AI/ML Course Assignment

Linear Algebra, Python, NumPy & OpenCV Image Processing

Samriddha Pathak

July 30, 2025

Assignment Overview
Topics Covered: Linear Algebra, Python, NumPy, OpenCV Image Processing
Difficulty Level: Easy to Intermediate
Total Questions: 8

1 Question 1: Linear Algebra Fundamentals

Write a Python function that performs the following operations on two matrices $\mathbf{A}$ and $\mathbf{B}$:

- Matrix multiplication ($\mathbf{A} \times \mathbf{B}$)
- Element-wise multiplication ($\mathbf{A} \odot \mathbf{B}$)
- Calculate the determinant of matrix $\mathbf{A}$
- Find the transpose of matrix $\mathbf{B}$ ($\mathbf{B}^T$)

Requirements:

- Use NumPy for all operations
- Handle cases where operations are not possible (e.g., incompatible dimensions)
- Test your function with at least two different matrix pairs

Sample Input:

```
A = [[2, 3], [1, 4]]
B = [[5, 2], [3, 1]]
```

2 Question 2: NumPy Array Manipulation

Create a Python program that:

1. Generates a 6×6 matrix filled with random integers between 10 and 100
2. Extracts the diagonal elements and stores them in a separate array
3. Replaces all even numbers in the original matrix with their square roots
4. Finds the mean, median, and standard deviation of the modified matrix
5. Reshapes the matrix into a 4×9 matrix

Bonus: Implement error handling for any potential mathematical errors.

3 Question 3: Image Loading and Basic Processing

Using OpenCV, write a program that:

1. Loads a color image from your local system
2. Converts the image to grayscale using OpenCV functions
3. Displays both the original and grayscale images side by side
4. Saves the grayscale image with a new filename
5. Prints the dimensions (height, width, channels) of both images

Requirements:

- Use `cv2.imread()`, `cv2.cvtColor()`, `cv2.imshow()`, and `cv2.imwrite()`
- Include proper window management (`cv2.waitKey()` and `cv2.destroyAllWindows()`)

4 Question 4: Image Resizing and Geometric Transformations

Develop a function that takes an image and performs the following operations:

1. Resize the image to three different sizes: 50% smaller, 200% larger, and fixed size 300×300 pixels
2. For each resized image, calculate and display:
   - Original dimensions
   - New dimensions
   - Scaling factors used
3. Save all resized images with descriptive filenames

Challenge: Maintain aspect ratio for one of the resizing operations and compare the results.

5 Question 5: Edge Detection and Blurring Techniques

Create a comprehensive image filtering program that:

1. Applies three different types of blurring to an input image:
   - Gaussian blur (kernel size 15×15)
   - Average blur (kernel size 10×10)
   - Median blur (kernel size 9)
2. Performs edge detection using:
   - Canny edge detection
   - Sobel edge detection (both X and Y gradients)
3. Creates a combined display showing all results in a grid format
4. Allows the user to adjust blur parameters through command-line arguments

6 Question 6: Face Detection Implementation

Implement a face detection system using Haar cascades that:

1. Loads a pre-trained Haar cascade classifier for face detection
2. Detects faces in both static images and real-time webcam feed
3. Draws bounding rectangles around detected faces with:
    - Different colors for different face sizes (small, medium, large)
    - Displays the number of faces detected
4. Saves images with detected faces marked
5. Implements confidence thresholding to reduce false positives

Requirements:

- Handle cases where no faces are detected
- Include proper error handling for webcam access
- Test with multiple images containing different numbers of faces

7 Question 7: Linear Algebra in Image Processing

Write a program that demonstrates the application of linear algebra in image processing:

1. Load a color image and convert it to grayscale
2. Represent the grayscale image as a 2D NumPy array (matrix)
3. Apply the following matrix operations:
    - Calculate the covariance matrix of image patches (use 5×5 patches)
    - Perform eigenvalue decomposition on the covariance matrix
    - Apply a custom transformation matrix to brighten/darken the image
4. Compare the original and transformed images
5. Plot histograms showing the pixel intensity distributions before and after transformation

Mathematical Component: Explain in comments how each linear algebra operation affects the image.

8 Question 8: Integrated Project - Image Analysis Pipeline

Design and implement a complete image analysis pipeline that combines all learned concepts:

8.1 Pipeline Requirements:

1. **Input Stage:** Load multiple images from a directory
2. **Preprocessing Stage:**
   - Resize all images to a standard size (512×512)
   - Convert to grayscale
   - Apply noise reduction using Gaussian blur
3. **Feature Extraction Stage:**
   - Detect edges using Canny edge detection
   - Find faces using Haar cascades (if any)
   - Calculate basic image statistics (mean intensity, contrast)
4. **Analysis Stage:**
   - Create a summary matrix containing features from all images
   - Use linear algebra operations (matrix multiplication, eigenvalues) to analyze the feature matrix
   - Identify the image with the highest edge density
5. **Output Stage:**
   - Generate a report showing results for each image
   - Create a collage showing original and processed versions
   - Save all intermediate results

8.2 Advanced Requirements:

- Implement the pipeline as a class with modular methods
- Add progress tracking for batch processing
- Include comprehensive error handling
- Generate a statistical summary of the entire batch

9 Submission Guidelines

9.1 Code Requirements:

- All code must be well-commented and follow PEP 8 style guidelines
- Include docstrings for all functions and classes
- Implement proper error handling and input validation
- Use meaningful variable names and organize code into logical functions

9.2 Documentation:

- Include a README file explaining how to run each program
- Provide sample outputs or screenshots for visual programs
- List all required dependencies and installation instructions
- Explain any assumptions made in your implementations

9.3 Testing:

- Test each program with multiple inputs
- Include edge cases in your testing
- Provide sample input files where applicable
- Document any limitations or known issues

9.4 File Structure:

```
assignment_submission/
          question_1/
                linear_algebra_ops.py
                test_matrices.py
          question_2/
                numpy_manipulation.py
          question_3/
                basic_image_processing.py
                sample_images/
          question_4/
                image_resizing.py
          question_5/
                filtering_and_edges.py
          question_6/
                face_detection.py
                haarcascade_frontalface_default.xml
          question_7/
                linear_algebra_imaging.py
          question_8/
                image_pipeline.py
                pipeline_class.py
                batch_results/
          README.md
```

9.5 Evaluation Criteria:

Criterion	**Weight**
Correctness (Programs produce expected outputs)	40%
Code Quality (Clean, readable, well-structured code)	25%
Documentation (Clear comments and explanations)	15%
Error Handling (Robust handling of edge cases)	10%
Creativity (Innovative approaches and additional features)	10%

Due Date: [5th August 2025]

10 Additional Resources

- OpenCV Documentation: `https://docs.opencv.org/`
- NumPy User Guide: `https://numpy.org/doc/stable/user/`
- Sample Haar Cascade files: OpenCV GitHub repository
- Test images: Use diverse images including portraits, landscapes, and group photos