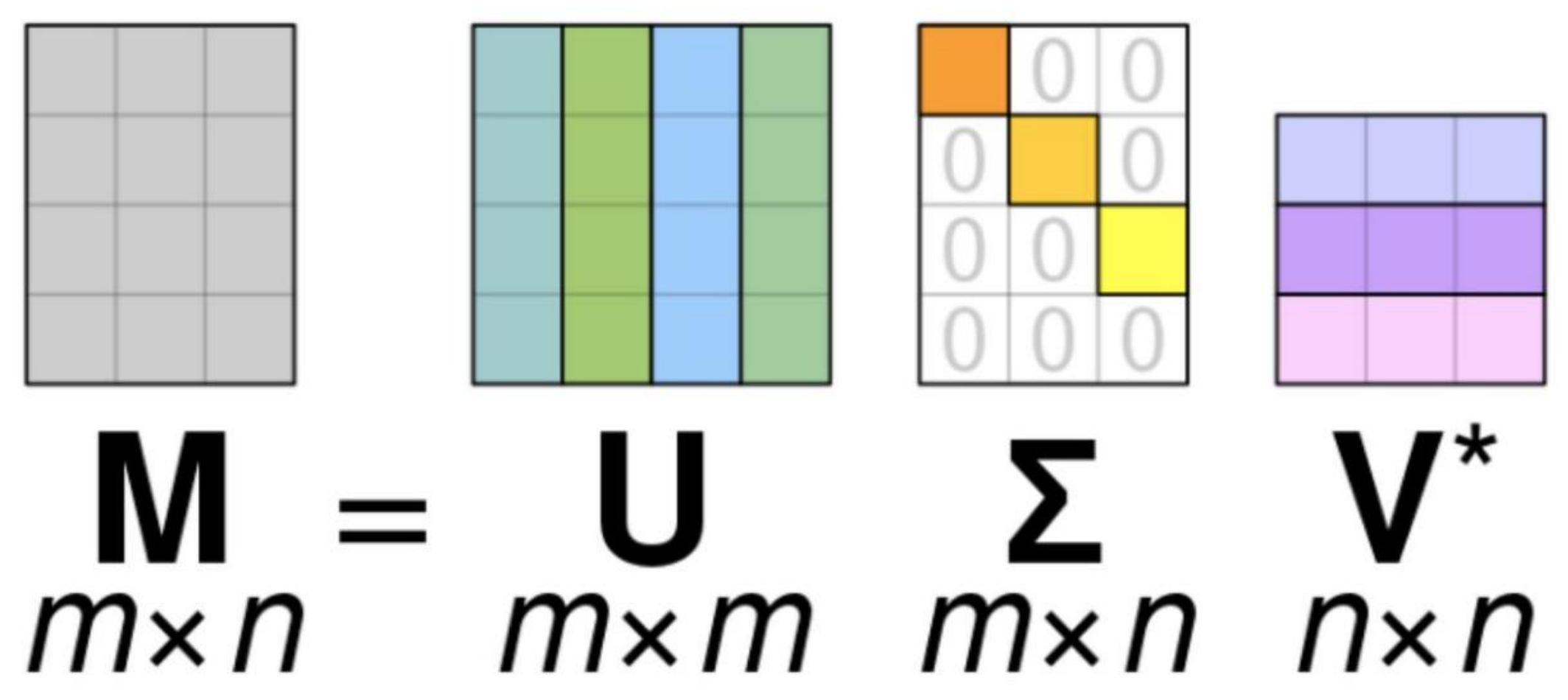
Mathematical Modeling for Digital Watermarking

Sophia Susanto, Aastha Malhotra, Isaac Termure, Maryna Sivachenko, Han Ji
Mentor: Narayani Choudhury
Mathematics and Computer Science

Abstract

- Automated digital watermarking is an excellent technique for prevention of online digital data from theft and piracy. With current advances in internet, network and social media technologies, protecting online data theft and piracy to preserve brand ownership remains a top priority.
- A watermark is a pattern inserted into a digital image, audio or video file which identifies a file's copyright information. Common types of signals to watermark are text, digital images, audio music clips and videos.
- Digital images are stored as arrays/matrices on computers which allows matrix-algebra based methods for image processing.
- We have applied mathematical modeling techniques using singular value decomposition (SVD) for digital watermarking of visual data. We wrote Python-based codes to digitize images and embedded copyright information into visual images using SVD based methods.
- We find that the embedded watermark is tamper resistant and the watermark could be retrieved from manipulated greyscale images subject to rotation and compression distortions.
- Our preliminary studies using greyscale images suggest that SVD based digital watermarking methods are robust and can be used to verify and authenticate data ownership. Digital watermarking can be used to prevent copyright infringement and data theft online.
- This project integrates advanced application of linear-algebra based mathematical methods with Python based programming to provide solutions to real-world problems of current interest.

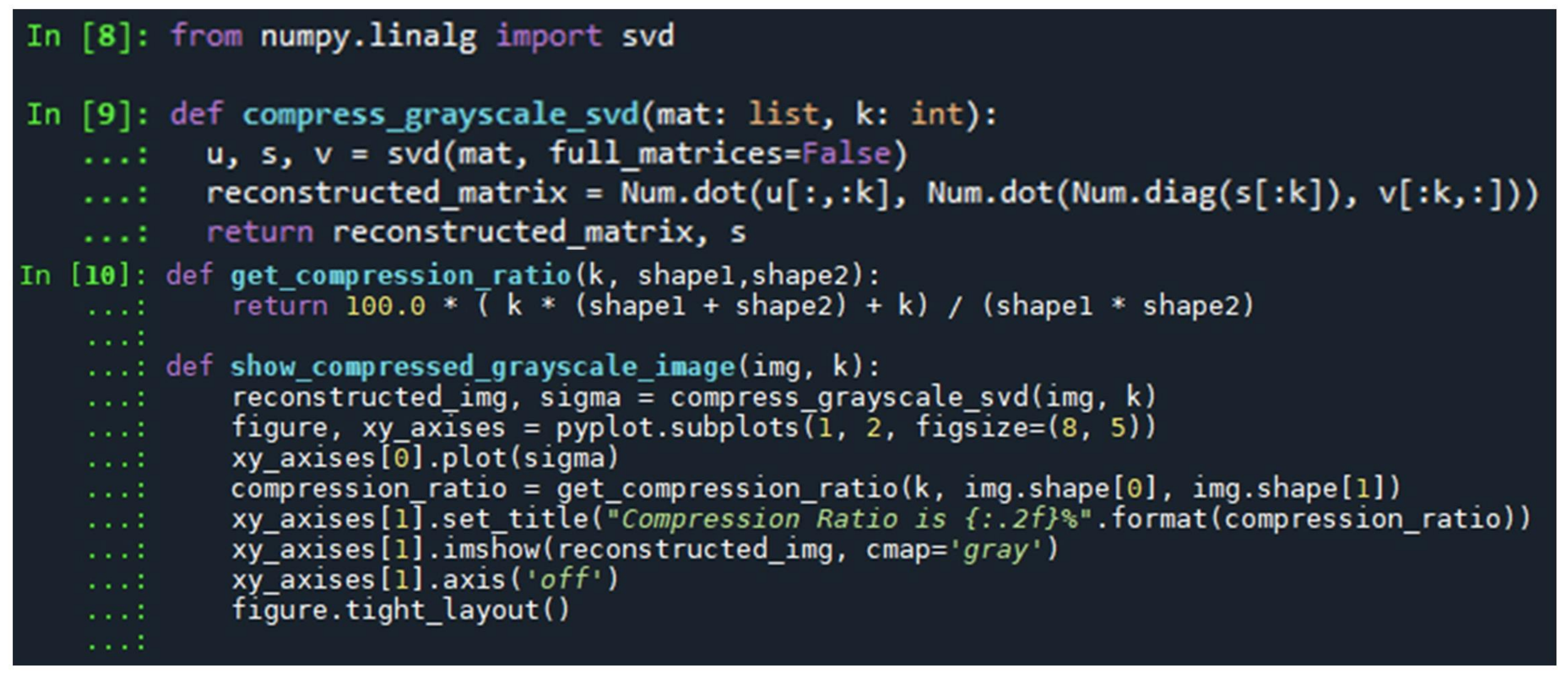
Singular Value Decomposition (SVD)



(i) MMT has dimensions mxm – Calculate eigenvalues and eigenvectors of MMT The eigenvectors of MMT form columns of U

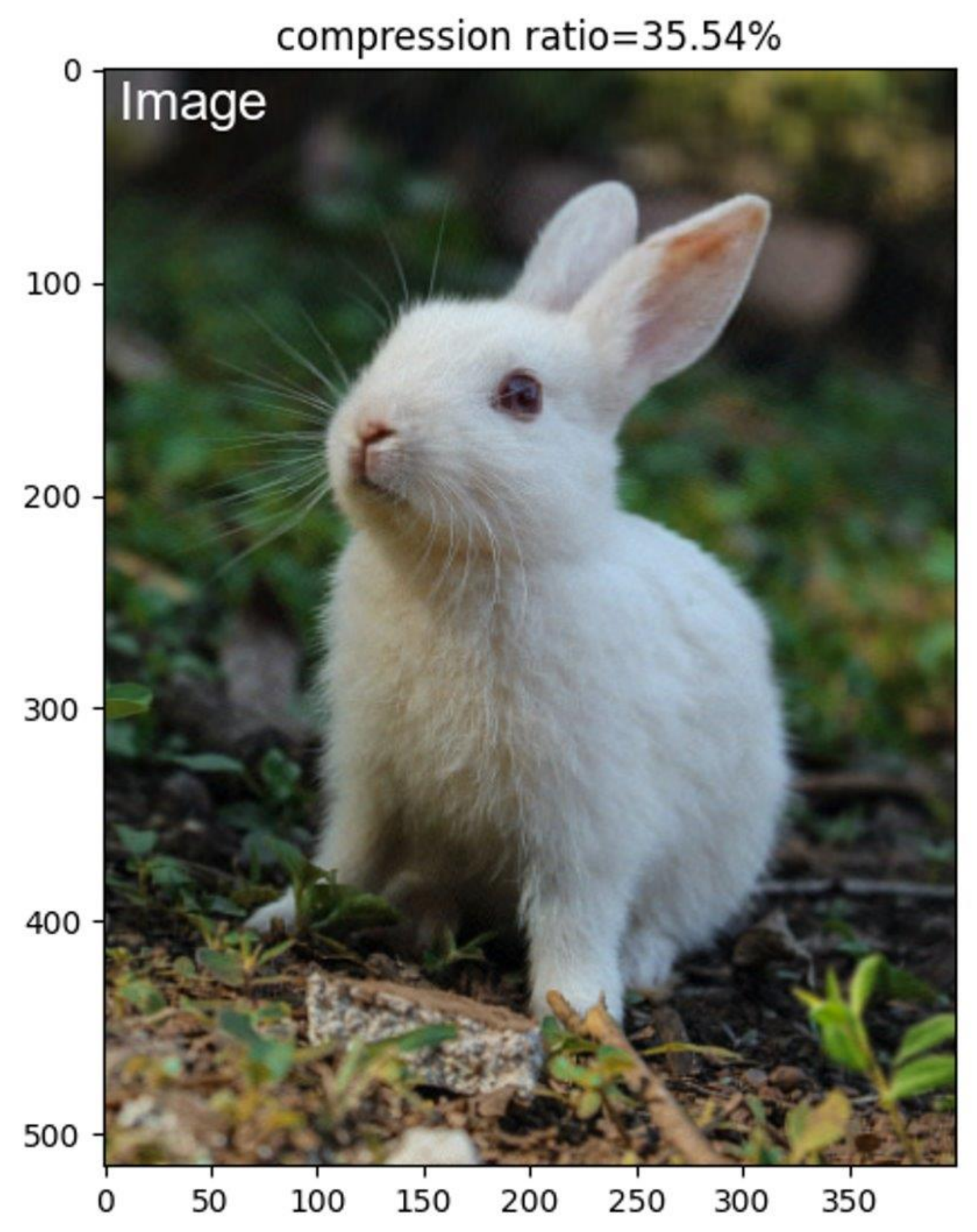
(ii) (ii) S is a diagonal mxn matrix and the diagonal elements of S are square root of the eigenvalues of MMT (with additional zeros in the diagonal if required for dimensional matching)

(iii) (iii) MTM has dimensions nxn - Calculate eigenvalues and eigenvectors of MTM . The eigenvectors of MTM form columns of V V*= VT



```
In [8]: from numpy.linalg import svd

In [9]: def compress_grayscale_svd(mat: list, k: int):
   ...:   u, s, v = svd(mat, full_matrices=False)
   ...:   reconstructed_matrix = Num.dot(u[:,:k], Num.dot(Num.diag(s[:k]), v[:k,:]))
   ...:   return reconstructed_matrix, s
In [10]: def get_compression_ratio(k, shape1,shape2):
    ...:     return 100.0 * ( k * (shape1 + shape2) + k) / (shape1 * shape2)
    ...:
    ...: def show_compressed_grayscale_image(img, k):
    ...:     reconstructed_img, sigma = compress_grayscale_svd(img, k)
    ...:     figure, xy_axises = pyplot.subplots(1, 2, figsize=(8, 5))
    ...:     xy_axises[0].plot(sigma)
    ...:     compression_ratio = get_compression_ratio(k, img.shape[0], img.shape[1])
    ...:     xy_axises[1].set_title("Compression Ratio is {:.2f}%".format(compression_ratio))
    ...:     xy_axises[1].imshow(reconstructed_img, cmap='gray')
    ...:     xy_axises[1].axis('off')
    ...:     figure.tight_layout()
    ...:
```

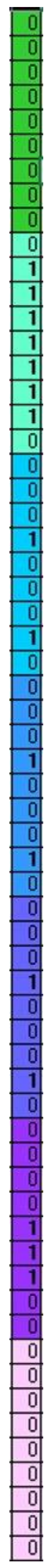
Watermarking



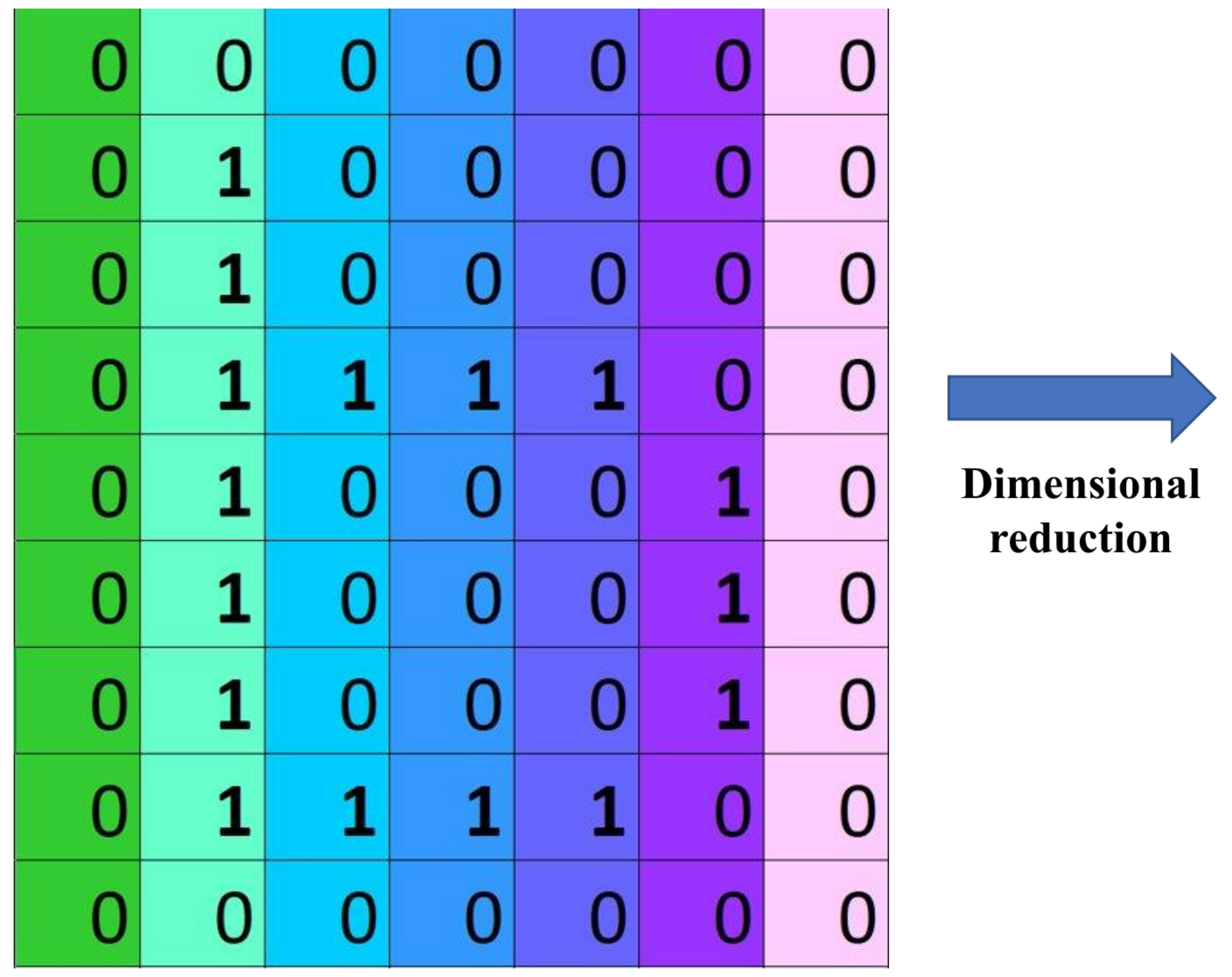
Conclusion

• We find that the embedded watermark is tamper resistant and the watermark could be retrieved from manipulated greyscale images subject to rotation and compression distortions. Preliminary studies using greyscale images suggest that SVD based digital watermarking methods are robust and can be used to verify and authenticate data ownership.

• This project integrates advanced application of linear-algebra-based methods with Python based programming methods to provide solutions to real world problems of current interest.

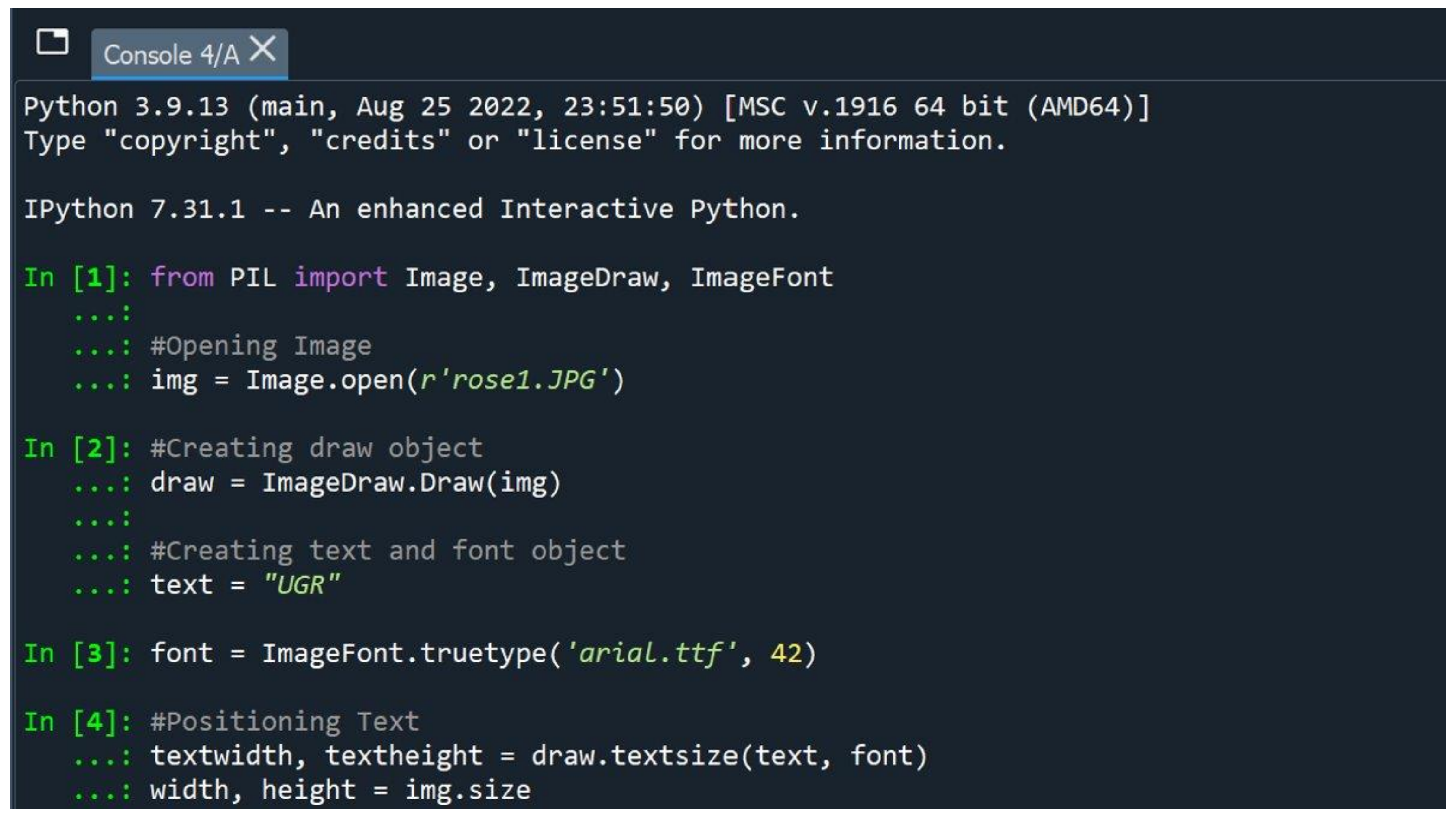


SVD Dimensional Reduction



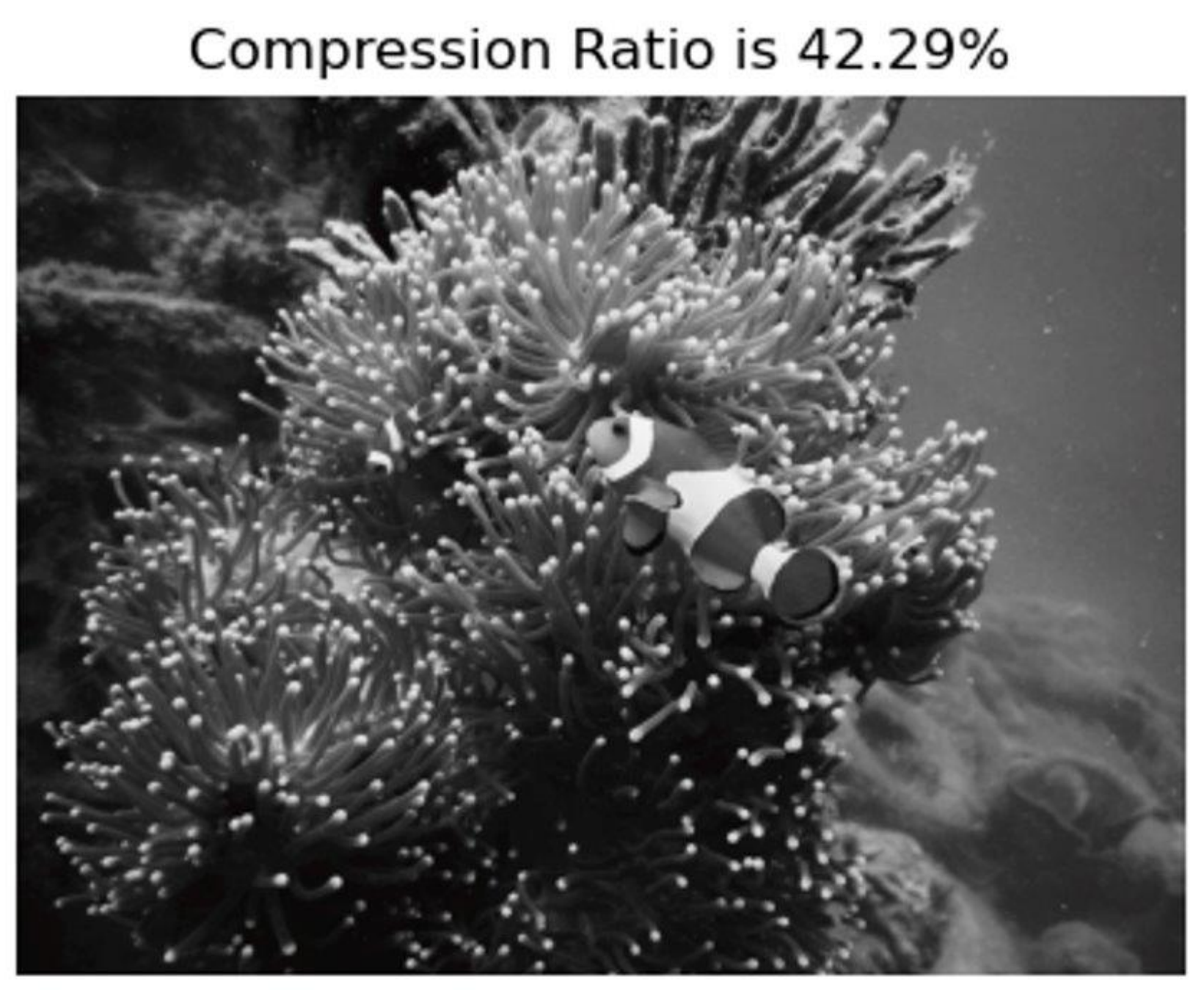
Python Code and Libraries

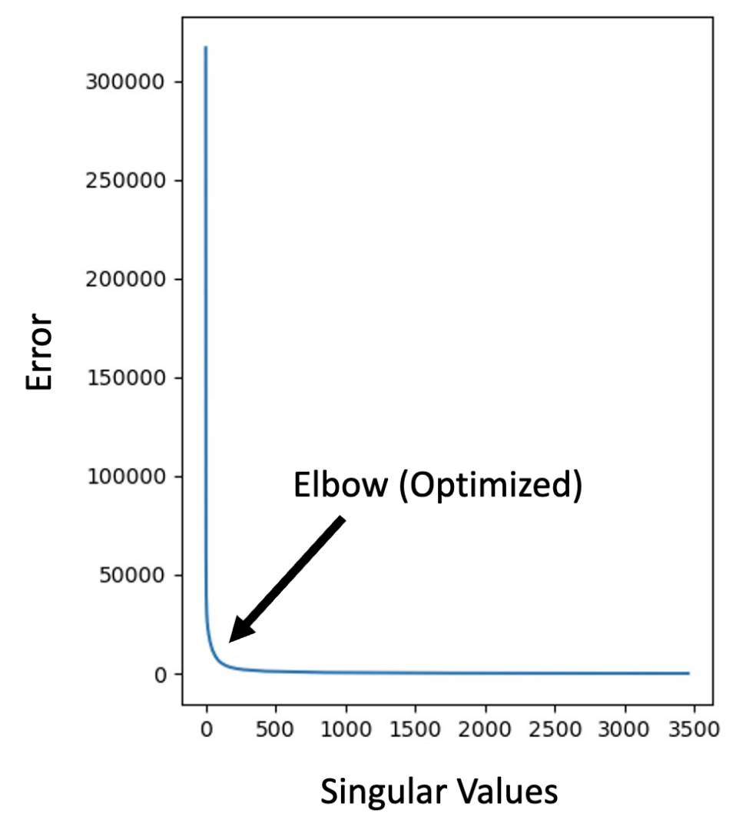
Using IPython and Numpy, we characterized color and grayscale images as arrays and implemented singular value decompositions (SVD) and principal component analysis (PCA) for grayscale and color image processing studies.

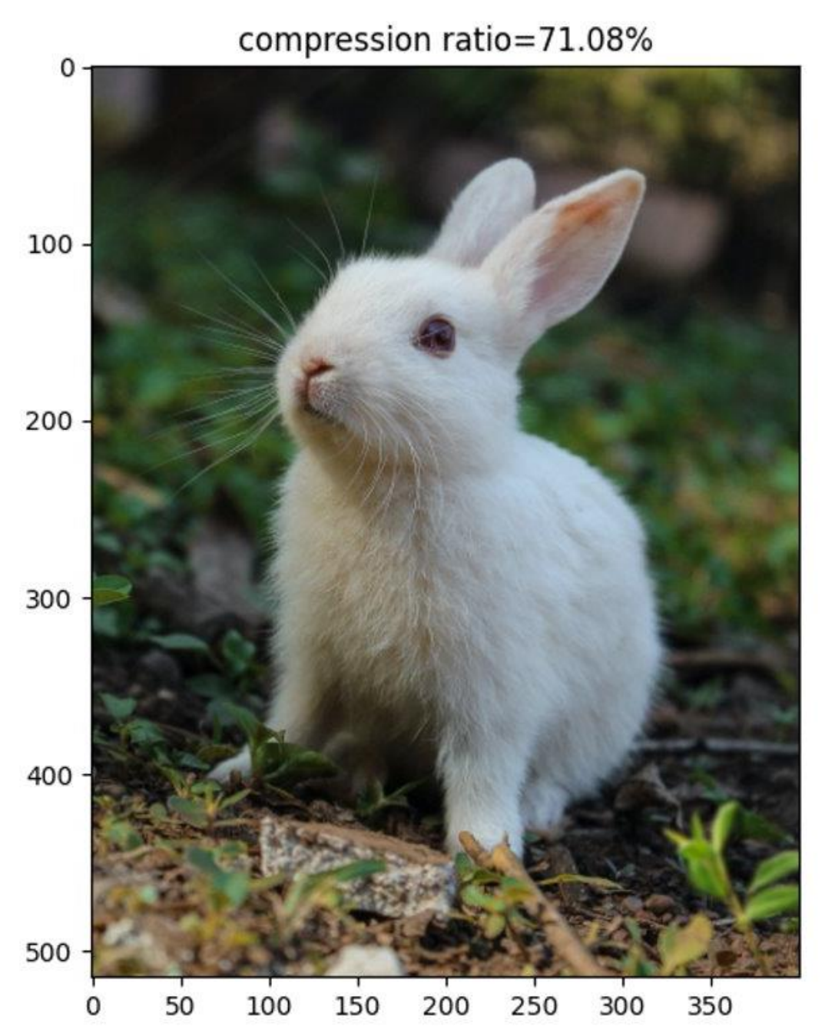


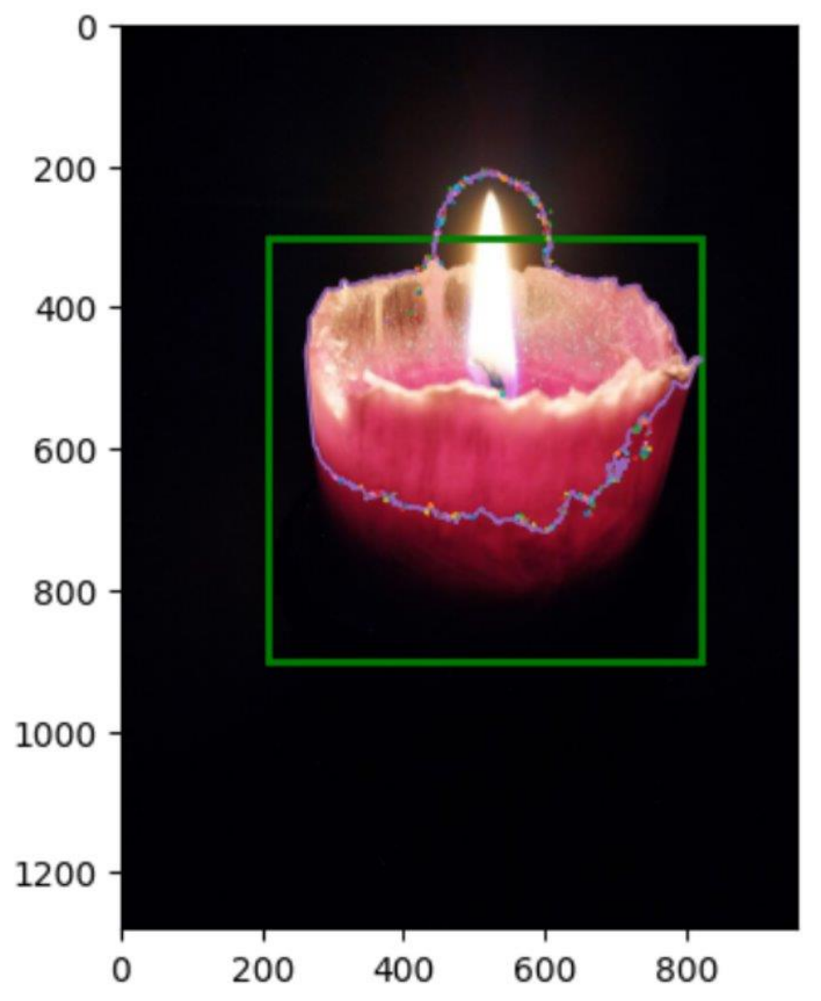
```
Console 4/A

Python 3.9.13 (main, Aug 25 2022, 23:51:50) [MSC v.1916 64 bit (AMD64)]
Type "copyright", "credits" or "license" for more information.

IPython 7.31.1 -- An enhanced Interactive Python.

In [1]: from PIL import Image, ImageDraw, ImageFont
   ...:
   ...: #Opening Image
   ...: img = Image.open(r'rose1.JPG')

In [2]: #Creating draw object
   ...: draw = ImageDraw.Draw(img)
   ...:
   ...: #Creating text and font object
   ...: text = "UGR"

In [3]: font = ImageFont.truetype('arial.ttf', 42)

In [4]: #Positioning Text
   ...: textwidth, textheight = draw.textsize(text, font)
   ...: width, height = img.size
```

Image Results









References

https://medium.com/@rameshputalapattu/jupyter-python-image-compression-and-svd-an-interactive-exploration-703c953e44f6

https://github.com/rameshputalapattu/jupyterexplore

https://devopedia.org/principal-component-analysis

https://scikit-image.org/

Lake Washington Institute of Technology™
Practical. Purposeful. Promising.