7-Day Postdoctoral Technical Challenge

AI Medical Imaging, Visual Language Models, and Semantic Retrieval

AlfaisalX: Cognitive Robotics and Autonomous Agents
MedX Research Unit, Medical Robotics & AI in Healthcare
College of Engineering and Advanced Computing
Alfaisal University, Riyadh, Saudi Arabia

Challenge Overview

This technical challenge is designed to evaluate your ability to build end-to-end AI systems for medical imaging applications. You will work on three interconnected tasks that demonstrate your skills in deep learning, large language models, and information retrieval. The challenge reflects real-world research scenarios where multiple AI components must work together to solve complex problems.

Submission Requirements:

- GitHub repository with complete Python implementation
- Google Colab notebook for online testing and demonstration

Duration: 7 days **Dataset:** MedMNIST v2 – PneumoniaMNIST

1 Introduction

Medical AI systems increasingly require the integration of multiple components: image classification models for diagnosis, language models for report generation, and retrieval systems for finding similar cases. This challenge asks you to build a prototype system that combines all three capabilities.

You will use the PneumoniaMNIST dataset, a subset of the MedMNIST benchmark containing chest X-ray images for pneumonia classification. This dataset is lightweight (approximately 6,000 training images of 28x28 pixels) and can be trained on standard hardware without GPU requirements, making it ideal for a time-constrained challenge.

The challenge consists of three tasks of increasing complexity. Each task builds upon foundational AI/ML skills while introducing modern techniques such as visual language models and semantic search. You are expected to demonstrate not only technical implementation skills but also research thinking, including proper evaluation, error analysis, and critical interpretation of results.

2 Dataset

You must use the **PneumoniaMNIST** dataset from MedMNIST v2 for all tasks. This dataset provides a standardized benchmark for chest X-ray classification with the following characteristics:

- **Task:** Binary classification (normal vs. pneumonia)
- **Training set:** Approximately 4,700 images
- **Validation set:** Approximately 500 images
- **Test set:** Approximately 600 images

- **Image format:** 28x28 grayscale images

The dataset can be installed and loaded using the official Python package:

```
pip install medmnist
```

For detailed documentation and usage examples, refer to the official website: https://medmnist.com/

3 Challenge Tasks

This challenge comprises three tasks. Each task is designed to evaluate different aspects of your technical capabilities. While we encourage you to attempt all three tasks, we understand that time constraints may limit what you can accomplish. Focus on delivering quality work rather than rushing through all tasks with incomplete implementations.

Task 1: CNN Classification with Comprehensive Analysis

Objective: Build a convolutional neural network (CNN) classifier for pneumonia detection and perform a thorough analysis of its performance.
This task evaluates your fundamental deep learning skills. You are expected to implement a complete machine learning pipeline from data preprocessing to model evaluation. The emphasis is not only on achieving high accuracy but also on demonstrating rigorous experimental methodology.
Requirements:
Your implementation must include a complete data pipeline with appropriate preprocessing steps. Consider normalization strategies suitable for medical images and implement data augmentation techniques that are meaningful for chest X-ray analysis. Your pipeline should handle train/validation/test splits properly and support batch processing for efficient training.
For the model architecture, you may choose any CNN-based approach, from simple custom architectures to established models like ResNet, EfficientNet or Vision Transformer (Recommended). Document your architectural choices and justify them based on the dataset characteristics. Implement proper training procedures including loss function selection, optimizer configuration, and learning rate scheduling.
The evaluation component is critical. You must report standard classification metrics including accuracy, precision, recall, F1-score, and Area Under the ROC Curve (AUC). Generate and analyze the confusion matrix to understand the types of errors your model makes. Identify and visualize failure cases—images that your model misclassifies—and provide analysis of why these errors might occur.
Deliverables:

- Training script with configurable hyperparameters
- Evaluation script that generates all required metrics
- Saved model weights (or instructions to reproduce training)
- Visualizations: training curves, confusion matrix, ROC curve, sample predictions
- **Markdown report:** `task1_classification_report.md` containing:
  - Model architecture description and justification
  - Training methodology and hyperparameters
  - Complete evaluation metrics with visualizations
  - Failure case analysis with example images
  - Discussion of model strengths and limitations

Task 2: Medical Report Generation using Visual Language Model

Objective: Use an open-source visual language model to generate medical reports from chest X-ray images.

This task evaluates your ability to work with state-of-the-art multimodal AI models. Visual language models (VLMs) can analyze images and generate natural language descriptions, making them valuable for automated medical report generation. You will integrate an existing pre-trained model into your pipeline rather than training one from scratch.

Recommended Model:

We recommend using **MedGemma**, Google's open-source medical visual language model available on Hugging Face. MedGemma is specifically designed for medical imaging tasks and can generate clinically relevant descriptions of medical images. Alternative models such as LLaVA-Med or other medical VLMs are also acceptable if you can justify your choice.

Requirements:

Implement a pipeline that takes a chest X-ray image as input and produces a natural language report describing the findings. Your implementation should handle the model loading, image preprocessing according to the model's requirements, and text generation with appropriate parameters.

Design prompts that guide the model to produce useful medical observations. Experiment with different prompting strategies and document their effects on output quality. Consider aspects such as prompt structure, level of detail requested, and any domain-specific instructions that improve results.

Evaluate the generated reports qualitatively. Select a representative sample of images (including both normal and pneumonia cases, as well as images your CNN misclassified) and analyze the quality of the generated reports. Discuss whether the VLM's observations align with the ground truth labels and your CNN's predictions.

Deliverables:

- Report generation script that processes images and outputs text reports
- Documentation of prompting strategies tested
- Sample generated reports for at least 10 images (mix of classes and difficulty levels)
- **Markdown report:** `task2_report_generation.md` containing:
  - Model selection justification (why MedGemma or alternative)
  - Prompting strategies tested and their effectiveness
  - Sample generated reports with corresponding images
  - Qualitative analysis comparing VLM outputs with ground truth and CNN predictions
  - Discussion of the model's strengths and limitations for this task

Task 3: Semantic Image Retrieval System

Objective: Build a content-based image retrieval (CBIR) system using medical image embeddings and a vector database.

This task evaluates your ability to work with embedding models and vector search systems, which are fundamental components of modern AI applications. You will create a system that can find similar medical images based on visual content, enabling use cases such as case-based reasoning and similar case retrieval for clinical decision support.

Requirements:

Select an appropriate embedding model for medical images. You may use the image encoder from a medical vision-language model (such as BioViL-T, MedCLIP, or PMC-CLIP) or fine-tune a general-purpose model. The key requirement is that the embeddings should capture medically relevant visual features.

Build a vector index containing embeddings for all images in the test set. Use a vector database or similarity search library such as FAISS, ChromaDB, or Pinecone. Your system should support efficient nearest-neighbor search to retrieve the most similar images to a query.

Implement two retrieval interfaces:

- **Image-to-image search:** Given a query image, retrieve the k most similar images from the database.
- **Text-to-image search (if supported by your embedding model):** Given a text description, retrieve relevant images.

Evaluate retrieval quality using appropriate metrics. At minimum, compute Precision@k: for each query image, measure what fraction of the top-k retrieved images share the same label as the query. This indicates whether the retrieval system groups similar cases together.

Deliverables:

- Embedding extraction script
- Vector index construction and storage
- Search scripts with command-line interfaces for both retrieval modes
- Visualization of retrieval results (query image alongside top-k results)
- **Markdown report:** `task3_retrieval_system.md` containing:
  - Embedding model selection and justification
  - Vector database implementation details
  - Retrieval system architecture and usage instructions
  - Quantitative evaluation with Precision@k metrics
  - Visualization of retrieval results with analysis
  - Discussion of retrieval quality and failure cases

4 Development Approach

4.1 Use of AI Coding Assistants

The use of AI coding assistants and vibe coding tools is **allowed and recommended** for this challenge. Modern AI development increasingly involves collaboration with AI tools, and we want to evaluate how effectively you can leverage these resources.

You may use tools such as GitHub Copilot, ChatGPT, Claude, Cursor, or similar assistants to help with code generation, debugging, and problem-solving. However, you must understand the code you submit and be prepared to explain your implementation decisions in a follow-up interview. The goal is to demonstrate your ability to direct AI tools effectively toward solving complex problems, not to test whether you can write every line of code from memory.

4.2 Submission Requirements

Your submission must include two components:

GitHub Repository: Create a public GitHub repository containing your complete implementation. The repository should be well-organized with clear directory structure, comprehensive documentation, and all necessary files to reproduce your results. Include a detailed README that explains how to set up the environment, run each component, and interpret the outputs.

Google Colab Notebook: Provide a Colab notebook that demonstrates your system working end-to-end. This notebook should allow reviewers to run your code online without local setup. Include clear markdown cells explaining each section, and ensure the notebook can execute successfully on Colab's free tier (with or without GPU, depending on your implementation).

4.3 Repository Structure

Organize your code with clear separation of concerns. A suggested structure is:

```
repository/
|-- data/              # Data loading and preprocessing utilities
|-- models/            # Model architectures and saved weights
|-- task1_classification/   # CNN classifier implementation
|-- task2_report_generation/  # VLM report generation
|-- task3_retrieval/   # Semantic search system
|-- notebooks/         # Colab notebook(s)
|-- reports/           # Generated outputs and analysis
|-- requirements.txt   # Python dependencies
`-- README.md          # Comprehensive documentation
```

5 Evaluation Criteria

Your submission will be evaluated holistically based on the following criteria:

Criterion	Weight	What We Look For
Technical Implementation	35%	Code quality, correct implementation of algorithms, appropriate use of libraries and frameworks
Experimental Rigor	25%	Proper evaluation methodology, meaningful metrics, thorough error analysis
Research Thinking	20%	Critical analysis of results, identification of limitations, thoughtful discussion of improvements
Documentation & Reproducibility	20%	Clear README, working Colab notebook, well-commented code, ease of reproduction

6 Important Notes

6.1 Partial Completion

We recognize that completing all three tasks to a high standard within seven days is ambitious. **It is perfectly acceptable if you cannot complete all tasks.** We are more interested in seeing how you manage the challenge and the quality of work you deliver than in checking boxes.

If you find yourself running short on time, prioritize depth over breadth. A well-implemented Task 1 with thorough analysis is more valuable than superficial implementations of all three tasks. Document what you accomplished, what challenges you encountered, and what you

would do differently with more time.

Your approach to managing the challenge—how you prioritize, how you handle obstacles, and how you communicate your progress—tells us as much about your potential as a researcher as the technical output itself.

6.2 Technical Considerations

- **Computational resources:** The challenge is designed to be completable on standard hardware. Task 1 can be done on CPU. Tasks 2 and 3 may benefit from GPU access, which is available through Google Colab's free tier.
- **API keys:** If you use commercial APIs (e.g., for LLM access), follow security best practices. Use environment variables, never commit API keys to your repository, and document what credentials are needed.
- **Model weights:** For large model weights that cannot be stored in GitHub, provide clear instructions for downloading them or use Hugging Face model hub references.

6.3 Questions

If you have clarifying questions about the requirements, please reach out. We want to evaluate your technical skills, not your ability to guess our intentions.

Contact:

- Prof. Anis Koubaa, `akoubaa@alfaisal.edu`
- Dr. Mohamed Bahloul, `mbahloul@alfaisal.edu`

7 Submission Deadline

Submit your GitHub repository URL and confirm that your Colab notebook is accessible within **7 days** of receiving this challenge, by **February 22, 2026**. Ensure all links are working and the repository is public (or shared with the reviewers if private).

Submission Form: https://docs.google.com/forms/d/1s9NHk9yt3c5xxSS83stFr8O8ZEYoLR8X_PBsY1-CBtI/edit

Good luck! We look forward to reviewing your submission and learning about your approach to this challenge. Remember: we value quality, clarity, and thoughtful analysis over rushed completeness.