5-Day Postdoctoral Technical Challenge

AI Medical Imaging + Agentic LLM Pipeline

Challenge Overview

This challenge evaluates your ability to design and implement an end-to-end AI system combining medical imaging, machine learning, and agentic LLM capabilities. You will build a complete pipeline from data processing to automated analysis and reporting.

Submission: GitHub repository with complete implementation
Duration: 5 days
Dataset: MedMNIST v2 – PneumoniaMNIST (provided)

1 Dataset Specification

You must use the **PneumoniaMNIST** dataset from MedMNIST v2:

- **Task:** Binary classification of chest X-ray images for pneumonia detection
- **Size:** Approximately 6,000 training images
- **Format:** 2D grayscale images (28×28 pixels)
- **Access:** Publicly available via Python package

Installation:

```
pip install medmnist
```

Documentation: https://medmnist.com/

2 Challenge Objectives

Build a complete system that accomplishes the following:

1. Train a deep learning model for pneumonia classification
2. Evaluate model performance using rigorous metrics
3. Implement an LLM-based agent to analyze experimental results
4. Generate an automated experiment report with insights and recommendations

3 Required Components

3.1 1. Data Pipeline

Implement a complete data processing pipeline including:

- Data loading and exploration
- Normalization and preprocessing
- Data augmentation strategies
- Train/validation/test split management
- Batch processing for training and inference

3.2 2. Vision Model

Train a deep learning model for pneumonia classification. You may choose any architecture (CNN, ResNet, Vision Transformer, or custom design).

Required outputs:

- Classification accuracy
- Area Under the ROC Curve (AUC)
- Confusion matrix
- Analysis of failure cases with examples

3.3 3. Agentic LLM Component

Design and implement an LLM-based agent that can analyze experimental results and provide insights.

Agent inputs:

- Training metrics and loss curves
- Evaluation results and performance statistics
- Error analysis data

Agent outputs:

- Performance explanation and interpretation
- Identification of model weaknesses
- Concrete suggestions for improvements
- Comprehensive experiment summary

Implementation notes:

- The agent may use tools such as metrics readers, dataset inspectors, or result summarizers
- You may use any framework (LangChain, DSPy, custom implementation)
- Agent should demonstrate reasoning capabilities beyond simple template filling

3.4 4. Automated Experiment Report

Your system must automatically generate a comprehensive report including:

- Summary of all metrics
- Visualization of sample predictions
- Error analysis with representative examples
- Agent-generated insights and recommendations
- Suggested next steps for model improvement

Output format: Markdown or PDF

3.5 5. Reproducibility Requirements

Your repository must enable complete reproducibility:

- Clear README with setup instructions
- Complete requirements file with all dependencies
- Simple commands to run training and evaluation
- Configuration files for hyperparameters
- Seed management for deterministic results

4 Expected Repository Structure

Organize your code with clear separation of concerns:

```
repository/
|-- data/              # Data loading and preprocessing
|-- models/            # Model architectures
|-- training/          # Training scripts and utilities
|-- evaluation/        # Evaluation and metrics
|-- agent/             # LLM agent implementation
|-- reports/           # Generated reports and outputs
|-- configs/           # Configuration files
|-- requirements.txt   # Python dependencies
‘-- README.md          # Documentation
```

5 Evaluation Criteria

Your submission will be evaluated according to the following rubric:

Area	**Weight**	**Key Aspects**
Pipeline & Model Quality	40%	Code organization, model design, training methodology, performance
Evaluation Rigor	25%	Comprehensive metrics, error analysis, statistical validity
Agent Usefulness	20%	Quality of insights, reasoning depth, actionable recommendations
Code & Reproducibility	15%	Documentation, code quality, ease of reproduction

6 Bonus Components (Optional)

Exceptional candidates may include additional features for extra credit:

- **Novel augmentation strategies** tailored to medical imaging
- **Model improvements** through architecture search or ensemble methods
- **Uncertainty estimation** with confidence calibration
- **Training optimization** using advanced techniques
- **Ablation studies** demonstrating component contributions
- **Medical visual embeddings + image retrieval:**
  - Use a pre-trained medical vision model (e.g., BioViL-T, MedCLIP, PMC-CLIP)
  - Build a content-based image retrieval (CBIR) system using FAISS
  - Implement image-to-image and text-to-image search
  - Evaluate retrieval quality with Precision@k metrics
  - Integrate retrieval capabilities with the LLM agent

7 Submission Guidelines

7.1 What to Submit

1. **GitHub repository URL** with complete implementation
2. **README.md** with:
   - Setup instructions
   - Commands to reproduce all results
   - Brief description of your approach
   - Summary of key findings
3. **Generated report** (in reports/ directory)
4. **Trained model weights** (or instructions to reproduce)

7.2 Minimum Viable Submission

At minimum, your repository must:

- Load and preprocess the PneumoniaMNIST dataset
- Train a model that achieves reasonable performance (¿70% accuracy)
- Generate all required metrics and visualizations
- Include a functional LLM agent that produces meaningful analysis
- Produce an automated report
- Be reproducible with clear documentation

8 Technical Requirements

- **Programming language:** Python 3.8+
- **Deep learning framework:** PyTorch or TensorFlow
- **LLM access:** You may use any LLM API (OpenAI, Anthropic, open-source models)
- **Version control:** Git with meaningful commit history
- **Documentation:** Clear comments and docstrings

9 Suggested Timeline

While you may organize your time as you see fit, a typical approach might be:

- **Days 1–2:** Data pipeline and baseline model implementation
- **Day 3:** Comprehensive evaluation and model improvements
- **Day 4:** Agent integration and testing
- **Day 5:** Report generation, documentation, and final cleanup

10 Evaluation Process

Your submission will be reviewed for:

1. **Technical competence:** Can you build robust ML pipelines?
2. **Research thinking:** Do you evaluate thoroughly and think critically?
3. **Modern AI integration:** Can you effectively use LLM-based tools?
4. **Code quality:** Is your code clean, documented, and maintainable?
5. **Problem-solving:** How do you approach challenges and debugging?

11 Important Notes

- **Originality:** Your code must be your own work. You may use libraries and reference documentation, but the implementation should demonstrate your understanding.
- **API keys:** If using commercial LLM APIs, ensure you follow best practices for API key management (environment variables, not hardcoded).
- **Computational resources:** The challenge is designed to be completable on a standard laptop. GPU access is helpful but not required.
- **Questions:** If you have clarifying questions about requirements, please reach out.

Good luck! We look forward to reviewing your submission and learning about your approach to this challenge.