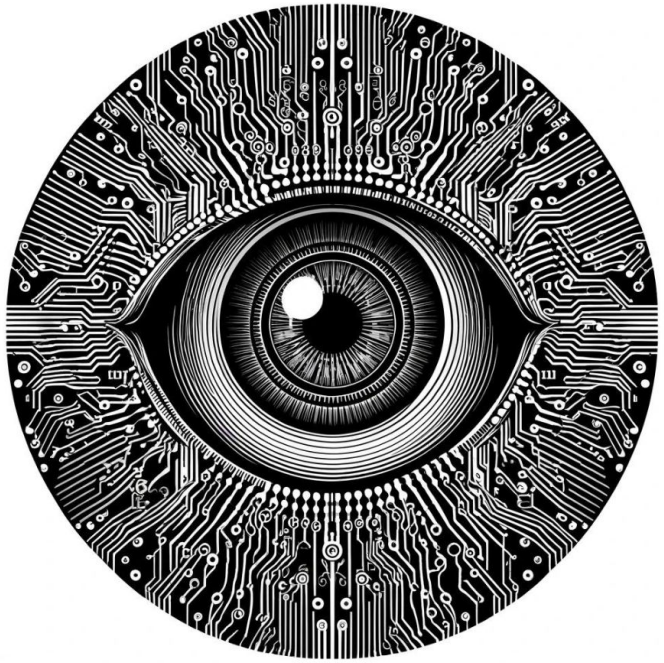
KI Service Zentrum by Hasso-Plattner-Institut HPI

Digital Images in PIL and NumPy



Antonio Rueda-Toicen

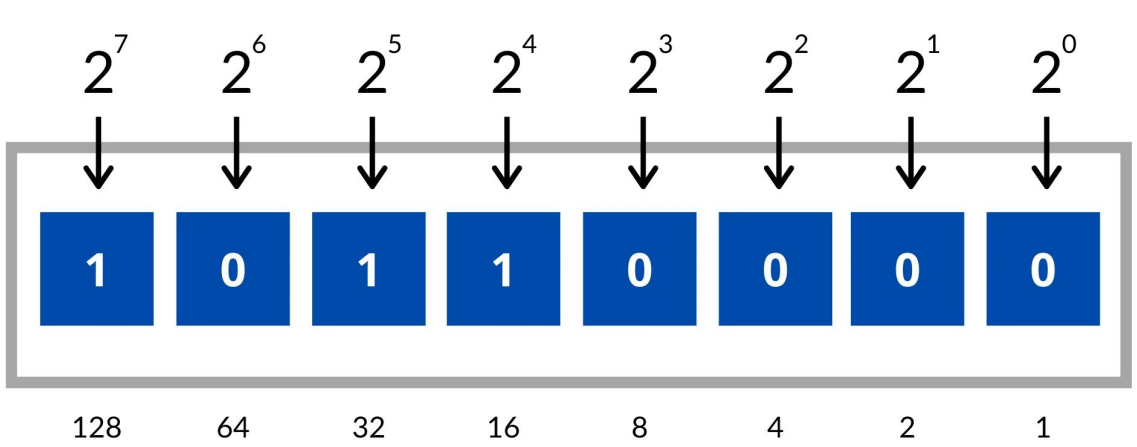
SPONSORED BY THE

Federal Ministry of Education and Research

Learning goals

- Understand image tensors as multidimensional arrays
- Gain familiarity with PIL images and NumPy arrays

The unsigned integer (`uint8`) datatype



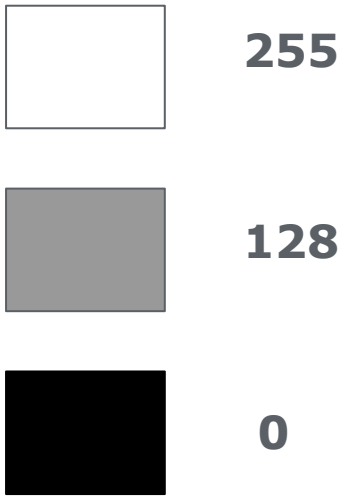
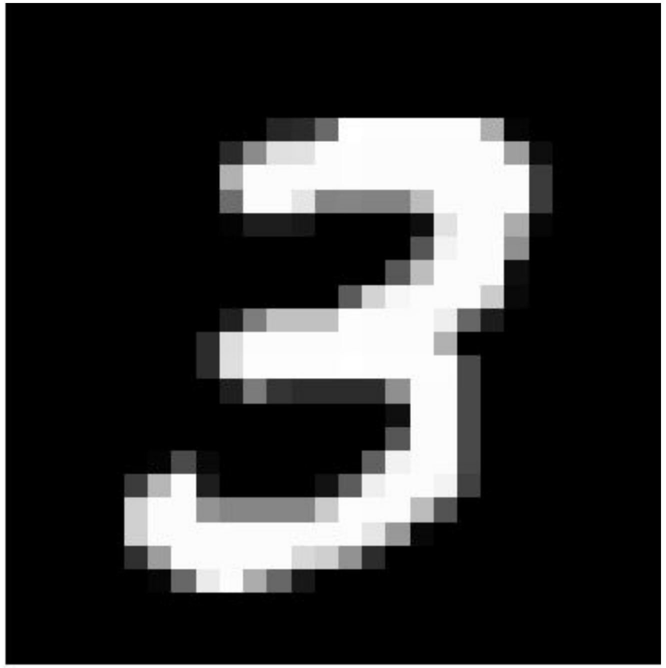
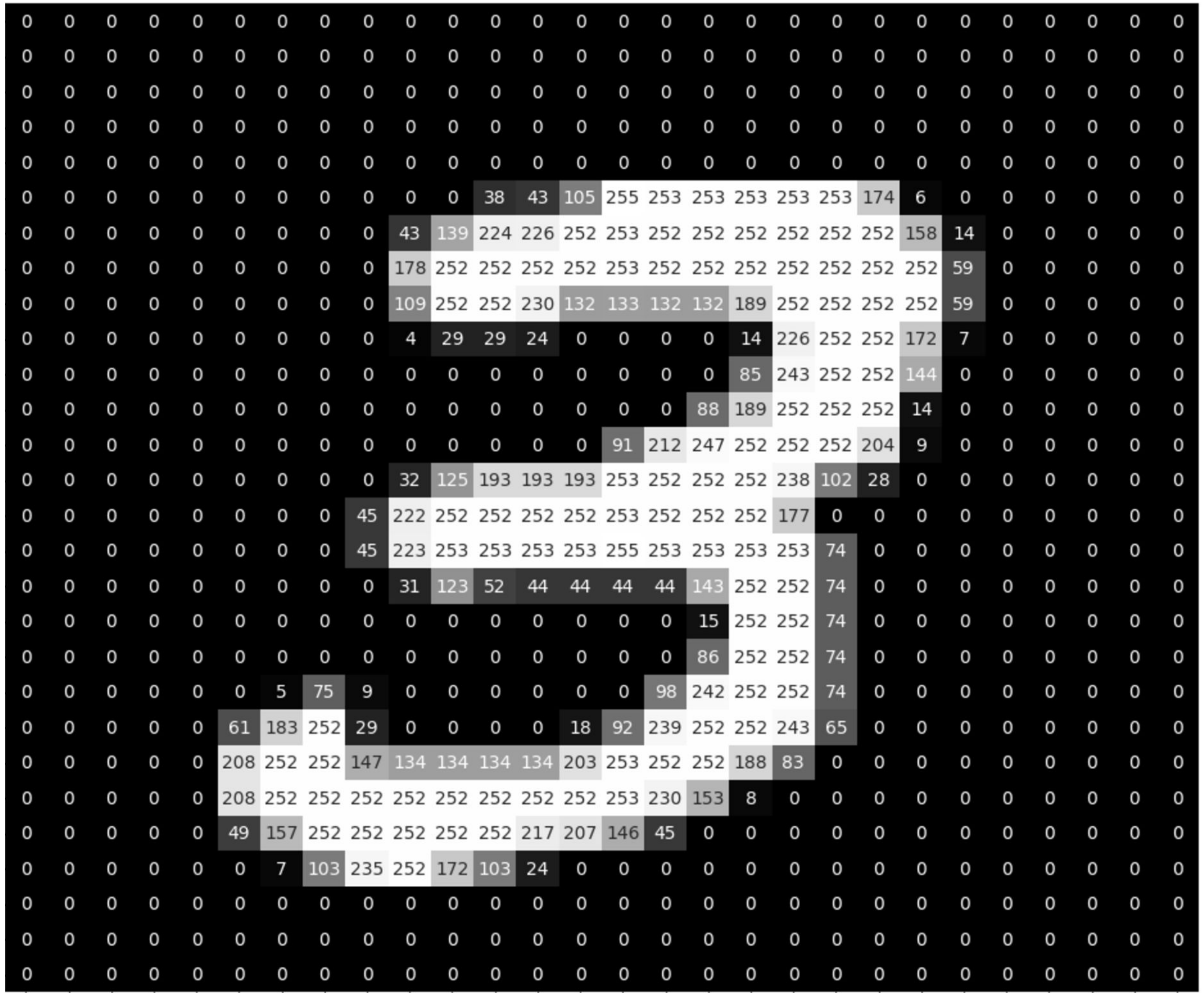


Image from [Why Are There 8 Bits in a Byte?]

Digital image representation for grayscale images



what we see



0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	38	43	105	255	253	253	253	253	253	174	6	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	43	139	224	226	252	253	252	252	252	252	252	252	158	14	0	0	0	0	0
0	0	0	0	0	0	0	0	0	178	252	252	252	252	253	252	252	252	252	252	252	252	59	0	0	0	0	0
0	0	0	0	0	0	0	0	0	109	252	252	230	132	133	132	132	189	252	252	252	252	59	0	0	0	0	0
0	0	0	0	0	0	0	0	0	4	29	29	24	0	0	0	0	14	226	252	252	172	7	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	85	243	252	252	144	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	88	189	252	252	252	14	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	91	212	247	252	252	252	204	9	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	32	125	193	193	193	253	252	252	252	238	102	28	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	45	222	252	252	252	252	253	252	252	252	177	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	45	223	253	253	253	253	255	253	253	253	253	74	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	31	123	52	44	44	44	44	143	252	252	74	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	15	252	252	74	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	86	252	252	74	0	0	0	0	0	0	0	0
0	0	0	0	0	0	5	75	9	0	0	0	0	0	0	98	242	252	252	74	0	0	0	0	0	0	0	0
0	0	0	0	0	61	183	252	29	0	0	0	0	18	92	239	252	252	243	65	0	0	0	0	0	0	0	0
0	0	0	0	0	208	252	252	147	134	134	134	134	203	253	252	252	188	83	0	0	0	0	0	0	0	0	0
0	0	0	0	0	208	252	252	252	252	252	252	252	252	253	230	153	8	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	49	157	252	252	252	252	252	217	207	146	45	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	7	103	235	252	172	103	24	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

what the computer "sees"

RGB images as “tensors” (stacks of matrices)

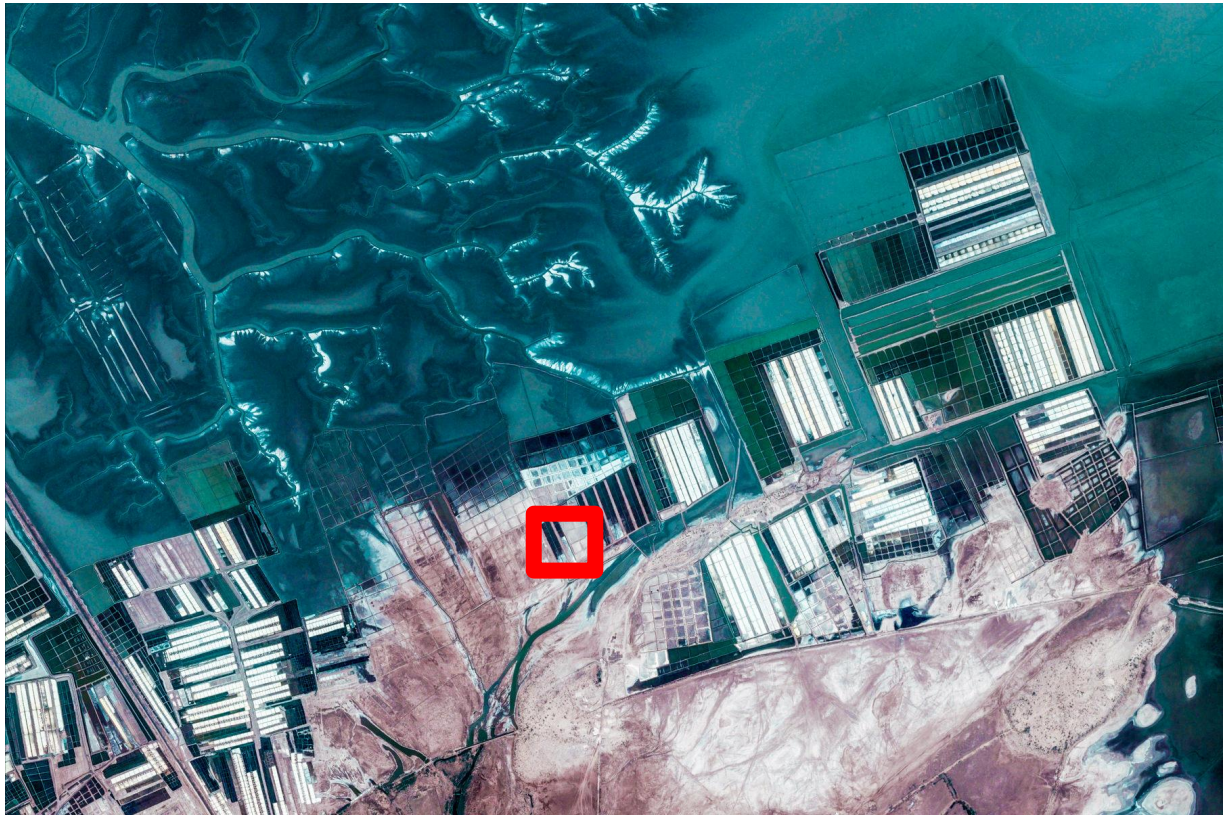


Image from Google Earth

What a human sees

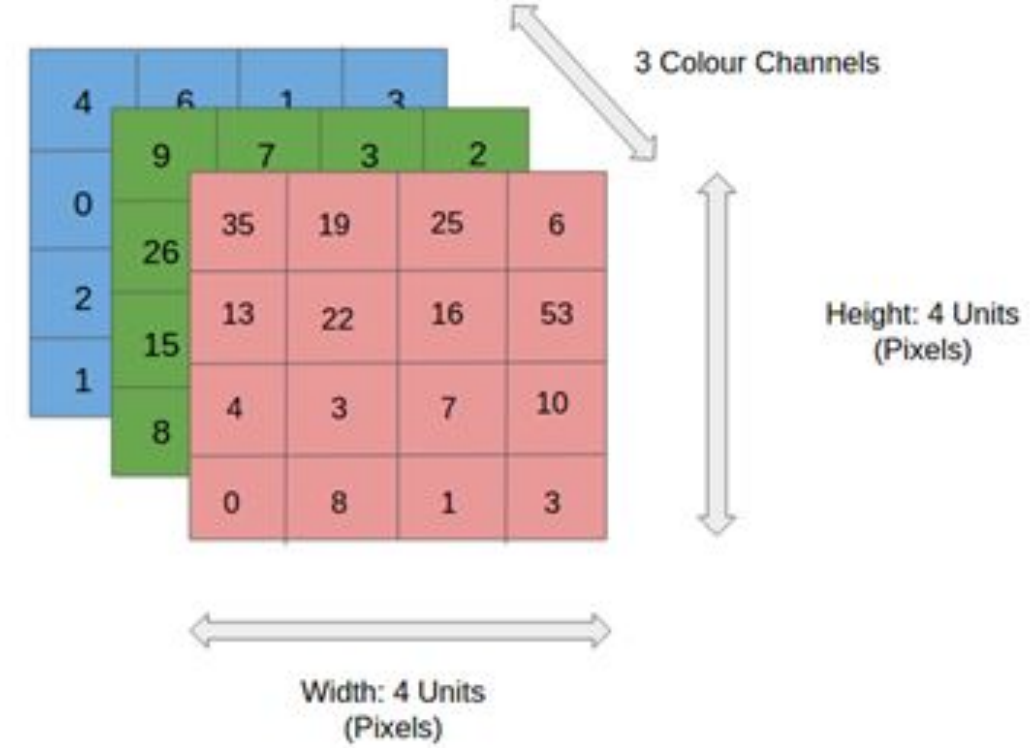
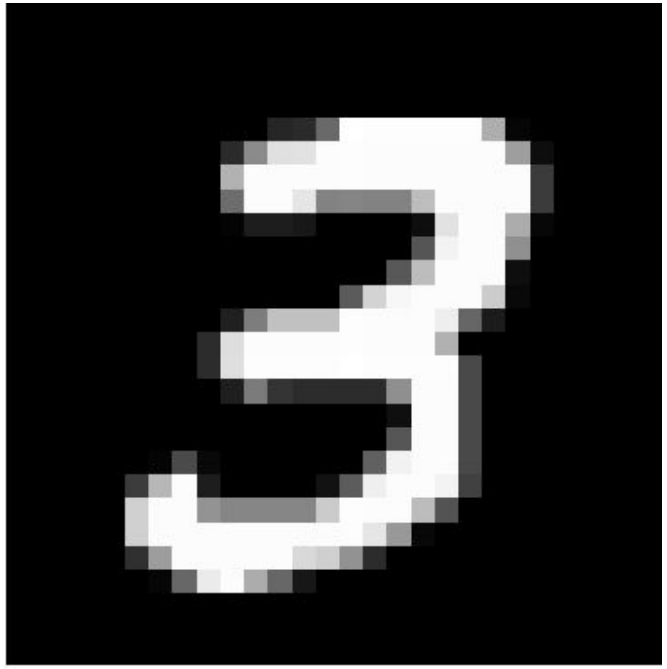


Image source

What the computer ‘sees’

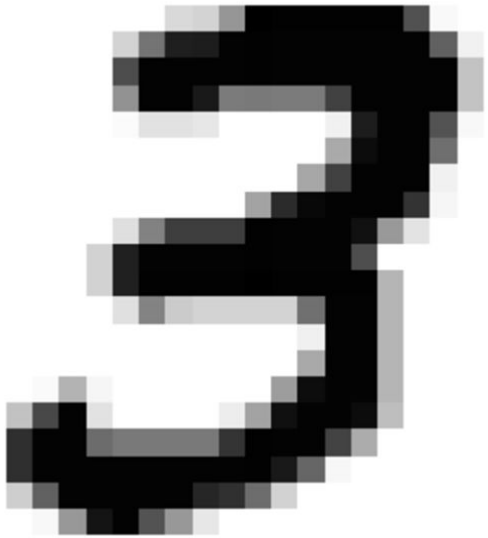
The Python Image Library (PIL)



real size is W = H = 28

```python
from PIL import Image

# Open image from file
pil_image = Image.open("path_to_image.jpg")

# Check the mode of the image, 'L' is for grayscale
# pil_image.get
print(pil_image.mode)

# Will print (28, 28), width first, height second
print(pil_image.size)

# Convert a NumPy ndarray to PIL
pil_image = Image.fromarray(np_array)

# Save image to storage as a PNG file
pil_image.save("path_to_image.png")

# Convert to numpy and upscale the image for visualization
import matplotlib.pyplot as plt
plt.imshow(pil_image)
```

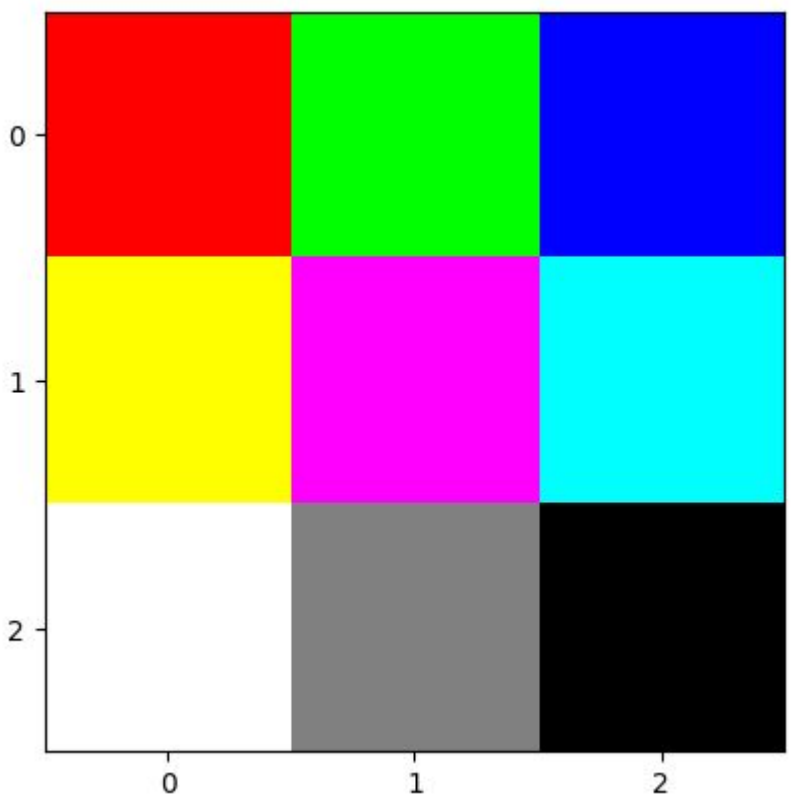
NumPy's N-dimensional array (`ndarray`)



```python
# Convert a PIL image to a Numpy array
np_array = np.array(pil_image)

# Check the shape of the array, prints (28, 28)
print(np_array.shape)

# Prints the data type, prints np.uint8
print(np_array.dtype)

# 255 minus current pixel values with NumPy's broadcasting
np_array_neg = 255 - np_array

# Show upscaled version on matplotlib
plt.imshow(np_array_neg)
```

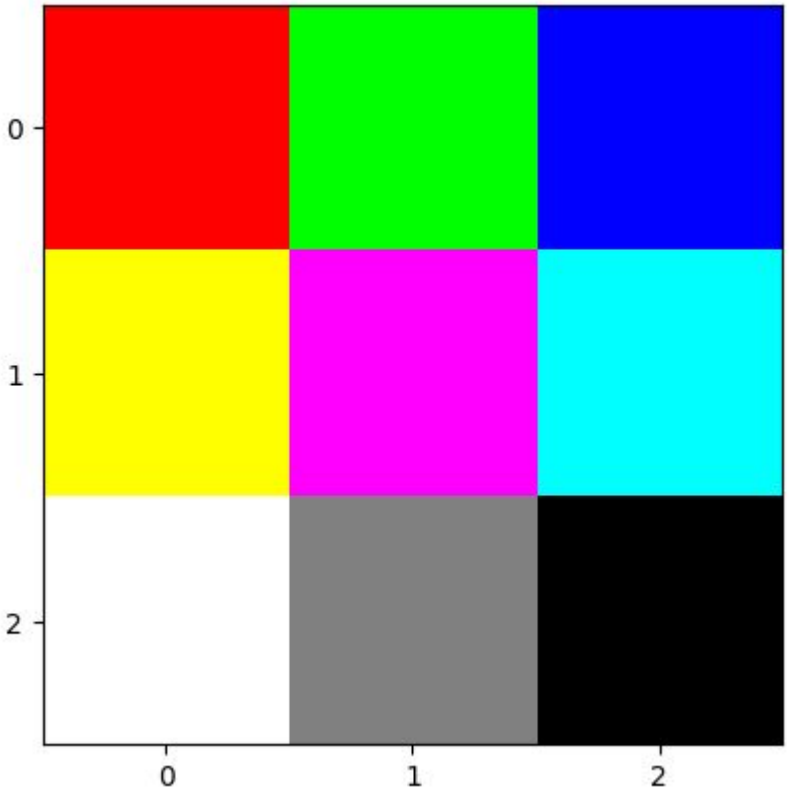
Red Green Blue (RGB) Images in NumPy

```
import numpy as np

# Create a3x3 RGB image array
rgb_image = np.array([
   # First row of pixels
   [[255, 0, 0],     # Pure red pixel
    [0, 255, 0],     # Pure green pixel
    [0, 0, 255]],    # Pure blue pixel

   # Second row of pixels
   [[255, 255, 0],   # Yellow pixel (red + green)
    [255, 0, 255],   # Magenta pixel (red + blue)
    [0, 255, 255]], # Cyan pixel (green + blue)

   # Third row of pixels
   [[255, 255, 255],# White pixel (all colors maximum)
    [128, 128, 128],# Gray pixel (all colors at half intensity)
    [0, 0, 0]]       # Black pixel (all colors minimum)
])
```

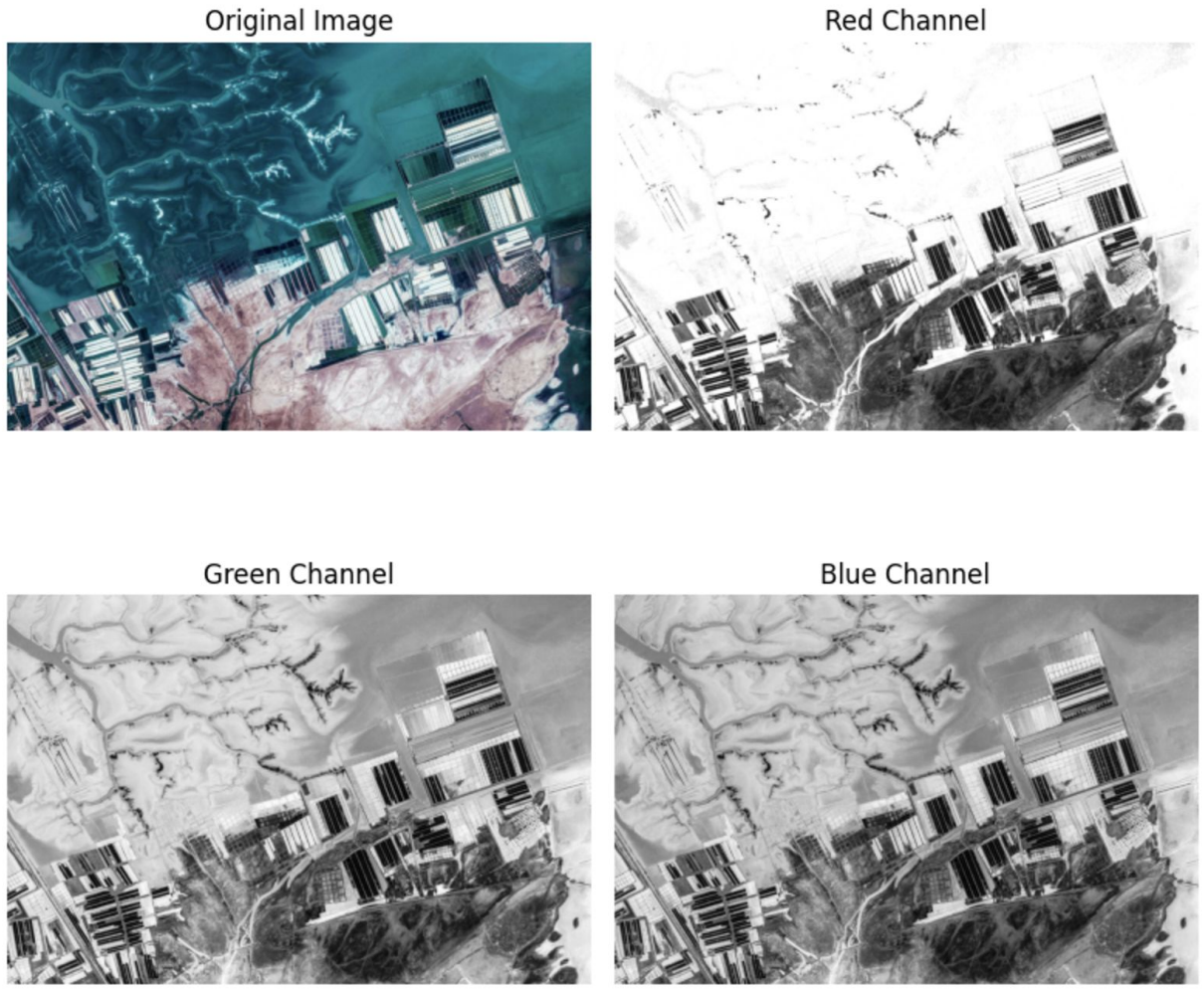


Indexing values in NumPy

```
# We select a row and a column
# Indexing starts at 0, as in standard Python
# We index first with height, then width
pixel = rgb_image[2, 0]

# Will show [255, 255, 255]
# Which index should we use to print magenta's values?
print(f"RGB values: {pixel}")
```

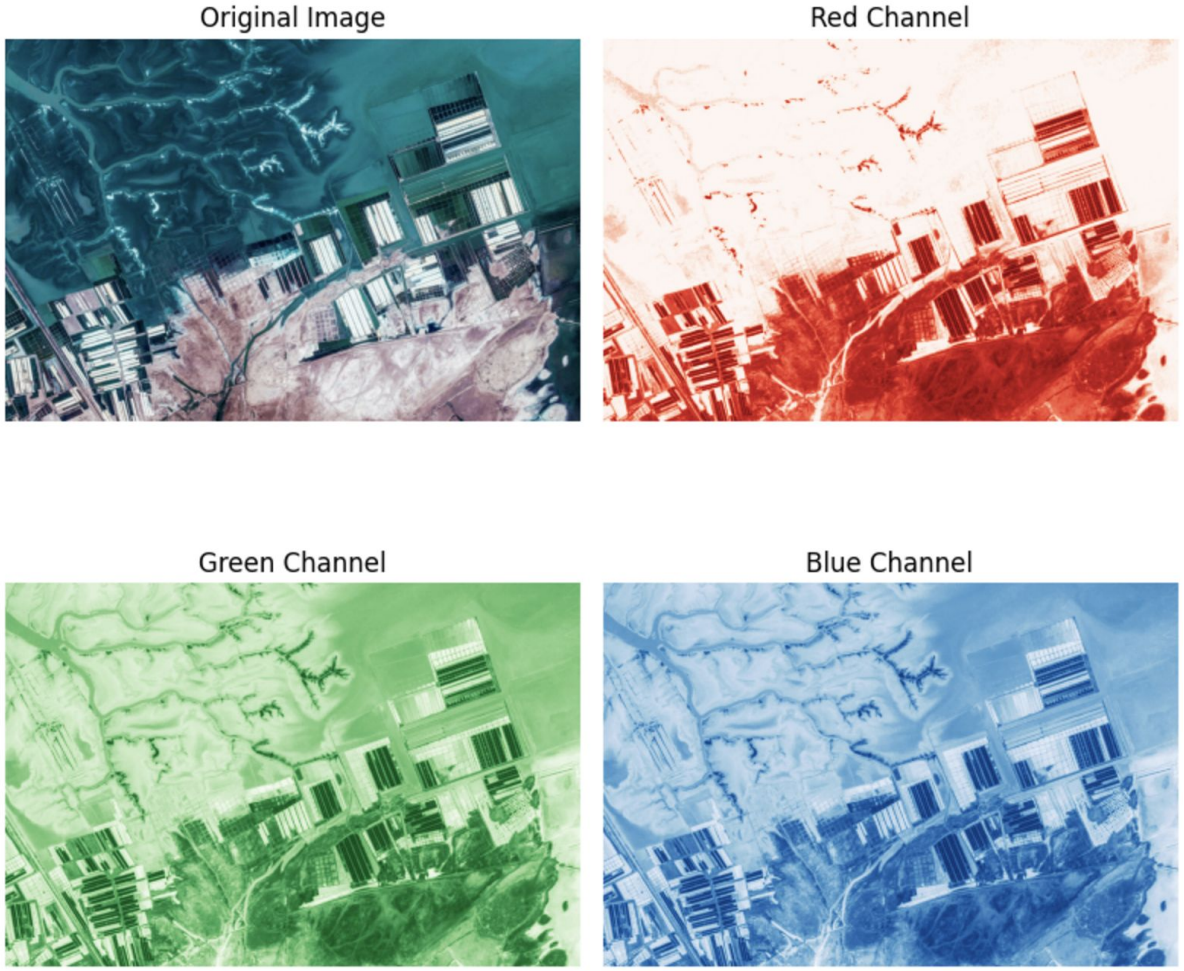


Intensities for RGB images in NumPy




```python
import maplotlib.pyplot as plt

# Intensity image from the R channel
plt.imshow(rgb_np_array[:, :, 0])

# Intensity image from the G channel
plt.imshow(rgb_np_array[:, :, 1])

# Intensity image from the B channel
plt.imshow(rgb_np_array[:, :, 2])
```

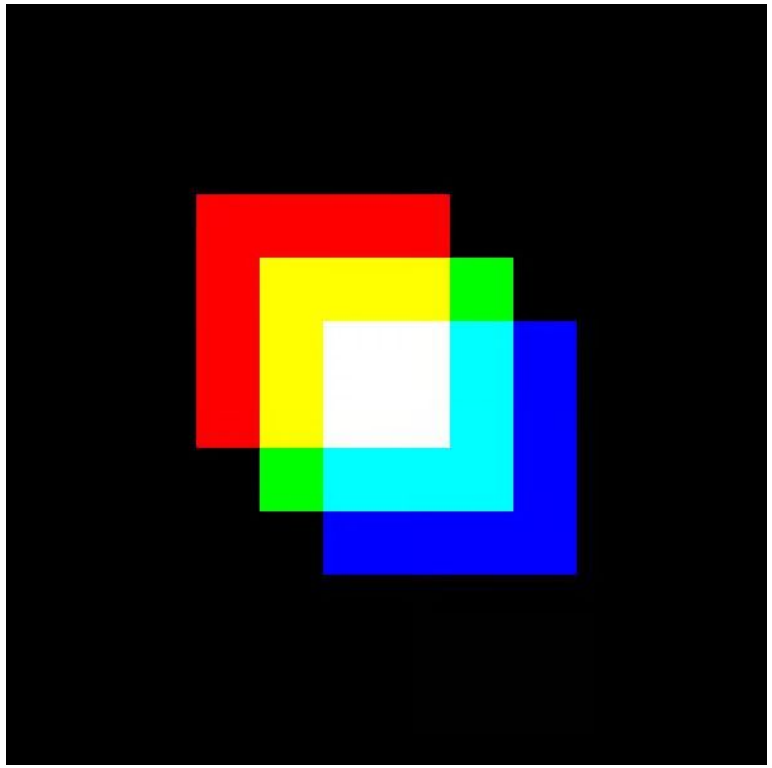
False colors for intensity images




```
import skimage.io as io

# Converts from JPEG to NumPy array
rgb_np_array =
io.imread("gujarat_image_url.jpeg")

# Images in NumPy are saved as
# Height (H), Width(W), Channels (C)
# Prints (1200, 1800, 3)
print(rbg_np_array.shape)

# Prints uint8
print(np_array.dtype)

# Intensity image with Red colormap
plt.imshow(rgb_np_array[:, :, 0], cmap='Reds')
plt.imshow(rgb_np_array[:, :, 1], cmap='Greens')
plt.imshow(rgb_np_array[:, :, 2], cmap='Blues')
```

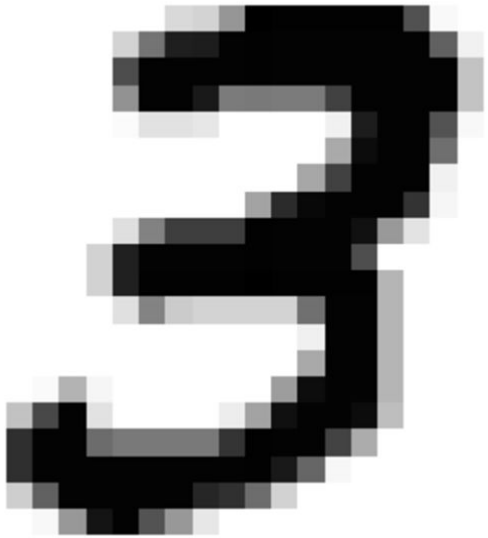
Image arithmetic in NumPy

```
import numpy as np
from PIL import Image

# Create arrays of of zeros
red = np.zeros((600, 600))
green = np.zeros((600, 600))
blue = np.zeros((600, 600))

# Set sections to maximum intensity
red[150:350, 150:350] = 255
green[200:400, 200:400] = 255
blue[250:450, 250:450] = 255

red_img = Image.fromarray(red).convert("L")
green_img = Image.fromarray(green).convert("L")
blue_img = Image.fromarray((blue)).convert("L")

# Merge channels
square_img = Image.merge("RGB", (red_img, green_img, blue_img))
square_img.show()
```



Code and image from source

The dimensions of PIL and NumPy arrays



NumPy image tensors follow the format: `[H, W, C]` where:

- `H` = height in pixels
- `W` = width in pixels
- `C` = channels (e.g., 1 for grayscale, 3 for RGB)

PIL image tensors follow the format `[W, H]` with mode

- mode is `"L"` for grayscale images
- mode is `"RGB"` for color images

KI Service Zentrum by Hasso-Plattner-Institut HPI

Summary

Digital images are represented as numeric pixel values

- uint8 is memory-efficient and commonly used to store images

We use different Python libraries for image processing

- PIL for image loading and saving to JPEG or PNG
- NumPy for numerical operations and visualization with matplotlib

PIL and NumPy use different formats to represent images

- Height, width, and number of channels is the standard for NumPy
- Width, height and mode is the standard for PIL

SPONSORED BY THE
Federal Ministry of Education and Research

References

NumPy's N-dimensional array

- https://numpy.org/doc/2.1/reference/arrays.ndarray.html

Intro to PIL and NumPy for image manipulation

- https://realpython.com/image-processing-with-the-python-pillow-library/

SPONSORED BY THE
Federal Ministry of Education and Research