmentation: Flips the image randomly along the horizontal or vertical axis.
- Random Rotation Augmentation: Rotates the image by a random angle within the specified range.
- Random Brightness Augmentation: Adjusts the brightness of the image randomly.
- Random Zoom Augmentation: Zooms into or out of the image randomly

```
from tensorflow.keras.preprocessing.image import ImageDataGenerator,
load_img, img_to_array
import numpy as np
import matplotlib.pyplot as plt
import os

def display_augmented_images(img_path, datagen, num_augmented=3):
    """
    Display original and augmented versions of an image using a given
    ImageDataGenerator.

    Parameters:
    - img_path (str): Path to the image.
    - datagen (ImageDataGenerator): Keras image data generator for
    augmentation.
    - num_augmented (int): Number of augmented images to display.
    """
```

```

if not os.path.exists(img_path):
    raise FileNotFoundError(f"Image not found: {img_path}")

# Load and prepare the image
img = load_img(img_path)
img_array = img_to_array(img)
img_array = np.expand_dims(img_array, axis=0)

# Generate augmented images
augmented_images = datagen.flow(img_array, batch_size=1)

# Display original and augmented images
plt.figure(figsize=(4 * (num_augmented + 1), 5))

# Show original image
plt.subplot(1, num_augmented + 1, 1)
plt.title('Original')
plt.imshow(img)
plt.axis('off')

# Show augmented images
for i in range(num_augmented):
    batch = next(augmented_images)
    aug_img = batch[0].astype('uint8')
    plt.subplot(1, num_augmented + 1, i + 2)
    plt.title(f'Augmented {i+1}')
    plt.imshow(aug_img)
    plt.axis('off')##

```

Horizontal and Vertical Flip Augmentation

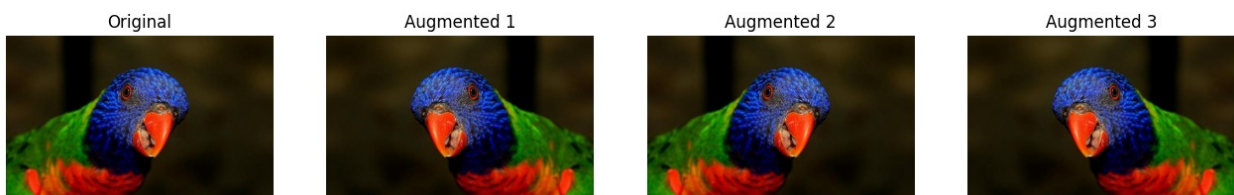
An image flip means reversing the rows or columns of pixels in the case of a vertical or horizontal flip respectively

horizontal_flip

```

datagen = ImageDataGenerator(horizontal_flip=True)
display_augmented_images('bird.jpg', datagen)

```



vertical_flip

```

datagen = ImageDataGenerator(vertical_flip=True, )
display_augmented_images('bird.jpg', datagen, num_augmented=4)

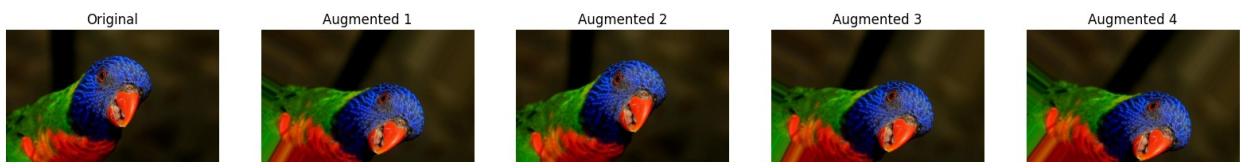
```



rotation augmentation randomly

A rotation augmentation randomly rotates the image clockwise by a given number of degrees from 0 to 360

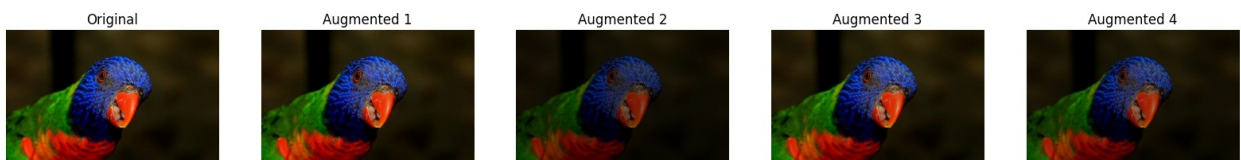
```
datagen = ImageDataGenerator(rotation_range=40)
display_augmented_images('bird.jpg', datagen, num_augmented=4)
```



andom Brightness Augmentation

The brightness of the image can be augmented by either randomly darkening images, brightening images, or both. The intent is to allow a model to generalize across images trained on different lighting levels

```
datagen = ImageDataGenerator(brightness_range=[0.5, 1.5])
display_augmented_images('bird.jpg', datagen, num_augmented=4)
```



Random Zoom Augmentation

A zoom augmentation randomly zooms the image in and either adds new pixel values around the image or interpolates pixel values respectively. Image zooming can be configured by the `zoom_range` argument to the `ImageDataGenerator` constructor. You can specify the percentage of the zoom as a single float or a range as an array or tuple. If a float is specified, then the range for the zoom will be $[1-\text{value}, 1+\text{value}]$. For example, if you specify 0.3, then the range will be $[0.7, 1.3]$, or between 70% (zoom in) and 130% (zoom out)

```
datagen = ImageDataGenerator(zoom_range=0.6,)
display_augmented_images('bird.jpg', datagen)
```



How Download data from Kaggle

- Create a Kaggle Account (if you don't have one): Go to <https://www.kaggle.com> and sign up/log in.
- Go this link: <https://www.kaggle.com/datasets/adityamahimkar/iqothnccd-lung-cancer-dataset/versions/1> and download the dataset
- This dataset is about lung cancer and consists of three classes: Benign, Malignant, and Normal.

Splitting dataset

We split the dataset into different folders: training set, validation set, and testing set. Each folder contains subfolders named after the respective classes.

```
import os, shutil
from sklearn.model_selection import train_test_split

original_dataset_dir = 'The IQ-0THNCCD lung cancer dataset'
base_dir = 'dataset'
train_dir = os.path.join(base_dir, 'train')
validation_dir = os.path.join(base_dir, 'validation')
test_dir = os.path.join(base_dir, 'test')

for dir in [train_dir, validation_dir, test_dir]:
    os.makedirs(dir, exist_ok=True)

train_ratio, validation_ratio, test_ratio = 0.7, 0.1, 0.2

def split_and_copy_files(class_dir, dest_base_dir):
    all_images = [os.path.join(class_dir, f) for f in
os.listdir(class_dir) if os.path.isfile(os.path.join(class_dir, f))]
    train_images, temp_images = train_test_split(all_images,
test_size=(1 - train_ratio), random_state=42)
    validation_images, test_images = train_test_split(temp_images,
test_size=test_ratio/(validation_ratio + test_ratio), random_state=42)

    for images, folder_name in zip([train_images, validation_images,
test_images], ['train', 'validation', 'test']):
        dest_dir = os.path.join(dest_base_dir, folder_name,
os.path.basename(class_dir))
        os.makedirs(dest_dir, exist_ok=True)
        for file in images:
            shutil.copy(file, dest_dir)
```

```
# Apply to all class subfolders
for class_dir in os.listdir(original_dataset_dir):
    full_class_dir = os.path.join(original_dataset_dir, class_dir)
    if os.path.isdir(full_class_dir):
        split_and_copy_files(full_class_dir, base_dir)
```