Computer Vision - Spring 25

Assignment 1

Deadline : 23 January 2025 11:55 P.M.

Instructors: Prof Ravi Kiran Sarvadevabhatla
and Prof Makarand Tapaswi

January 12, 2025

1 General Instructions

- Your assignment must be implemented in Python.
- While you're allowed to use LLM services for assistance, you must explicitly declare in comments the prompts you used and indicate which parts of the code were generated with the help of LLM services.
- Plagiarism will only be taken into consideration for code that is not generated by LLM services. Any code generated with the assistance of LLM services should be considered as a resource, similar to using a textbook or online tutorial.
- The difficulty of your viva or assessment will be determined by the percentage of code in your assignment that is not attributed to LLM services. If during the viva you are unable to explain any part of the code, that code will be considered as plagiarized.
- Clearly label and organize your code, including comments that explain the purpose of each section and key steps in your implementation.
- Properly document your code and include explanations for any non-trivial algorithms or techniques you employ.
- Ensure that your files are well-structured, with headings, subheadings, and explanations as necessary. Your assignment will be evaluated not only based on correctness but also on the quality of code, the clarity of explanations, and the extent to which you've understood and applied the concepts covered in the course.
- Make sure to test your code thoroughly before submission to avoid any runtime errors or unexpected behavior.

- **Report all the analysis, comparison and any metrics in the notebook or a separate report that is part of the submission itself. No external links to cloud storage files, wandb logs or any other alternate will be accepted as part of your submission. Only the values and visualizations as part of your commits will be graded.**

2 Cloverleaf Bridge (80 Marks)

This problem focuses on applying digital image processing techniques to analyze an aerial image of a cloverleaf interchange. You will implement algorithms to detect the structures forming the clover leaves, measure their dimensions, and calculate the approximate area enclosed by each leaf of the clover.

2.1 Dataset and Preprocessing (20 Marks)

You are provided with an aerial image of a cloverleaf interchange: `cloverleaf_interchange.png`, available here. The image is also shown in Figure 1.

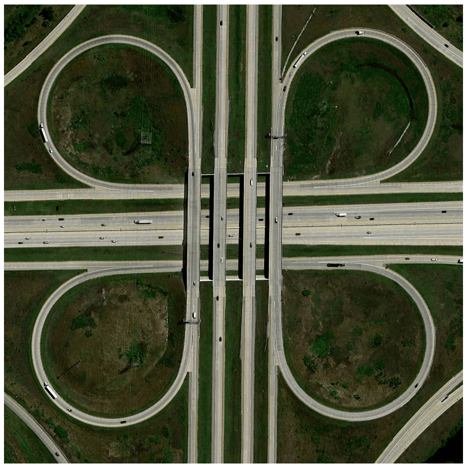


Figure 1: Cloverleaf Bridge

2.1.1 Image Loading and Exploration (5 Marks)

- ✓ Implement a function `load_cloverleaf_image()` that reads the provided image.
- ✓ Display the original image and provide basic information about it (e.g., dimensions, pixel value range, and distribution).
- ✓ Implement the histogram function independently and then compare your version with the one provided by OpenCV.

2.1.2 Image Preprocessing (15 Marks)

- ✓ Implement a function `preprocess_image(image)` that applies appropriate preprocessing techniques to enhance the image for the detection of clover leaves. Note that the clover leaves form a quarter-and-a-half circle.

You may consider techniques for color conversion, noise reduction, edge detection, etc.

- Display the preprocessed image after each major step.
- Justify your choices of preprocessing techniques and parameters.

2.2 Cloverleaf Detection and Measurement (60 Marks)

This section focuses on implementing algorithms to detect the circular structures in the image, measure their radii, and approximate the area they enclose.

2.2.1 Cloverleaf Detection and Visualization (25 Marks)

- Implement a function `detect_cloverleaves(image)` that detects the circular structures that form the cloverleaves in the image. Note that the shape might be slightly distorted.
- Output an image in which the detected circles are clearly marked.
- Analyze the impact of preprocessing techniques on the image and its suitability for detection.
- In your report, describe your chosen method and explain your parameter choices. Discuss any challenges in detecting clover leaf circles and how you addressed them.
- Analyze cases where your algorithm might fail or produce suboptimal results.

2.2.2 Measuring the Radii of the Detected Cloverleaves (20 Marks)

- Extend your function from Question 2.2.1 to `calculate_radii(image)` to estimate the radius of each detected cloverleaf circle in pixels.
- Print a list of estimated radii (in pixels) for each detected circle.
- Display an image with the detected circles, their centers, and their radii clearly marked and labelled.
- Explain how your chosen detection method provides the information needed to calculate the radius. Discuss any assumptions made and potential sources of error in your estimation.

2.2.3 Approximating the Area Enclosed by Each Cloverleaf (15 Marks)

- Implement a function `calculate_area(image)` that approximates the area enclosed by each detected cloverleaf. You can use the calculated radius to approximate the area (assuming a perfect circle) or explore other methods like counting pixels within the detected regions. Compare different methods and report you observations.

- Report the approximate area enclosed by each cloverleaf (in square pixels).

3 Line Segmentation in Historical Document (100 Marks)

Optical Character Recognition (OCR) systems often rely on accurate text line segmentation as a crucial preprocessing step. This problem focuses on developing simple methods for detecting and segmenting text lines within a historical document image. You will implement algorithms to extract lines based on rectangular bounding boxes, tighter polygon boundaries, and segment lines within a circular region, mimicking the challenges faced in real-world OCR pipelines.

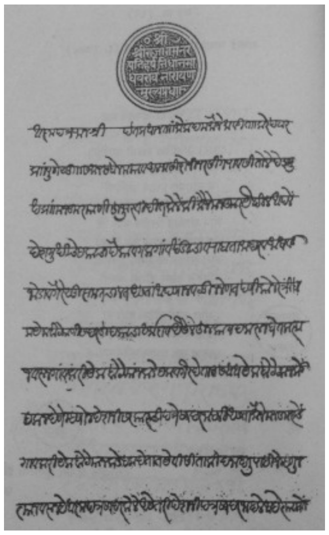


Figure 2: Ancient Document

3.1 Document Image Preprocessing (20 Marks)

You have been provided with a historical document image: `historical-doc.png`, available here. This image contains text lines with varying orientations and a circular seal with text inside.

3.1.1 Data Loading and Exploration (5 Marks)

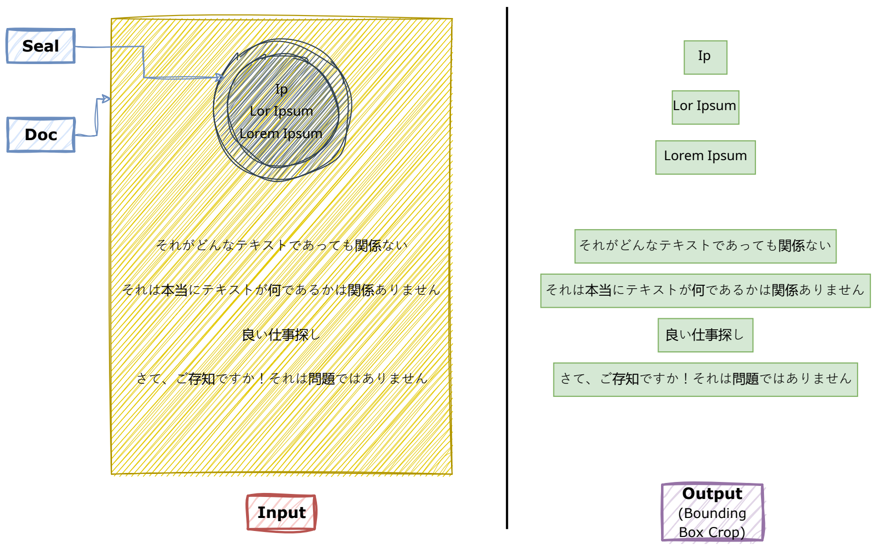
- Implement a function `load_document_image()` that reads the provided image.
- Display the original image and provide basic information about it, such as its dimensions, pixel value range and distribution (histogram).

3.1.2 Image Preprocessing for Text Line Detection (15 Marks)

- Implement a function `preprocess_image(image)` that applies a sequence of image processing techniques to enhance the image for text line detection. You may consider techniques that might be useful for extracting lines like grayscale conversion, noise reduction, binarization, and morphological operations.
- **Evaluation:**
  - Display the image after each major preprocessing step.
  - Justify your choices of preprocessing techniques and parameters in your report. Analyze their impact on the image and its suitability for subsequent text line detection.

3.2 Text Line Detection and Segmentation (50 Marks)

This section focuses on implementing algorithms to detect and segment text lines in the preprocessed image. An illustrative example of the output has been provided in Figure 3



Seal
Doc
Ip
Lor Ipsum
Lorem Ipsum
それがどんなテキストであっても関係ない
それは本当にテキストが何であるかは関係ありません
良い仕事探し
さて、ご存知ですか！それは問題ではありません
Input
Ip
Lor Ipsum
Lorem Ipsum
それがどんなテキストであっても関係ない
それは本当にテキストが何であるかは関係ありません
良い仕事探し
さて、ご存知ですか！それは問題ではありません
Output
(Bounding Box Crop)

Figure 3: Sample Output

3.2.1 Rectangular Bounding Box Detection (25 Marks)

- Implement a function `detect_lines_with_bboxes(image)` that detects text lines and encloses each with a rectangular bounding box.

- Choose and implement appropriate image processing algorithms for detecting text lines and determining the coordinates of the bounding boxes.
- **Output:**
  - Display the original image with the detected bounding boxes overlaid on the text lines.
  - Save each cropped text line as a separate image file (e.g., `line_1.png`, `line_2.png`, etc.).
- Discuss the performance of your algorithm. Analyze cases where it might fail or produce suboptimal results.

3.2.2 Line Segmentation within the Circular Region (25 Marks)

- Implement a function `segment_lines_in_seal(image)` that detects and segments text lines within the circular seal.
- Adapt your line detection algorithm from Question 3.2.1 or develop a new method to segment the text lines within the circular region.
- **Output:**
  - Display the detected circular seal (which contains text) overlaid on the original image.
  - Save each segmented text line within the circle as a separate image file (e.g., `circle_line_1.png`, etc.).
- Discuss the challenges of segmenting the lines inside the seal compared to the main document.

3.3 Tighter Polygonal Boundary Detection (30 Marks)

This section explores a more refined approach to text line segmentation using polygonal boundaries.

- Implement a function `detect_lines_with_polygons(image)` that detects a tighter polygonal boundary around each text line.
- Display the original image with the detected polygonal boundaries overlaid on the text lines.
- Compare the polygonal boundaries with the rectangular bounding boxes.
- Discuss the advantages and disadvantages of using polygons. Analyze the impact of your chosen method for polygon on the results.

4 MLP Classification on Image Features (70 Marks)

4.1 Dataset Analysis and Preprocessing (10 Marks)

1. Load the dataset linked here. Use the train and test csv files by reshaping the 784-pixel columns to obtain the images. Split the train set further to get your validation set (5 Marks)

2. Visualize examples of images from the dataset and their labels to understand the variety of clothing items. (5 Marks)

3. List the clothing types represented by each class label (e.g., T-shirt, Shoe, etc.). Avoid looking up external references, and rely on your own exploration.

4.2 Model Implementation and Training (25 Marks)

1. Train three separate MLP models, each using different input features:
   - **Model 1:** Flattened raw images (1D vector) without any additional preprocessing.
   - **Model 2:** Edge-detection features using techniques like Canny features.
   - **Model 3:** Another transformation, such as HOG features or anything else (use your own creativity for feature extraction, limited to basic image processing features).

2. Each model should be implemented as a simple multi-layer perceptron (MLP) using the `pytorch` library, with appropriate layers, activation functions, and dropout (if required). Ensure each model is trained for sufficient epochs to reach convergence. (10 Marks)

3. Evaluate each model using metrics such as accuracy, precision, recall, F1-score, and a confusion matrix on the test dataset. Save the metrics for comparison. Use `wandb` to (15 Marks)

4.3 Plots and Performance Visualizations (25 Marks)

1. Train your models on different hyperparameters of the MLP and feature extractor chosen. Try out at least 3 variants for each of the models and report the basic results for all of them, while giving detailed analysis for the rest (10 Marks)

2. Plot training and validation accuracy/loss curves over epochs for all three best models. Compare their convergence behavior and discuss any observations. Use `wandb` to log all the metrics for all hyperparameters(5 Marks)

3. Visualize example outputs for each of the best models, such as misclassified images or confusion matrices, to understand their strengths and weaknesses. (5 Marks)

4. Create a table summarizing the performance metrics (accuracy, precision, recall, and F1-score) for all three models and discuss how the input features influenced the results. (5 Marks)

4.4 Analysis and Logical Comparison (10 Marks)

1. Discuss the performance differences between the three models based on their input features. What kinds of features seem to be more helpful for classification? (5 Marks)

2. Explain how the choice of feature extraction impacts the MLP's ability to learn and generalize, and identify scenarios where each transformation might perform better. (5 Marks)

Good luck with the assignment!