Final Group Project

STAT7008A: Programming for Data Science (Fall 2025)

presentation due: **November 28, 2025, Friday, 11:59 PM**
Document Submission Due: **December 5, 2025, Friday, 11:59 PM**

Submission Notes

For questions on the topics, you are encouraged to post them on Moodle.

For individual questions, please contact:

- Mr. Peixiang Huang (u3011737@connect.hku.hk)
- Mr. Kailing Wang (wkl151@connect.hku.hk)
- Mr. Weizhi Zhong (u3013076@connect.hku.hk)

Requirements:

(1) Code:

- Source code to reproduce the project
- ReadMe file
- Any additional files if required by the topic

(2) Presentation slides

(3) Project report: A single PDF file

* Only one set of files needs to be submitted to Moodle, by the team leader on behalf of the whole team.

Grading:

(1) Code (15%)

- Organize the code in a well-structured way.
- Provide a clear README.txt to with detailed step-by-step instruction on how to run the program.
- Include clear and concise comments in the code.

(2) Presentation (20%)

- Clear and concise presentation with well-prepared slides.
- Each team member should participate in the presentation.

(3) Project Report (20%)

In the report (1500-3000 words), please include:

- Cover page: Project title and team members
- Section 1: Introduction – describe the project and your main idea to solve it.
- Section 2: Methodology – describe your program design and detailed methods, with some diagrams or figures.
- Section 3: Implementation details – summarize the details on how the program is implemented.
- Section 4: Experiments– evaluate your method(s) and analyze the results, both qualitatively and quantitatively, with tables, figures, and discussions.
- Section 5: Conclusion – summarize what you have achieved, discuss whether the program has any limitations, demonstrate possible further improvement. This is especially important for the submissions that do not fully meet the project objectives.
- Section 6: Reference – list of related and cited papers or other works in the project.
- Appendix: (1) a table to specify each individual's contribution to the report/code/presentation; (2) other useful information or details (if any).

Project Topics

Topic 1: Interactive Data Visualization with Streamlit

Introduction

Data visualization transforms raw data into meaningful visual insights that drive decision-making. Using Streamlit's Python framework (https://streamlit.io/), this project aims to create an interactive dashboard that enables users to explore complex datasets through dynamic visualizations. The goal is to develop a user-friendly interface that combines powerful analysis tools with intuitive interaction design.

Objectives

You are expected to implement:

(1) Interactive Dashboard Development

- Create diverse visualizations including scatter plots for correlation analysis, line charts for trend identification, and heatmaps for pattern detection, using the data you create or download from the Web.
- Implement filters (date ranges, categories, value ranges) and selectors (dropdown menus, sliders)
- Display key statistics (mean, median, variance) and data summaries
- Design data upload interface and apply the visualizations above.

(2) Interactive Feature Enhancement

- Create dynamic filters that update visualizations in real-time
- Add export functionality for charts and processed data
- Implement customizable parameters for visualization settings
- Design custom layouts with responsive themes that adapt to different screen sizes

(3) Advanced Feature Implementation

- Enable real-time data updates through local file monitoring.

- Handle multiple datasets with automatic format detection and preprocessing
- Build an AI chat (e.g., DeepSeek) interface for analyzing the data, using an API key.
- Implement comprehensive error handling for data loading, processing, and user inputs.

(4) Real-Time Data Analysis

- Connect to online, real-time web data (for example, through API connections) of any two different types from the following — stock market prices, weather data, or other types of real-world data — and analyze them using the system you designed above.
- Perform data cleaning: handle missing values, outliers, and inconsistent formats

* The Assigned Tutor: Mr. Weizhi Zhong (u3013076@connect.hku.hk)

Topic 2: Web Data Crawling and Analysis

Introduction

We are living in an era of information explosion. Whether it is news on the internet or newly released papers in certain research fields (e.g., AI), the amount of information updated every day is enormous. Reading through all of it would consume an excessive amount of time and energy. Therefore, learning how to filter, extract, and identify key information from such vast data sources has become increasingly important. In this project, we will take as an example the Computer Vision papers updated on a specific day on the arXiv website (https://arxiv.org/catchup/cs.CV/2025-09-30?abs=True), and perform web data crawling, analysis, and visualization based on that website.

Objectives

You are expected to implement:

(1) Crawl key information of papers from the webpage.

- Crawl all papers listed under "New submissions" and "Cross submissions" on the page.
- For each paper, crawl its title and abstract information.
- For each paper, go into its "html" link and crawl the author list and author affiliations (if available) information.

(2) Analyze the crawled paper data using AI techniques.

- For each paper, call a large language model (LLM, e.g., DeepSeek) to analyze its title and abstract and generate a concise task description (e.g., "video synthesis") for it. You may use a locally deployed LLM (if GPU permits) or invoke an LLM via API.
- Classify or cluster all papers based on task descriptions of papers, grouping papers with similar tasks together.
- Export the crawled and analyzed results to an Excel file.

(3) GUI design and data Visualization

- Build an interactive graphical user interface (e.g., with Streamlit) to run the crawling and analysis workflow above.
- Allow users to specify arXiv websites of different dates, and display a progress bar showing the current crawling progress and estimated remaining time.
- List all crawled paper titles on the UI; when a title is clicked, show the corresponding abstract and author/affiliation information.
- Visualize the paper classification/clustering results on the UI using an interactive graph/figure. For example, when hovering over a node representing a paper, display the paper's details (e.g., title), or even open the corresponding paper webpage by clicking.

* The Assigned Tutor: Mr. Weizhi Zhong (u3013076@connect.hku.hk)

Topic 3: Intelligent Web Image Drive with Flask

Introduction

This project asks you to build a production-leaning Flask web application that functions as a *pure image drive*. The system supports *secure image upload*, browsing, and downloading; *semantic search* via CLIP-like embeddings; *OCR-based search* over text contained in images; and *image-to-image similarity* retrieval. Image embeddings must be computed once on ingest and *persisted as metadata* for indexing—**never re-embedded on each search**. A small, server-side base dataset with at least **10 major categories** (e.g., flowers, food, cars, dogs, road signs, buildings, landscapes, birds, electronics, people. You can select them from open datasets like ImageNet series, OmniBenchMark. Note that since you'll encode these images using CLIP-like models, there's no need for the images to be labeled) should be curated and stored on the backend as example user data. Basic statistics and visualizations over this taxonomy are required.

Objectives

You are expected to implement:

(1) Data Curation and Indexing Plan:

- Curate a base image database with $\geq$**10 categories**; define a simple, common taxonomy and tag images automatically.
- Implement an **embedding pipeline** (e.g., CLIP) that computes vectors once per image at ingest; store vectors as metadata (DB table, vector store, or sidecar files).
- Build and maintain a **vector index** (e.g., FAISS/pgvector) for nearest-neighbor queries; maintain a **text index** (e.g., SQL FTS) for OCR text.
- Implement an **OCR pipeline** (e.g., Tesseract/PaddleOCR); persist extracted text per image to enable keyword/substring search.
- Provide minimal dataset hygiene (dedup checksums).

(2) System Implementation and Web Interface:

- Develop a local web application using Flask.
- Support the following user-facing functions:
  - **User Auth**: secure registration/login, password hashing, token-based sessions (expiry/refresh).
  - **Upload/Browse/Download**: drag&drop upload, gallery view (grid/list), detail view, secure downloading.
  - **Text Search**: encode query text with the text encoder and retrieve semantically similar images via the vector index; full-text/substring search over OCR'd text.
  - **Image-to-Image Similarity**: upload a query image, return nearest neighbors in embedding space.
  - **Similar Panel on Detail**: when viewing one image, show a side panel of similar images.
  - **Analytics View**: show category distribution, upload trends, size distributions; **Download Analysis** (CSV/JSON).
  - **Logging system**: structured logs of different levels for uploads, downloads, searches, and errors; rotation and basic audit trails. Just logging as files is ok.
- Expose clean endpoints for upload, listing, detail, search (semantic/OCR/similar), analytics summary, and analysis export.

(3) Analysis (For Report Writting):

- Evaluate retrieval functionality with small-scale diagnostics (e.g., `top-`k examples, latency snapshots, index size vs. memory trade-offs) and discuss limitations and future improvements.

Notes. If you wish to map the service to public Internet via `frp`, please contact the TAs for guidance.

* The Assigned Tutor: Mr. Kailing Wang (wkl151@connect.hku.hk)

Topic 4: Local VLLM + Playwright Web Agent

Introduction

This project challenges you to build an autonomous web agent that executes medium-difficulty, natural-language instructions by integrating a **local Vision-Language Model (VLLM)** with the **Python Playwright** library. The agent must operate in a multi-round loop: the VLLM analyzes the webpage (via a screenshot and a compact DOM representation), plans the next action, and instructs Playwright to execute it. This cycle repeats until the user's goal is met, the agent self-terminates, or a timeout is reached. The system should maintain its state, allowing users to issue follow-up instructions within the same session.

Example Task

A representative task for this agent would be:

*"Find the **most recent** technical report (PDF) about* Qwen*, then **interpret Figure 1** by describing its purpose and key findings."*

Objectives

You are expected to implement:

(1) Core Agent Framework:

- Implement the primary control loop where the local VLLM plans actions and Playwright executes them.
- Ensure that after each action, a new screenshot and a simplified DOM summary are passed back to the model for continuous, stateful decision-making.

(2) Pure Python Toolchain Implementation:

- Develop a set of tools that the VLLM can call to interact with the browser and the local file system. All tools must be implemented in Python, without relying on external runtimes like MCP.
- The required toolset includes:
  - **Browser Control:** `goto`, `click`, `type`, `press`, `wait`
  - **Information Gathering:** `screenshot`, `dom_summary`
  - **File Operations:** `download_pdf`, `pdf_extract_text`, `pdf_extract_images`, `save_image`, `write_text`
  - **(Optional)** `ocr`

(3) Deliverables and Analysis:

- **A Working System:** A functional agent that accepts a natural language goal, executes the task through multiple steps, and allows for follow-up interaction.
- **Execution Artifacts:** A collection of output files from a sample run, including execution logs, step-by-step screenshots, the final downloaded PDF, any saved image crops, and the final interpretation saved to a `.txt` file.

- **Analysis Report:** A brief report detailing any failure cases encountered and their causes. If multiple models were tested, include a performance comparison.

Notes.

- **Hardware Guidance:** If you do not have access to a high-end GPU (e.g., NVIDIA RTX 3090 or an Apple Silicon Mac with more than 20 GB of unified memory), it is highly recommended to use a VLLM with fewer than 10 billion parameters.
- **Locale Alignment:** To maximize model performance, the browser's display language should match the primary language of the VLLM (e.g., use an English UI for Gemma, a Chinese UI for Qwen).
- **Fallback Option:** If local models perform poorly, you may use an external model API as a fallback. A **comparative analysis** of different models' capabilities is considered a significant contribution.
- **Design References:** You may take inspiration from MCP-style tool design, but you must implement everything in Python. The Sentra-Auto-Browser repository is a useful reference.

* The Assigned Tutor: Mr. Kailing Wang (wkl151@connect.hku.hk)

Topic 5: 3D Shape Analysis

Introduction

When working with 3D objects, it is indeed true that there are various representations that can be used to describe and analyze them. Each representation offers distinct advantages and is suitable for different applications. Let's explore these representations further:

- Point Cloud: A point cloud consists of a set of 3D coordinates (points) in space. Each point represents a specific location on the surface of an object. Point clouds can be generated through techniques like 3D scanning or LiDAR. Point clouds are useful for capturing the shape and structure of objects and are commonly used in applications like 3D modeling, computer graphics, and virtual reality.
- Mesh: A mesh is a collection of vertices, edges, and faces that creates a polygonal representation of a 3D object. It is the most used representation for 3D models in computer graphics and simulation. Meshes provide a surface-based representation, allowing for detailed rendering and visualization. They can be used for applications like gaming, simulation, and animation.
- Voxel: Voxel stands for volumetric pixel and represents a 3D grid of uniformly sized cubes called voxels. Each voxel can store attributes like color, density, or material properties. Voxel-based representations are commonly used in medical imaging, scientific visualization, and virtual reality. They allow for efficient storage and manipulation of volumetric data.
- Multi-view: Multi-view representations involve capturing an object from multiple viewpoints or cameras. By combining these views, a more complete and detailed representation of the object can be obtained. Multi-view representations are often used in applications like 3D reconstruction, object recognition, and augmented reality.

It is important to note that these representations are not mutually exclusive, and they can be converted into each other depending on the requirements of a particular task.

Objectives

In this project, we will focus on two widely used 3D shape analysis datasets: ModelNet-10 and Modelnet-40. You are expected to:

- Apply different 3D shape classification algorithms to these datasets based on each of the different representations mentioned above.

- Compare the performance of different 3D shape classification algorithms across the different representations (point cloud, mesh, voxel, and multi-view). Analyze factors such as accuracy, computational efficiency, and robustness to variations in data.
- Explore different feature extraction techniques for each representation. Investigate the feature extraction methods unique to point cloud, mesh, voxel, and multi-view representations for 3D shape analysis, which can include both hand-designed features and learned features using deep learning methods.

* The Assigned Tutor: Mr. Peixiang Huang (u3011737@connect.hku.hk)

Topic 6: A Multi-Agent System for Chest X-ray Analysis

Introduction

Chest X-rays (CXRs) are one of the most common and critical diagnostic imaging tools in modern medicine. The recent advancement in artificial intelligence has led to the development of highly specialized models that excel at individual tasks such as disease classification, organ segmentation, and automated report generation. However, these models often operate in isolation, creating a fragmented ecosystem that limits their utility in complex clinical workflows that require multi-step reasoning.

This project challenges you to bridge this gap by designing and building a prototype of a unified system for intelligent CXR analysis. The core idea is to create a multi-agent framework that can leverage a collection of specialized AI models as tools. When presented with a user's request, this system should be able to understand the goal, select the appropriate tool (or sequence of tools), execute, and synthesize the results into a coherent response. This approach moves beyond single-task models towards a more dynamic, flexible, and powerful paradigm for medical image interpretation.

Objectives

You are expected to implement:

(1) Foundational Research:

- Conduct a literature review on AI models for common CXR analysis tasks. You should investigate and summarize methods for **at least two** of the following: disease classification, abnormality localization (e.g., segmentation or object detection), report generation and visual question answering (VQA).
- Perform a focused analysis of different methods for automated radiology report generation:
  - Supervised training from scratch: RGRG [1], CMCRL [2] ...
  - Foundation models inference: Chexagent [3], MAIRA [4], Medgemma [5], Lingshu [6] ...

  For these two distinct methodological approaches, please **select one specific method from each category** for a comparative analysis. You can directly use the provided model parameters from its official repo to inference. The evaluation should include:

  - Common report generation metrics (**at least 4 different metrics**, referencing ReXrank)
  - Inference efficiency
  - Computational resource consumption during inference
  - Your perspective on the strengths and limitations of each methodological paradigm

  The evaluation should be conducted on a **randomly sampled 80% subset** of the chosen dataset, using a fixed random seed of 42 to ensure reproducibility.

(2) Tool Curation and Web Interface:

- Develop a local web application (e.g., using Streamlit, Flask ...) that serves as the user interface for your system. The application should be user-friendly and allow for easy interaction.
- The interface must allow a user to upload a CXR image and perform **at least two** of the following functions for CXR analysis (You may use pre-trained models for this purpose). Teams can choose which tools to implement based on their capabilities:
  - Interactive Classification: Run a classification model on the uploaded image and detect the presence of 14 observations defined by CheXpert [7]) with their corresponding confidence scores.
  - Visual Grounding: Visualize the location of a key finding on the X-ray, for example, by drawing a bounding box around a nodule or creating a segmentation mask over an area of consolidation.
  - Automated Report Generation: Use one of the report generation models from your survey to produce a draft radiological report (e.g., a "Findings" section) for the image.
  - Visual Question Answering (VQA): Implement a simple interface where a user can ask a question in natural language about the image (e.g., "Is the heart enlarged?") and receive a text-based answer.

(3) System Architecture Design: You may choose one of the following two approaches based on your group's technical proficiency to design the system architecture:

- Option A (Standard: The Modular Approach): Design a user interface with separate, clearly defined controls for each function. For example, have distinct buttons for "Classify Image", "Generate Report", or "Localize Abnormality". In this approach, the system's logic is a straightforward mapping of user input to a specific tool, without a central reasoning agent.
- Option B (Advanced: The Agentic Approach): Design a system where a central agent (e.g., powered by an LLM like GPT through an API) interprets a user's free-text query, decides which tool(s) to use, executes them, and synthesizes the results. For example, a query like "Diagnose the main disease in this X-ray and show me where it is" would cause the agent to first call the classification tool and then the localization tool.

(4) Analysis and Documentation:

- Prepare a final report covering: (1) literature review findings, (2) analysis of methods for automated radiology report generation, (3) implementation of system architecture and web interface.
- **Submit a complete demo**, we will **focus on system completeness, not performance**. Feel free to use small or quantized models (e.g., medgemma-4b-it-gguf), and CPU execution is fine if resources are limited.

Suggested Datasets. You may use MIMIC-CXR (a large-scale dataset containing hundreds of thousands of chest X-ray images paired with free-text radiology reports) for your project. Please be mindful of the access requirements and usage agreements for this dataset.

* The Assigned Tutor: Mr. Peixiang Huang (u3011737@connect.hku.hk)

References

[1] T. Tanida, P. Müller, G. Kaissis, and D. Rueckert, "Interactive and explainable region-guided radiology report generation," in *CVPR*, 2023.

[2] W. Chen, Y. Liu, C. Wang, J. Zhu, G. Li, C.-L. Liu, and L. Lin, "Cross-modal causal representation learning for radiology report generation," *IEEE Transactions on Image Processing*, vol. 34, pp. 2970–2985, 2025.

[3] Z. Chen, M. Varma, J.-B. Delbrouck, M. Paschali, L. Blankemeier, D. V. Veen, J. M. J. Valanarasu, A. Youssef, J. P. Cohen, E. P. Reis, E. B. Tsai, A. Johnston, C. Olsen, T. M. Abraham, S. Gatidis, A. S. Chaudhari, and C. Langlotz, "Chexagent: Towards a foundation model for chest x-ray interpretation," *arXiv preprint arXiv:2401.12208*, 2024. [Online]. Available: https://arxiv.org/abs/2401.12208

[4] S. Bannur, K. Bouzid, D. C. Castro, A. Schwaighofer, A. Thieme, S. Bond-Taylor, M. Ilse, F. Pérez-García, V. Salvatelli, H. Sharma, F. Meissen, M. P. Ranjit, S. Srivastav, J. Gong, N. C. F. Codella, F. Falck, O. Oktay, M. P. Lungren, M. T. A. Wetscherek, J. Alvarez-Valle, and S. L. Hyland, "Maira-2: Grounded radiology report generation," *arXiv*, vol. abs/2406.04449, 2024. [Online]. Available: https://arxiv.org/abs/2406.04449

[5] A. Sellergren, S. Kazemzadeh, T. Jaroensri, A. Kiraly, M. Traverse, T. Kohlberger, S. Xu, F. Jamil, C. Hughes, C. Lau, J. Chen, F. Mahvar, L. Yatziv, T. Chen, B. Sterling, S. A. Baby, S. M. Baby, J. Lai, S. Schmidgall, L. Yang, K. Chen, P. Bjornsson, S. Reddy, R. Brush, K. Philbrick, M. Asiedu, I. Mezerreg, H. Hu, H. Yang, R. Tiwari, S. Jansen, P. Singh, Y. Liu, S. Azizi, A. Kamath, J. Ferret, S. Pathak, N. Vieillard, R. Merhej, S. Perrin, T. Matejovicova, A. Ramé, M. Riviere, L. Rouillard, T. Mesnard, G. Cideron, J. bastien Grill, S. Ramos, E. Yvinec, M. Casbon, E. Buchatskaya, J.-B. Alayrac, D. Lepikhin, V. Feinberg, S. Borgeaud, A. Andreev, C. Hardin, R. Dadashi, L. Hussenot, A. Joulin, O. Bachem, Y. Matias, K. Chou, A. Hassidim, K. Goel, C. Farabet, J. Barral, T. Warkentin, J. Shlens, D. Fleet, V. Cotruta, O. Sanseviero, G. Martins, P. Kirk, A. Rao, S. Shetty, D. F. Steiner, C. Kirmizibayrak, R. Pilgrim, D. Golden, and L. Yang, "Medgemma technical report," 2025. [Online]. Available: https://arxiv.org/abs/2507.05201

[6] L. Team, W. Xu, H. P. Chan, L. Li, M. Aljunied, R. Yuan, J. Wang, C. Xiao, G. Chen, C. Liu, Z. Li, Y. Sun, J. Shen, C. Wang, J. Tan, D. Zhao, T. Xu, H. Zhang, and Y. Rong, "Lingshu: A generalist foundation model for unified multimodal medical understanding and reasoning," 2025. [Online]. Available: https://arxiv.org/abs/2506.07044

[7] J. Irvin, P. Rajpurkar, M. Ko, Y. Yu, S. Ciurea-Ilcus, C. Chute, H. Marklund, B. Haghgoo, R. Ball, K. Shpanskaya, J. Seekins, D. A. Mong, S. S. Halabi, J. K. Sandberg, R. Jones, D. B. Larson, C. P. Langlotz, B. N. Patel, M. P. Lungren, and A. Y. Ng, "Chexpert: A large chest radiograph dataset with uncertainty labels and expert comparison," 2019. [Online]. Available: https://arxiv.org/abs/1901.07031