Luleå University of Technology, Department of Computer Science, Electrical and Space Engineering

Assignment 1

Data Visualization (D7055E)

GROUP NO. 13 members:
Madelene Gustafsson
Carlos González Mendoza
Fatima Juairiah
Michelle Lima Ruda
Talha Imran

Date: November 17, 2025

1. Contributions

For the first assignment, everyone started coding the assignment individually at their own pace to learn the basics, including the practice exercise with Canada.xlsx. Once everyone had finished, we scheduled a Zoom meeting to review each person's code and compare approaches, so we could learn from one another and finish assignment 1 together. After that, we divided the report so that everyone could contribute by writing the report equally, and then choosing the best visualizations to include in the report.

1. Introduction

This report presents the work completed for Assignment (1) in the course Data Visualization D7055E. The purpose of the assignment is to practice the full workflow of creating clear and informative visualizations using Python. This includes loading the required datasets, performing data cleaning, transforming the data when necessary, and generating visual representations that support meaningful interpretation.

The assignment consists of seven separate tasks (Q1–Q7), each focusing on different aspects of visualization. The first three tasks involve analysing and comparing yearly averages of WTI and Brent oil prices between 1992 and 2002. Later tasks explore bar and pie charts using sales data and basic image transformations such as colormaps and filters. For each task, the report includes snapshots of the produced visualizations as well as short observations describing the observations identified from the figures.

2. Data Preparation and Cleaning

First, the relevant data was loaded in the required formats. The datasets 'wti-daily.csv', 'brent-daily.csv', and 'sales.xlsx' were loaded as DataFrames. As for the image '2.jpg', it was first loaded as a NumPy array for Question 6 and then as PIL image for Question 7. Before modifying anything, the data was visualized in order to understand what types of data we were working with, what they represent, and how they are structured.

Next, several checks were performed to ensure that there were no duplicated rows or missing values. We also made sure that the data types used to represent the information were consistent.

For each of the questions, the corresponding changes were applied as needed. In particular:

- In the 'wti-daily.csv' and 'brent-daily.csv' datasets, the 'Date' column was set as the index and then converted to a datetime type (it was originally stored as an object).
- The image '2.jpg' was loaded as a NumPy array.

Overall, a clear and structured workflow was followed to keep the data clean, prevent potential errors, and meet the project requirements.

3. Visualizations and Observations/Insights

Q1.

To answer Q1, the daily WTI oil price dataset was first resampled into yearly averages using the resample('YE')method. Only the years 1992–2002 were selected according to the assignment instructions. The resulting values were plotted in a simple line chart that displays the long-term trend of the WTI price over the selected period. The x-axis shows each year, and the y-axis shows the corresponding average price in USD per barrel.

- The WTI price shows a general upward trend across the decade, although with several notable fluctuations.
- Prices dropped significantly around 1998, reaching the lowest point in the selected range.
- After 1998, there was a sharp recovery, with the price rising steeply towards 2000–2001.
- The overall pattern suggests sensitivity to global market conditions, with both short-term volatility and long-term growth visible in the data.

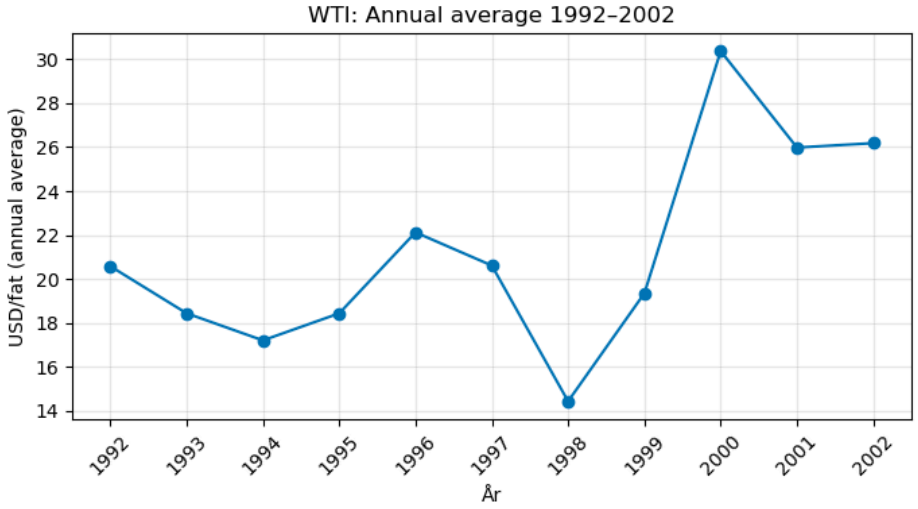


Figure 1. WTI: Annual average 1992-2002

Q2.

The Brent dataset was processed in the same way as for WTI in Q1_ daily prices were resampled to yearly averages, and only the years 1992–2002 were included. The resulting values were plotted in a line chart showing how the Brent price developed over the selected period.

- Brent follows a very similar trend to WTI, with a decline during the early 1990s, reaching a low point around 1994.
- The price increases again towards 1996, followed by another drop that reaches a clear minimum in 1998, where Brent falls to about 13 USD/fat.
- After 1998, the price climbed sharply, peaking around 2000 at nearly 29 USD/fat.
- The period ends with Brent stabilising in the mid-20s during 2001–2002.

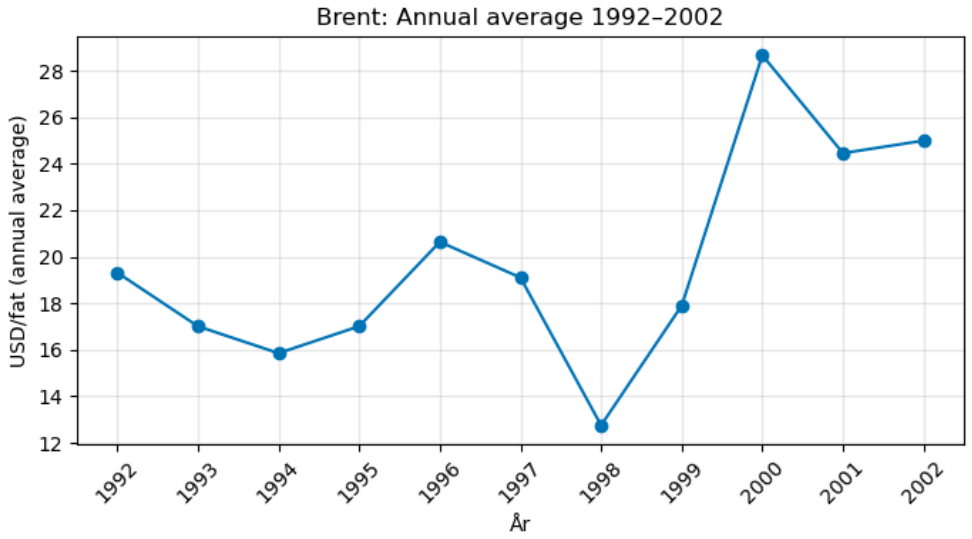


Figure 2. Brent: Annual average 1992-2002

Q3.

For Q3, the yearly averages of WTI and Brent were combined into a single dataset in order to visualize both trends in the same chart. This allows direct comparison between the two oil benchmarks over the period 1992–2002. Both lines were plotted together, and the three years with the largest absolute differences (Δ) between WTI and Brent were identified and highlighted with black "x" markers. The corresponding Δ-values were shown as text to the right of the chart for clarity.

- Throughout the entire period, WTI and Brent follow very similar overall trends, showing almost parallel movement in most years.
- The largest price differences appear in 1998, 2000 and 2001, where WTI is consistently slightly higher than Brent. The Δ-values are moderate (between 1.5–1.7 USD/fat), indicating that even at their widest separation, the two benchmarks remain closely aligned.
- Similar to Q1 and Q2, both series hit their lowest point in 1998, followed by a sharp price recovery leading up to 2000–2001. The simultaneous rise confirms that global market factors affected both benchmarks in a comparable way.

Overall, the comparison demonstrates that while small differences exist between WTI and Brent, their long-term patterns are almost identical, which could reflect the connected nature of the global oil market during this period.

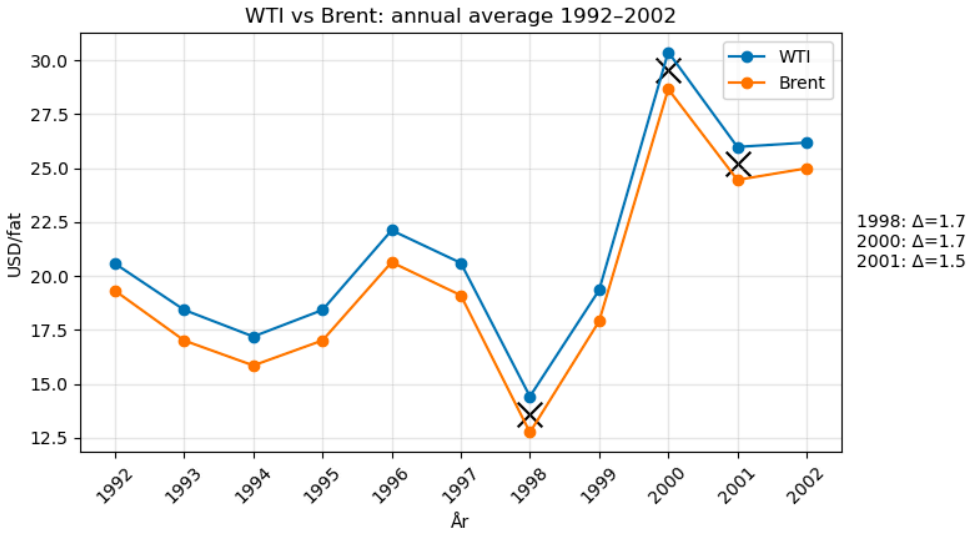


Figure 3. WTI vs. Brent: annual average 1992-2002

Q4.

The sales dataset shows Financial Data and Product Details of three cities in Myanmar – Yangon, Naypyitaw and Mandalay.

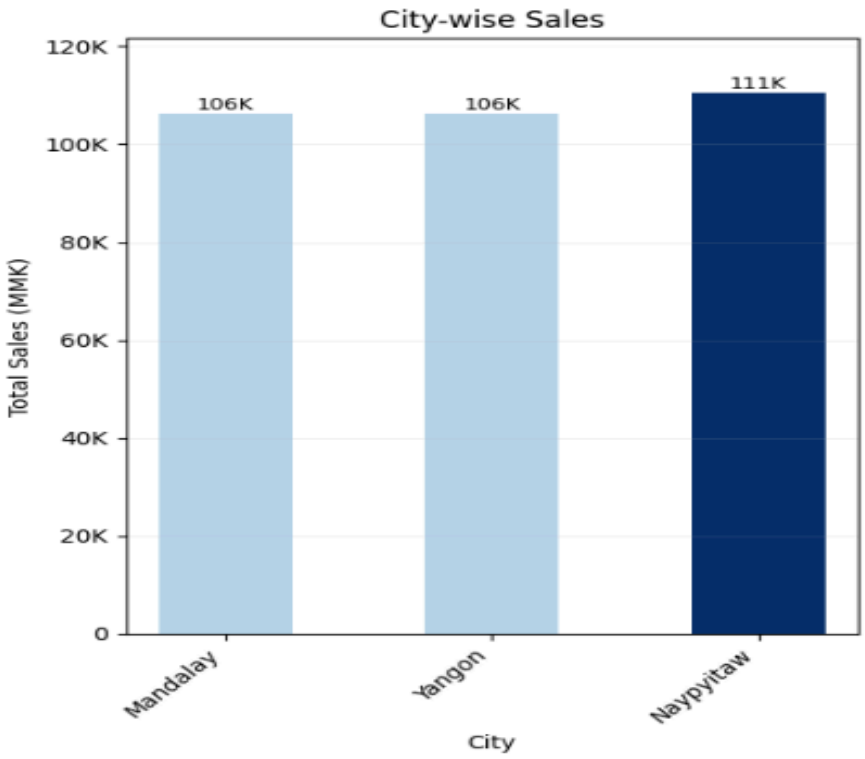


Figure 4. City Wise Sales Distribution

Here we visualized the city wise sales.

- We aggregated the total sales by city using the sales dataset, that represents the sum of sales for each city, providing a clear view of which cities contribute most to overall sales.
- We preferred a vertical bar chart as it can clearly display the total sales for each city with vertical bars, making it easy to compare sales values across different cities. The formatting includes color gradients, value labels, and formatted axes.
- We applied a color gradient to the bars, where darker shades represent higher sales values, enhancing the visual distinction.
- Naypyitaw recorded the highest total sales at approximately 111K Myanmar Kyat (MMK), while Mandalay and Yangon showed nearly equal sales figures, trailing closely behind.

Q5.

We see payment methods have an impact on sales using a bar chart and a pie chart.

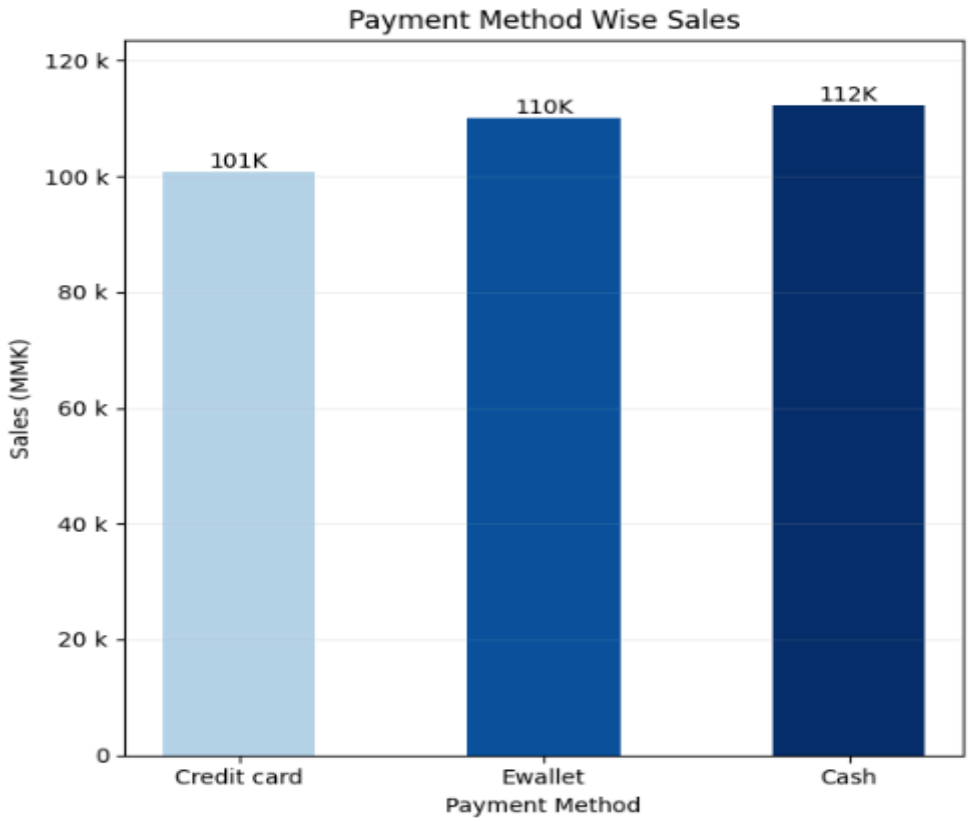


Figure 5.1. Sales Distribution by Payment Method

- Cash had the highest sales at about 112K Myanmar Kyat (MMK), followed by Ewallet at 110K MMK, while credit card sales were the lowest at around 101K MMK.
- The max number of people in these cities prefer the payment method as Cash.

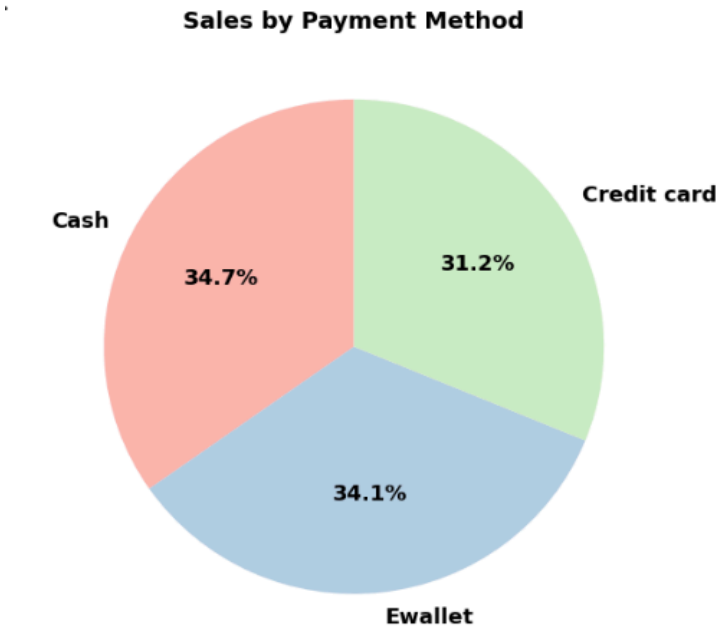


Figure 5.2. Sales Distribution by Payment Method (Pie Chart)

We also plotted the pie chart using percentage values of total sales by payment method, which provides a clear view of how closely the categories are proportioned. The pie chart shows the distribution of sales by payment method. Cash accounts for 34.7%, E-wallet for 34.1%, and Credit Card for 31.2%. The three categories have very similar proportions, so their slices occupy almost the same area in the pie chart.

Q6.

For loading the image, matplotlib.image.imread was used, which already loads the image directly as a NumPy array. Alternatively, PIL.Image.open could have been used, followed by converting the resulting PIL.Image.Image object into a NumPy array as required.

After completing the loading steps mentioned above, the next step was to convert the image (RGB) into a grayscale format and obtain the data as a 2D array. Then, the colormaps to be applied to the image were selected. These were:

- **gray:** neutral version
- **viridis:** uniform contrast, excellent for analysis
- **hot:** highlights bright areas (useful for energy, heat, density)
- **coolwarm:** useful for observing transitions or symmetries
- **hsv:** very colorful; useful for periodicity or angular information

- **plasma:** enhances contours and textures

The result of applying these colormaps is shown below:

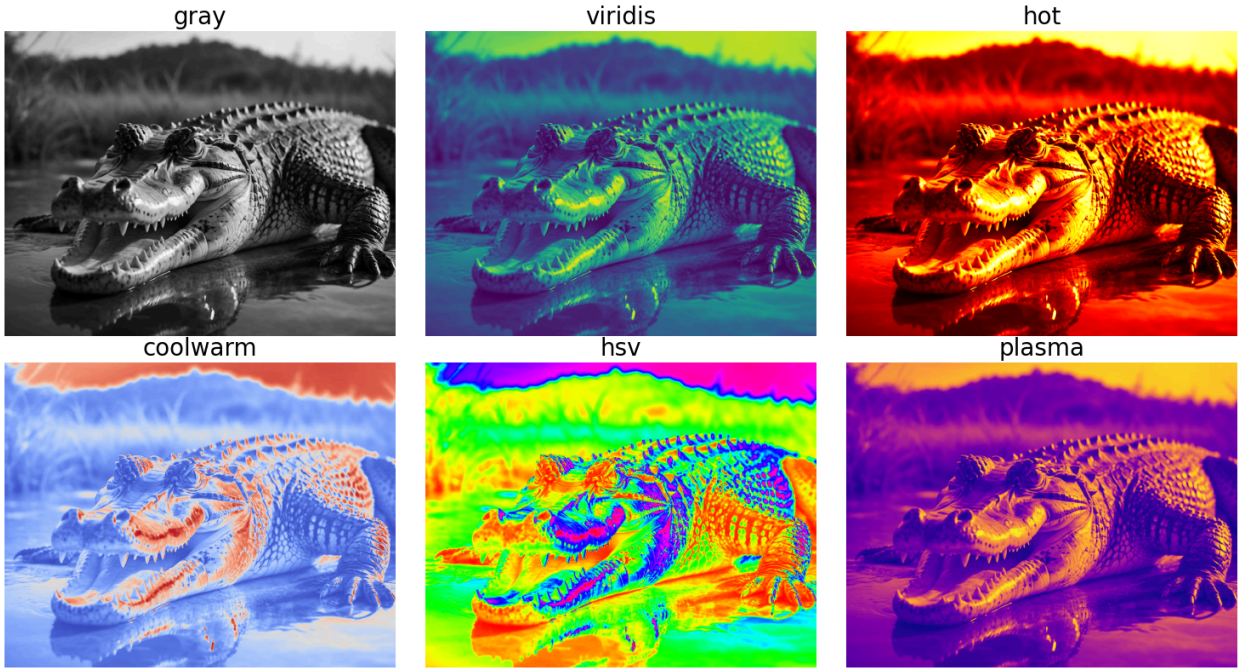


Figure 6. Colormaps applied to '2.jpg'

What we're doing when converting the image to grayscale is extracting only intensity value (one number per pixel). This way, each intensity value (from 0 to 255 or from 0 to 1) is mapped to a color according to a chosen colormap. This is mainly useful for highlighting patterns, improving perceptual contrast, and making visual analysis easier.

We can say that applying colormaps does not change the underlying image, but it changes how our brain interprets it, making the image:

- easier to detect patterns
- easier to distinguish intensities
- more structurally revealing
- more informative overall

Q7.

Here, matplotlib.image.imread() was used to read the file *"2.jpg"*, which automatically loads the image as a NumPy array. Since the Pillow enhancement functions require a PIL image, it was first converted the array into a PIL.Image object.

After that, we applied six different filters, each demonstrating a different type of image transformation:

1. **Brightness Adjustment:** This helps highlight darker regions and improves visibility in low-light areas.

2. **Contrast Enhancement:** Contrast amplification makes the difference between bright and dark regions more pronounced. This reveals patterns and improves structure visibility.
3. **Sharpening:** Sharpening enhances fine details and textures (e.g., edges of objects in the image). It makes the image look clearer and more defined.
4. **Gaussian Blur:** Gaussian Blur removes high-frequency noise, useful for denoising and seeing coarse structure.
5. **GrayScale Conversion:** This converts the image from RGB to a single intensity channel, making it easier to analyse brightness patterns and apply filters like edge detection.
6. **Edge Detection:** This identifies sharp changes in intensity, highlighting boundaries and outlines in the image.
   It is useful for:
   - Shape detection
   - Structure analysis
   - Feature extraction before computer vision algorithms

.resize((width//2, height//2)) was then used. This filter demonstrates geometric transformations by scaling the image. It is commonly used in:

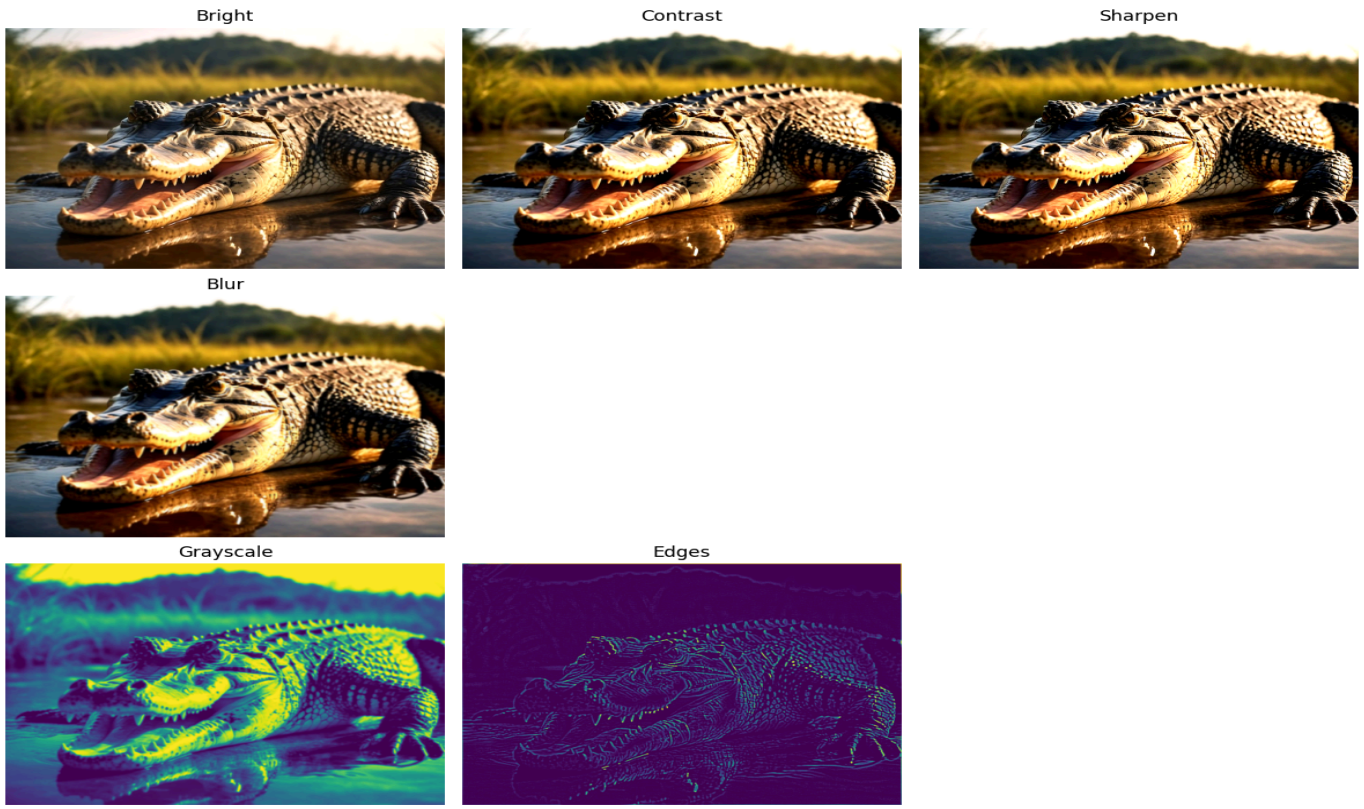
- Compression
- Thumbnail generation
- Reducing memory usage

To match the assignment structure, a 3×3 grid of subplots was used. Only positions 1, 2, 3 (first row), 4 (second row), 7 and 8 (third row) were filled.

This creates the required layout:

- 3 images in row 1
- 1 image in row 2
- 2 images in row 3
- leaving some empty cells intentionally

Grayscale and Edge-detected images were displayed using: cmap='gray'. because they are single-channel images. All other filters display their natural RGB colors. The results for applying these basic filters are shown below:



Conclusion: The applied filters together enhance visibility, highlight important structures, and simplify the image for analysis. Brightness/contrast adjustments improve clarity, sharpening emphasizes details, and Gaussian blur smooths noise. Grayscale conversion and edge detection reveal intensity patterns and boundaries, while resizing makes the image more practical for storage and processing.

5. References

(IEEE format like the following or another one of your choice)
[1] Matplotlib Developers, *matplotlib.image.imread — Matplotlib documentation*. Available: https://matplotlib.org/stable/api/image_api.html#matplotlib.image.imread

[2] Matplotlib Developers, *Choosing Colormaps in Matplotlib*. Available: https://matplotlib.org/stable/users/explain/colors/colormaps.html