VisiHealth AI

Making Medical AI Human-Friendly

Supervisor:
Sir Abdul Rahman

Submitted by:
Junaid Mohi Ud Din (01-134222-071)
Hammad Ur Rehman (01-134222-059)

Department of Computer Science Bahria University, Islamabad

Chapter 1: Introduction

1. Introduction

1.1 Project Overview

VisiHealth AI is a multimodal Medical Visual Question Answering (Med-VQA) which aims to bring artificial intelligence and clinical trust closer to each other. The system combines computer vision and natural language processing features to not only provide answers to clinical questions using medical images but it also includes giving rationales based on medical knowledge. The system is able to generate verifiable explanations by decoding visual features of radiology images and interpreting textual clinical queries, thereby resolving the problem of black box nature of existing medical AI models.

The classic Med-VQA systems are usually aimed at giving straight forward answers without any evidence that the medical practitioners cannot rely on in such an emergency. To address them, this project makes use of the SLAKE dataset, which is compiled of radiology photographs, semantic segmentation masks, and structured medical knowledge graph. The VisiHealth AI uses a CNN that has been trained fresh to extract image features and localize regions of interest (ROI) as well as a fine-tuned a pre-trained BioBERT model.

The solution suggested combines these textual and visual inputs to deduce clinical responses and at the same time extracts pertinent information in a medical knowledge graph. The outcome of this is the production of a system giving a prediction and a chain of reasons, which focuses on interpretability and clinical relevance. Essentially, VisiHealth AI is a major advance in Explainable AI (XAI) in healthcare, and this will form a basis to AI-enforced medical diagnosis and education.

1.2 Problem Description

The issue of AI systems interpretability is mandible in the healthcare sector as much as it is the case in terms of accuracy. Although currently Med-VQA studies have had immense performance in tasks with data sets such as VQA-RAD or VQA-Med, these networks are primarily viewed as a black box. The yes without giving the answer, which poses an obstacle

to apply it in a medical practice where evidence is needed.

The fundamental issues that can be concluded upon are:

Lack of Explainability: The state of the art Med-VQA models give answers, but do not provide rationales or evidence to justify their answer.

Trust Factor: An AI system that only delivers the answer of Abnormality detected will never be trusted by medical professionals who do not indicate the particular image or medical fact, which resulted in such response.

Limitations of Data Sets: It is notable that lots of data sets do not have the semantic data or knowledge graph triplets to train models based on reasoning.

Lack of Context: Typical models do not consider any of the underlying medical expertise needed to relate visual symptoms with diagnostic terms.

VisiHealth AI can solve these problems by introducing a reasoning aware architecture. In comparison to ordinary classifiers, the system bases its responses on particular image regions (ROIs) and confirmed medical facts, and as a result of this, the decision making process of the AI is not only transparent, verifiable, but also clinically significant.

1.3 Project Objective

The main purpose of the proposed project is to create and introduce a multimodal Med-VQA system enhancing trust in AI-supported healthcare because it provides both correct responses and readable answers. The following are the key objectives:

i. To create a hybrid Med-VQA architecture to combine both computer vision and natural language processing methods.

ii. To apply a ROI detector to identify and mark distinct organs or abnormalities in the medical images.

iii. To incorporate a Knowledge Graph retrieval module that retrieved the relevant medical fact with reference to the identified visual features and key questions.

iv. To build a generation engine rationale model that can combine retrieved facts in the form of KG and localized image areas into well-formed explanation.

v. To obtain grounded reasoning, to train and evaluate the model on the basis of the SLAKE dataset, one needs to use its semantic labels and knowledge triplets.

vi. To install the system as web based platform that would enable users to upload images,

pose questions and get the results in form of interpretation in real time.

1.4 Project Scope

The VisiHealth AI project scope can be explained by the following statement: the project aims at the development of a clear and transparent AI model in medical diagnostics. The system is a combination of the deep learning feature extraction and knowledge retrieval in order to give context aware responses.

Inclusions:

Multimodal Processing: Training a CNN and fine tuning BioBERT.

Dataset Use: The dataset we are going to use is the English data of the SLAKE dataset which consists of images, QA pairs, segmentation masks, and knowledge graph triplets.

Explainability Features: Region of interest (ROI) detection and template based rationale generation.

Web Application: This is a Web based interface which is developed and allows users to interact with the model without any local installation.

Exclusions:

Multi-Language Support: The system will specialize in the English QA pairs. While SLAKE is bilingual, this project restricts its scope to the English subset.

Clinical Deployment: The project is research and support prototype, the project will not entail extensive deployment in active clinical operations or clinical trials.

Large-Scale Pre-training: Fine-tuning of enormous pre-trained domain specific models will be done because of the constraints of resources, it will not be possible to train extremely large foundational models directly.

Target Users:

i. Radiologists: To support evidence based decision making by offering second opinion advice.

ii. Medical Students: To be used as a learning tool connecting the visual conclusions and theoretical medical information.

iii. Healthcare Researchers: Healthcare medical imaging has the potential to use Explainable

AI (XAI).

Project Deliverables:

An effective Med VQA prototype which will answer medical related queries.

A built in module giving text based clarification.

An online web interface to pictures upload and responses.

Chapter 2: Literature Review

2. Literature Review

2.1 Overview

Medical Visual Question Answering (Med-VQA) lies at the intersection of Computer Vision (CV) and Natural Language Processing (NLP), enabling AI systems to analyze medical images and respond to clinical questions. Unlike general-domain VQA, Med-VQA must handle anatomical precision, specialized medical terminology, and the high stakes of clinical decision-making. As a result, explainability, trust, and reasoning play a far greater role.

Recent advancements in Med-VQA focus not only on improving accuracy but also on integrating Explainable AI (XAI) techniques, such as visual grounding and knowledge-based reasoning. With the growing need for transparent medical AI systems, research has shifted toward models that justify their decisions through interpretable evidence. The proposed VisiHealth AI aligns with this direction by combining CNN-based feature extraction, BioBERT embeddings, Region of Interest (ROI) localization, and Knowledge Graph (KG)-driven rationale generation, as also emphasized in the project's methodology section

.

2.2 Deep Learning in Medical Image Analysis

Before multimodal systems emerged, medical AI research primarily focused on analyzing visual data through deep learning. These foundations serve as the "visual encoder" in Med-VQA systems.

2.2.1 Convolutional Neural Networks (CNNs)

CNNs remain the dominant architecture for extracting structured information from medical scans.

- **ResNet** (He et al. [6]) introduced residual connections, enabling deeper and more stable models. ResNet-50/101 are widely used in radiology due to their ability to detect subtle pathologies.

- **VGGNet** (Simonyan & Zisserman [7]) uses simple stacked convolutions but generates powerful representations, making it a common choice for transfer learning.

- **DenseNet**, used in QCR [17], improves gradient flow and feature reuse, which benefits training on small datasets.

These CNNs produce global image features but lack explicit localization of abnormalities—leading to further research into segmentation-based solutions.

2.2.2 Semantic Segmentation and ROI Detection

Precise localization of anatomical structures is crucial in clinical applications. U-Net (Ronneberger et al. [8]) became a landmark model for medical image segmentation due to its encoder–decoder design and skip connections.

In Med-VQA, ROI detection helps ensure that the system's answer is grounded in anatomically relevant regions. The SLAKE dataset used in VisiHealth AI provides segmentation masks for 39 organs and 12 diseases, which enables models to identify and highlight meaningful image areas before reasoning or answering questions. This segmentation-driven grounding is also emphasized directly in the proposal methodology

2.2.3 Transfer Learning vs. Training from Scratch

A critical debate in medical image analysis is the choice between transfer learning (using weights pre-trained on ImageNet) and training from scratch.

Transfer Learning: Most Med-VQA systems use encoders pre-trained on large natural image datasets (e.g., ImageNet). While this accelerates convergence, natural images (cats, dogs, cars) possess different statistical properties than grayscale medical scans (X-rays, CTs).

Training from Scratch: Recent studies suggest that for specialized medical tasks with sufficient data, training from scratch can yield features that are more domain-specific.

VisiHealth AI Approach: As outlined in the methodology, our project explores training a custom CNN specifically on the SLAKE dataset. To mitigate the risk of overfitting associated with smaller datasets, we employ rigorous augmentation and regularization techniques, prioritizing domain-specific feature learning over generic pre-trained weights.

2.3 Evolution of Med-VQA Architectures

2.3.1 Generation 1: Joint Embedding Models

The earliest Med-VQA models used CNNs for images and LSTMs or Bag-of-Words models for questions. These features were fused through concatenation or bilinear pooling. Although simple, these methods struggled with:

- complex clinical terms,
- reasoning requirements, and
- spatial understanding.

They also lacked explainability.

2.3.2 Generation 2: Attention-Based Models

Attention mechanisms improved performance by highlighting important regions or words. Examples include:

- **Stacked Attention Networks (SAN)** [10]
- **Bilinear Attention Networks (BAN)**

However, these models often suffered from language bias predicting answers based on statistical patterns rather than true visual understanding. Heatmaps generated through attention did not provide clinically precise ROI evidence.

2.3.3 Generation 3: Transformer-Based Models

Transformers transformed both NLP and CV research.

- **BERT** [12] enabled contextual question embeddings.
- **BioBERT** [13], trained on biomedical texts, significantly improves understanding of clinical terminology.
- **Vision Transformers (ViT)** brought self-attention into imaging but require large amounts of training data.

Modern multimodal Transformers combine ViT with BERT/BioBERT to perform cross-modal reasoning. VisiHealth AI adopts this generation's principles through BioBERT text encoding and a fusion module that integrates image and KG features for more robust inference.

2.4 Knowledge-Enhanced Reasoning

2.4.1 Role of Knowledge Graphs

Traditional Med-VQA models cannot answer questions requiring medical expertise beyond what is visually detectable. Knowledge Graphs (KGs), represented as triplets (entity–relation–fact), support real-world medical reasoning.

Example:
(Cardiomegaly → associated with → Enlarged heart)

KDs enable:

- multi-hop reasoning,
- retrieval of disease–symptom relationships,
- more structured explanations.

2.4.2 SLAKE Dataset

SLAKE provides:

- radiology images,
- QA pairs,
- segmentation masks, and
- **2,600+ KG triplets**.

This combination uniquely supports explainability-oriented models. Because VisiHealth AI uses ROI segmentation and KG retrieval for rationale generation, SLAKE is particularly well suited to its architecture, as outlined in the proposal's dataset justification section.

2.4.3 Graph Neural Networks (GNNs) vs. Knowledge Retrieval

While some state-of-the-art systems employ Graph Neural Networks (GNNs) to process the entire medical knowledge graph, this approach is computationally expensive and often opaque.

GNN Approach: Propagates information across thousands of nodes to find an answer. While powerful, it can be difficult to trace exactly which fact led to the decision.

Retrieval Approach (VisiHealth AI): We opt for a Knowledge Retrieval module. By querying the KG for specific triplets related to the detected ROI (e.g., "Left Lung"), we retrieve discrete, human-readable facts. This "retrieve-and-generate" approach is not only more efficient for our resource constraints but also inherently more explainable, as the specific triplet used for reasoning is explicitly displayed to the user.

2.5 Explainable AI (XAI) in Med-VQA

Explainability is critical for medical deployment. Even accurate systems may be rejected by clinicians if they cannot justify their decisions.

2.5.1 Visual Explainability

Grad-CAM [14] highlights influential pixel regions for a model's prediction. While widely used, traditional heatmaps:

- lack precision,
- sometimes highlight irrelevant artifacts,

- cannot replace segmentation-based anatomical ROI masks.

VisiHealth AI therefore uses segmentation masks for clear, clinically grounded visual evidence.

2.5.2 Textual Rationales

Research (Vollmer et al. [16]) shows clinicians prefer textual explanations that clarify *why* a particular answer was chosen. Rationale generation offers transparency by linking:

- detected ROIs,
- KG facts, and
- predicted answers.

The VisiHealth AI proposal implements template-based rationale generation using retrieved KG triplets and localized image regions, ensuring human-readable explanations such as:

> "Detected enlargement in the left lung region. KG links this with Pneumonia. Therefore, answer = Yes."

Such reasoning chains improve user trust and educational value.

2.5.3 Evaluating Explainability

Measuring the quality of explanations is a known challenge in XAI. Standard metrics like BLEU or METEOR evaluate text similarity but fail to capture clinical correctness.

Human Evaluation: The "gold standard" involves having medical professionals rate explanations for usefulness and accuracy.

IoU (Intersection over Union): Used to evaluate the visual explanation (ROI) by measuring the overlap between the predicted mask and the ground truth organ boundary.

Factual Consistency: Evaluating whether the generated rationale contradicts the predicted answer. VisiHealth AI prioritizes this by using structured templates that enforce logical consistency between the retrieved fact and the final output.

2.6 Comparative Analysis of Representative Models

Feature / Model	SAN (2016)	MEVF (2021)	QCR (2022)	VisiHealth AI
Visual Encoder	VGG	MAML	DenseNet	**Custom CNN + ROI masks**
Text Encoder	LSTM	BERT	BERT/BioBERT	**Fine-tuned BioBERT**
Fusion Strategy	Attention	Bilinear pooling	Cross-modal	**Knowledge-aware fusion**
Knowledge Integration	No	No	No	**Yes (KG retrieval)**
Visual Grounding	Weak	Weak	None	**Strong (Segmentation masks)**
Explainability	Low	Medium	Medium	**High (ROI + KG rationale)**

This comparison highlights that while accuracy has improved across generations, **explainability** remains limited in most frameworks. VisiHealth AI addresses this gap by integrating ROI detection and KG-based reasoning—features not present in earlier models.

2.7 Additional Considerations in Med-VQA Research

2.7.1 Dataset Challenges

Medical datasets are often small due to privacy concerns. SLAKE mitigates this with rich

annotations, though the proposal's feasibility study acknowledges the risk of overfitting due to its small size

Techniques such as data augmentation, dropout, and early stopping are therefore essential.

2.7.2 Fusion Techniques

Fusion is a core aspect of Med-VQA:

- **Early fusion**: simple concatenation
- **Bilinear pooling**: captures multiplicative interactions
- **Cross-modal attention**: deeper image–text alignment
- **Knowledge-aware fusion**: integrates external medical reasoning

VisiHealth AI employs a hybrid approach that combines CNN–BioBERT embeddings with KG retrieval.

2.7.3 Evaluation Metrics

Med-VQA systems are evaluated through:

- **Accuracy** (categorical answers),
- **BLEU/METEOR** (textual answers),
- **IoU** (grounding via segmentation masks),
- **Human evaluation** (explanation quality).

As XAI becomes more important, rationale evaluation is becoming increasingly standardized.

2.8 BioBERT vs. ClinicalBERT vs. BlueBERT

A crucial component of our methodology is the text encoder. While standard BERT is powerful, it is trained on general corpora (Wikipedia, BookCorpus) and struggles with medical jargon.

ClinicalBERT: Pre-trained on MIMIC-III clinical notes. It excels at understanding discharge summaries and unstructured hospital text.

BlueBERT: Also trained on PubMed and clinical notes, aiming for a balance between biomedical and clinical language.

BioBERT: Pre-trained on large-scale biomedical corpora (PubMed abstracts and PMC full-text articles).

Selection for VisiHealth AI: We selected BioBERT because the QA pairs in the SLAKE dataset are derived from textbook-style medical knowledge and radiology reports, which align closely with the biomedical literature BioBERT was trained on. This ensures superior embedding of terms like “pleural effusion” or “cardiomegaly” compared to a generic BERT model.

2.9 Summary and Research Gaps

The literature reveals several persistent challenges in Med-VQA:

1. Limited Visual Grounding

Attention-based heatmaps are not precise enough for clinical reliability.

2. Lack of Actionable Explanations

Most models provide answers without reasoning, making them unsuitable for medical decision support.

3. Poor Integration of External Knowledge

KG-based reasoning is rare due to dataset limitations.

4. Small Dataset Constraints

Models risk overfitting, as acknowledged in the project’s feasibility analysis, and require strong regularization strategies.

VisiHealth AI: Addressing Existing Gaps

VisiHealth AI offers a solution by:

- adopting ROI segmentation for accurate visual grounding,

- integrating KG triplets for medically-sound reasoning,
- generating template-based textual rationales, and
- combining multimodal deep learning with structured knowledge retrieval.

This positions the system as a modern, explainability-first Med-VQA architecture suited for both medical education and decision support.

Chapter 3: Requirement Specifications

3. Requirement Specifications

3.1 Introduction

The Requirement Specifications phase is the backbone of the entire VisiHealth AI project. In this chapter, we translate the high-level goal—"making medical AI human-friendly"—into concrete, technical definitions. The purpose here is to clearly define what the system must do (functional requirements) and how it must behave (non-functional requirements) to be useful in a real-world clinical or educational setting.

By establishing these requirements early, we ensure that the final multimodal architecture doesn't just produce "correct" answers, but produces them in a way that aligns with the needs of radiologists and medical students: transparently, reliably, and with supporting evidence. This chapter serves as the blueprint that guides our design, implementation, and testing phases.

3.2 Existing system

Currently, the field of Medical Visual Question Answering (Med-VQA) is dominated by what we call "black box" models. Research typically focuses on chasing higher accuracy scores on benchmark datasets like VQA-RAD or VQA-Med. Existing systems usually follow a standard pipeline: they take an image and a question, pass them through deep neural networks (like a standard ResNet and LSTM), and output a classification label such as "Yes," "No," or a specific disease name.

While these systems are often statistically accurate, they have significant shortcomings in a practical healthcare environment:

- **Lack of Explanations**: They provide a diagnosis without offering any visual or textual evidence to support it. A doctor cannot verify why the AI thinks a lung is abnormal.
- **Missing Visual Context**: Most existing models process the whole image globally. They rarely point out the specific Region of Interest (ROI) or the exact pixel area that triggered the diagnosis.
- **No External Knowledge**: Current systems rely solely on patterns learned during training. They cannot reference medical textbooks or established medical facts (Knowledge Graphs) to justify their reasoning.

In summary, the existing approach asks clinicians to "blindly trust" the AI, which is a major barrier to adoption in high-stakes medical fields.

3.3 Proposed System

VisiHealth AI proposes a paradigm shift from simple prediction to "prediction with rationale." Our system is designed as a multimodal assistant that mimics the diagnostic process of a human expert: it looks, reads, reasons, and then answers.

The proposed solution improves upon existing systems by introducing three specific layers of transparency:

1. **Visual Grounding**: Instead of just looking at the whole image, VisiHealth AI will detect specific Regions of Interest (ROIs) through attention weighted classification using a custom-trained CNN. If the question is about the heart, the system explicitly highlights the heart.
2. **Knowledge Integration**: We integrate a Knowledge Graph (KG) retrieval module. The system doesn't just guess; it actively pulls relevant medical facts (e.g., "Pleural Effusion is located in the Pleural Space") from the SLAKE dataset's knowledge base.
3. **Human-Readable Rationales**: The final output is not just a one-word answer. It is a composite response containing the predicted answer plus a generated explanation sentence that links the visual finding to the medical fact.

This system is built to be a "second opinion" tool—one that doesn't just tell the user what is wrong, but shows them the evidence.

3.4 Requirement Specifications

To successfully build VisiHealth AI, we have categorized our requirements into two main types: Functional and Non-Functional. Functional requirements define the specific actions the software must perform (the "features"), while Non-Functional requirements define the quality attributes, such as how fast, secure, or reliable the system needs to be.

3.4.1 Functional Requirements

These are the core features that VisiHealth AI must deliver to meet the project objectives:

- **Image Input & Validation:** The system must allow users to upload medical radiology images (specifically inputs compatible with the SLAKE dataset domain). It needs to validate file formats (JPG, PNG) and ensure the image is readable before processing.
- **Clinical Question Processing:** The system must accept text-based clinical questions regarding the uploaded image. It needs to use the fine-tuned BioBERT model to understand specialized medical terminology (e.g., "lesion," "opacity," "edema") rather than just general English.
- **Region of Interest (ROI) Detection:** Upon receiving an image, the system must automatically run the segmentation model to identify and isolate relevant anatomical regions (like the lungs, liver, or heart). This segmentation mask is a critical output for visual verification.
- **Knowledge Graph Retrieval:** The system must be able to query the internal Knowledge Graph database. Based on the detected organ or the keywords in the question, it needs to fetch the most relevant "Entity → Relation → Fact" triplet.
- **Multimodal Fusion:** The system must mathematically combine the features from the visual branch (CNN) and the textual branch (BioBERT) to predict the most likely answer with high confidence.
- **Rationale Generation:** The system must synthesize the retrieved KG fact and the detected ROI into a natural language sentence (the rationale) using a predefined template structure.
- **Result Display:** The final interface must display three things clearly to the user: the Predicted Answer, the Generated Rationale, and the Original Image (ideally with the ROI highlighted or referenced).

3.4.2 Non-Functional Requirements

These requirements ensure the system is usable and trustworthy in a research or educational context:

- **Accuracy & Reliability:** In the medical domain, incorrect answers can be dangerous. The system must prioritize high precision. If the confidence score for an answer is too low, the system should ideally flag it rather than guessing blindly.
- **Interpretability:** This is our primary quality attribute. Every answer *must* be accompanied by a rationale. An answer without an explanation is considered a failure of the system requirements.

- ➢ **Performance (Inference Time):** Since we are designing this as a web-based or interactive tool, the inference time (the time from clicking "Submit" to seeing the result) should be reasonable—ideally under 5-10 seconds on a standard GPU-enabled machine, ensuring a smooth user experience.
- ➢ **Resource Efficiency:** Recognizing that the project is being developed on local hardware (typical student laptops with GPUs or Google Colab), the models must be optimized to run within 8GB to 16GB of VRAM. We cannot use massive billion-parameter models that require enterprise servers.
- ➢ **Usability:** The web interface needs to be clean and intuitive. Medical students or doctors should be able to upload an image and ask a question without needing to write code or understand the backend Python scripts.
- ➢ **Scalability (Data):** While the current scope is limited to the SLAKE dataset (English), the system architecture should be modular enough that new disease types or larger datasets could be added in future iterations without rewriting the core code.

3.5 Hardware and Software Requirements

Although VisiHealth AI is a software-based solution, the computational demands of deep learning—specifically training CNNs and fine-tuning Transformer models—dictate a strict set of hardware and software prerequisites. These requirements define the environment needed to develop and run the system effectively.

Hardware Requirements (Development & Inference Environment):

- **Graphics Processing Unit (GPU):** A CUDA-enabled NVIDIA GPU is mandatory for accelerating tensor operations. We require a minimum of **8GB VRAM** (e.g., NVIDIA RTX 3060 or higher) to handle the batch sizes for BioBERT and image batches without memory overflow.
- **System RAM:** A minimum of **16GB** system memory is recommended to load the SLAKE dataset and preprocessing libraries efficiently.
- **Processor (CPU):** A modern multi-core processor (Intel Core i5/i7 or AMD Ryzen 5/7) is required for efficient data loading and image augmentation tasks that occur before GPU processing.

Software Requirements:

- **Programming Language:** Python (v3.8+) is the core language due to its extensive ecosystem for AI development.
- **Deep Learning Framework: PyTorch** will be used for constructing the neural networks, managing tensors, and backpropagation.
- **NLP Libraries: Hugging Face Transformers** is required to load and fine-tune the pre-trained BioBERT/ClinicalBERT models.

- **Computer Vision Libraries: OpenCV** and **FiftyOne** are needed for image preprocessing, augmentation, and visualizing segmentation masks.
- **Web Framework:** For the user interface, a lightweight web framework (such as Flask or Streamlit) will be used to serve the model in a browser.

3.6 Use Case Scenarios

To better understand how the functional requirements come together in a real-world setting, we define specific use case scenarios. These narratives describe the user experience from the perspective of our two primary target audiences: medical professionals and students.

Use Case 1: The Radiologist's Second Opinion

- Actor: Dr. Ali, a Junior Radiologist.
- Scenario: Dr. Ali is reviewing a chest X-ray that shows potential signs of lung opacity, but he is unsure if it indicates pneumonia or another condition. He wants a quick verification before finalizing his report.
- Interaction: He opens the VisiHealth AI web dashboard and uploads the X-ray image. He types the question: *"Is there any abnormality in the lung region?"*
- System Response: Within seconds, the system displays the image with the left lung highlighted in red (ROI Detection). The text output reads: *"Yes, abnormality detected."* Crucially, the system provides the rationale: *"Detected opacity in the left lower lobe. Knowledge Graph links 'Left Lower Lobe Opacity' to 'Pneumonia'. Therefore, answer is Yes."*
- Outcome: Dr. Ali verifies that the AI is looking at the correct spot (the lower lobe) and uses this confirmation to confidently finalize his diagnosis.

Use Case 2: The Medical Student's Study Session

- Actor: Sarah, a final-year medical student.
- Scenario: Sarah is studying for her pathology exams and is struggling to link visual patterns in X-rays to theoretical diseases. She has a dataset of practice images but doesn't understand why certain answers are correct.
- Interaction: She uploads a scan of a liver and asks, *"Is the liver size normal?"*
- System Response: The system segments the liver area and calculates the visual features. It responds: *"No, the liver is enlarged."* The rationale adds: *"Detected enlarged liver boundary. KG facts state: Enlarged Liver -> Hepatomegaly. Therefore, answer is No (Abnormal)."*
- Outcome: Sarah learns to visually associate the specific boundary shape with the medical term "Hepatomegaly," bridging the gap between her textbooks and visual diagnostics.

3.7 Summary

This chapter laid the technical foundation for VisiHealth AI by defining the functional and non-functional requirements necessary to solve the "black box" problem in medical AI. We specified that the system must not only process images and text (Multimodal Processing) but also explicitly retrieve knowledge and generate human-readable rationales (Explainability).

We also established the resource constraints, ensuring the project is feasible on local GPU hardware while meeting the needs of radiologists and students. With these requirements clearly defined, the next chapter will focus on the System Design, where we will translate these textual requirements into architectural diagrams and data flow models.

CHAPTER 4: DESIGN

4.1 System Architecture

The system architecture of **VisiHealth AI** defines the overall structural organization of the proposed multimodal Medical Visual Question Answering (Med-VQA) system. The architecture is designed to process heterogeneous inputs, namely medical images and natural language questions, and to generate both accurate answers and clinically interpretable rationales. The design follows a modular and pipeline-based approach, ensuring scalability, maintainability, and effective separation of concerns between visual perception, language understanding, knowledge reasoning, and decision making.

4.1.1 Architectural Overview

VisiHealth AI adopts a **multimodal architecture** that integrates visual, textual, and knowledge-based information to support clinically relevant question answering. The system operates by processing the medical image and the corresponding clinical question through independent yet coordinated processing pipelines.

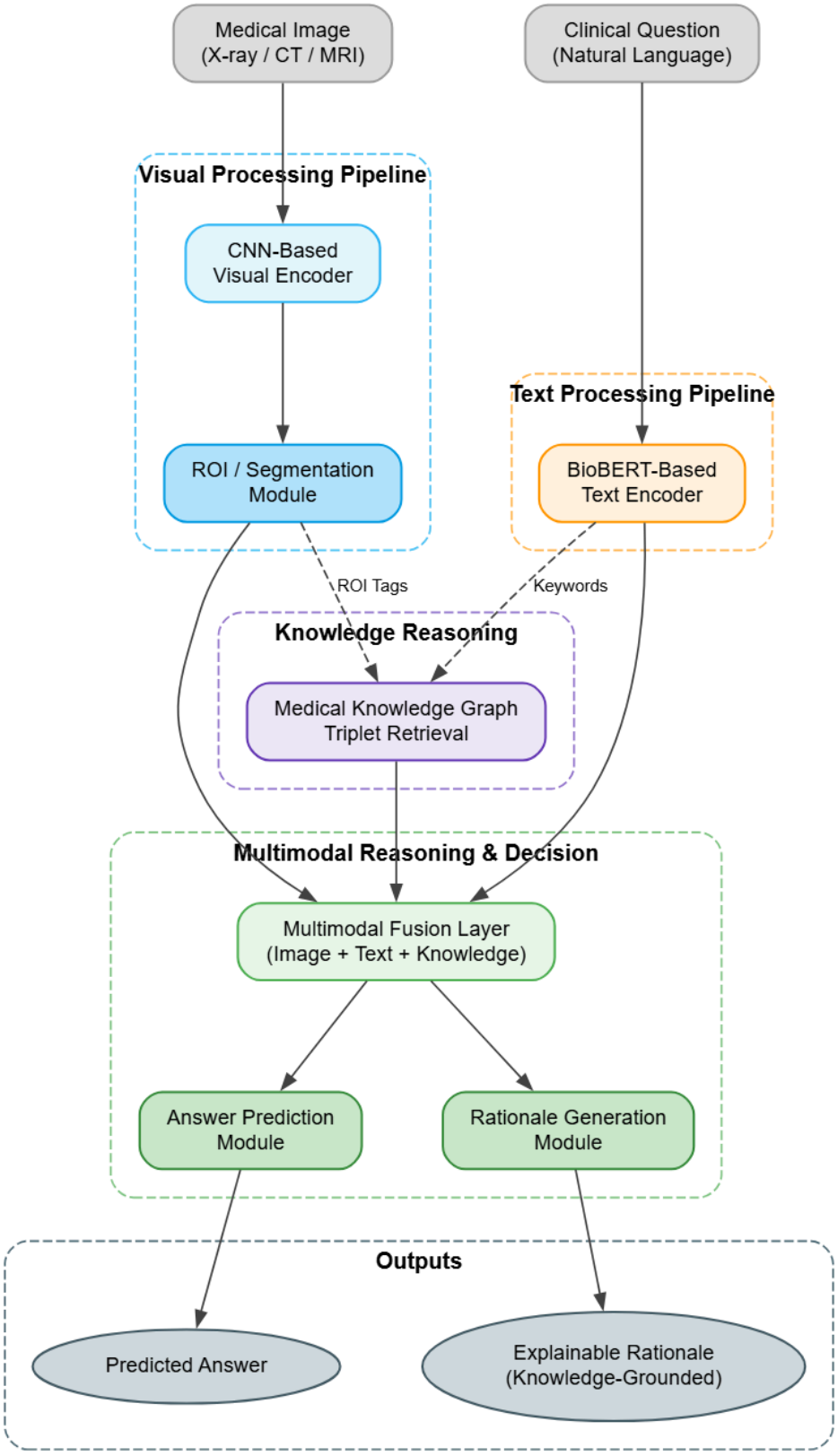
The visual pipeline employs a convolutional neural network (CNN) to extract discriminative features from the medical image. In parallel, a region-of-interest (ROI) or segmentation module identifies anatomically or pathologically relevant regions to focus visual attention on clinically meaningful areas. The textual pipeline utilizes a domain-specific language model, such as BioBERT, to encode the input question into a semantic representation suitable for reasoning.

In addition to image and text processing, the system incorporates a **knowledge reasoning component** that retrieves relevant medical knowledge graph triplets based on the semantic

content of the question and visual cues. These multimodal representations are then combined within a fusion module, enabling joint reasoning across image features, textual embeddings, and structured medical knowledge.

The final stages of the architecture consist of an answer prediction module that generates the most appropriate response to the input question and a rationale generation module that provides an interpretable explanation grounded in retrieved medical knowledge. This design ensures that the system not only produces accurate answers but also enhances transparency and trustworthiness, which are essential in medical AI applications.

4.1.2 System Architecture Diagram



4.1.3 System Architecture Diagram Description

The system architecture diagram visually represents the interaction and data flow between the major components of VisiHealth AI. The diagram begins with two primary inputs: a medical image and a corresponding natural language question. These inputs are processed through parallel pipelines to preserve modality-specific information.

The medical image is passed through the CNN-based visual encoder, followed by an ROI or segmentation module that highlights clinically relevant regions. Simultaneously, the input question is processed by the BioBERT-based text encoder to generate a contextualized language embedding. Based on both visual and textual representations, the knowledge graph retrieval module identifies relevant medical triplets that support reasoning.

All extracted features are forwarded to a multimodal fusion layer, which integrates visual, textual, and knowledge-based representations into a unified feature space. This fused representation is then used by the answer prediction module to generate the final response. In parallel, the rationale generation module utilizes the retrieved knowledge graph information to construct a human-readable explanation supporting the predicted answer.

The architecture diagram is structured in a left-to-right manner, clearly illustrating parallel processing paths, fusion points, and final outputs. This visual representation reinforces the modular and interpretable nature of the proposed system.

4.2 Design Constraints

Design constraints define the boundaries within which VisiHealth AI operates, ensuring realistic expectations for system development and deployment.

4.2.1 Hardware Requirements

VisiHealth AI requires GPU acceleration for efficient training and inference. A minimum of 8GB VRAM (NVIDIA RTX 3060 or equivalent) is necessary for training, while 4GB VRAM suffices for inference. The system requires 16GB system RAM to handle dataset loading and preprocessing, and at least 50GB disk space for storing the SLAKE dataset, model checkpoints, and training logs. A multi-core CPU (Intel i5/i7 or AMD Ryzen 5/7) is recommended for parallel data loading. For deployment, the system can run on standard workstations or cloud platforms such as Google Colab or AWS EC2 instances.

4.2.2 Software Requirements

The system runs on Linux, Windows, or macOS, with Python 3.8 or higher as the primary programming language. PyTorch 2.0 serves as the deep learning framework, while Hugging Face Transformers 4.30 enables BioBERT integration. Core libraries include NumPy for numerical operations, PIL for image processing, Matplotlib for visualization, and TensorBoard for training monitoring. PyYAML handles configuration

management, and tqdm provides progress tracking. Flask is designated for future web deployment. All dependencies are listed in the requirements file for easy installation.

4.2.3 Technical Limitations

The SLAKE dataset contains only 642 images, posing a significant overfitting risk when training the CNN from scratch. The system may experience performance degradation on unseen medical conditions or imaging modalities not represented in the training data. The knowledge graph, while containing over 2,600 triplets, has incomplete coverage compared to comprehensive medical ontologies. ROI grounding relies on classification rather than pixel-level segmentation masks, which are used only as an auxiliary training task. CPU-only inference can take 15-30 seconds per query, making GPU acceleration essential for practical deployment. The system currently supports only English language questions.

4.2.4 Assumptions

The system assumes that input medical images meet clinical imaging standards, are properly de-identified, and have sufficient resolution. Questions are expected to be relevant to the provided images and fall within the SLAKE dataset scope. Knowledge graph triplets are assumed to be clinically accurate and current, though the system does not validate their correctness. The architecture supports single-image, single-question queries only, without multi-image comparison or multi-hop reasoning capabilities. VisiHealth AI is designed as a decision-support tool for medical professionals, not a standalone diagnostic device, and all outputs must be reviewed by qualified personnel.

4.3 Design Methodology

The design methodology emphasizes modularity, multimodal integration, and explainable reasoning, ensuring that each component can be developed and optimized independently while maintaining seamless system-wide integration.

4.3.1 Modular System Decomposition

VisiHealth AI is decomposed into six functional modules. The Visual Perception Module employs a custom 5-layer CNN trained from scratch, incorporating ROI attention mechanisms at layers 3 and 4, with a segmentation head serving as an auxiliary training task. The Language Understanding Module fine-tunes BioBERT with the first 6 layers frozen, extracting contextualized question embeddings. The Knowledge Reasoning Module indexes medical triplets by entities and relations, retrieving relevant facts through similarity-based matching. The Multimodal Fusion Module concatenates visual and textual features, processing them through a multi-layer perceptron to produce unified representations. The Answer Prediction Module generates final responses through classification, while the Rationale Generation Module synthesizes

template-based explanations using retrieved knowledge graph facts.

4.3.2 Multimodal Data Flow Design

The data flow architecture preserves modality-specific information through parallel processing. Medical images and clinical questions are processed independently through their respective encoders without early-stage cross-modal interaction. Late fusion occurs only after both modalities have been fully processed, allowing each encoder to specialize in its domain while preventing textual biases from influencing visual feature extraction. Critically, knowledge graph retrieval and rationale generation occur as post-processing steps after answer prediction, ensuring that the core VQA model remains efficient while providing on-demand explainability without affecting inference speed.

4.3.3 Algorithmic Mapping and Reasoning Pipeline

The training algorithm processes batches through forward propagation, computing multi-task loss as a weighted combination of VQA loss (CrossEntropy with class weighting) and segmentation loss (Binary CrossEntropy) using weights of 1.0 and 0.3 respectively. Backpropagation includes gradient clipping with maximum norm of 1.0, followed by weight updates using differential learning rates: CNN at 1e-3, BioBERT at 2e-5, and Fusion at 5e-4. The system employs early stopping with 25-epoch patience and saves checkpoints periodically.

The inference pipeline preprocesses inputs by resizing images to 224×224 pixels and tokenizing questions with a maximum length of 128 tokens. The forward pass extracts visual features, encodes questions, fuses representations, and predicts answers with ROI scores. Post-processing extracts the predicted answer through softmax and argmax operations, identifies top ROIs, retrieves relevant knowledge graph triplets by matching ROIs and question keywords, and generates rationales using a structured template combining detected regions, medical knowledge, and confidence levels.

4.4 High Level Design

The high-level design presents system-level interactions through architectural diagrams, illustrating data flow and component communication.

4.4.1 Data Flow Diagram: Training vs. Inference

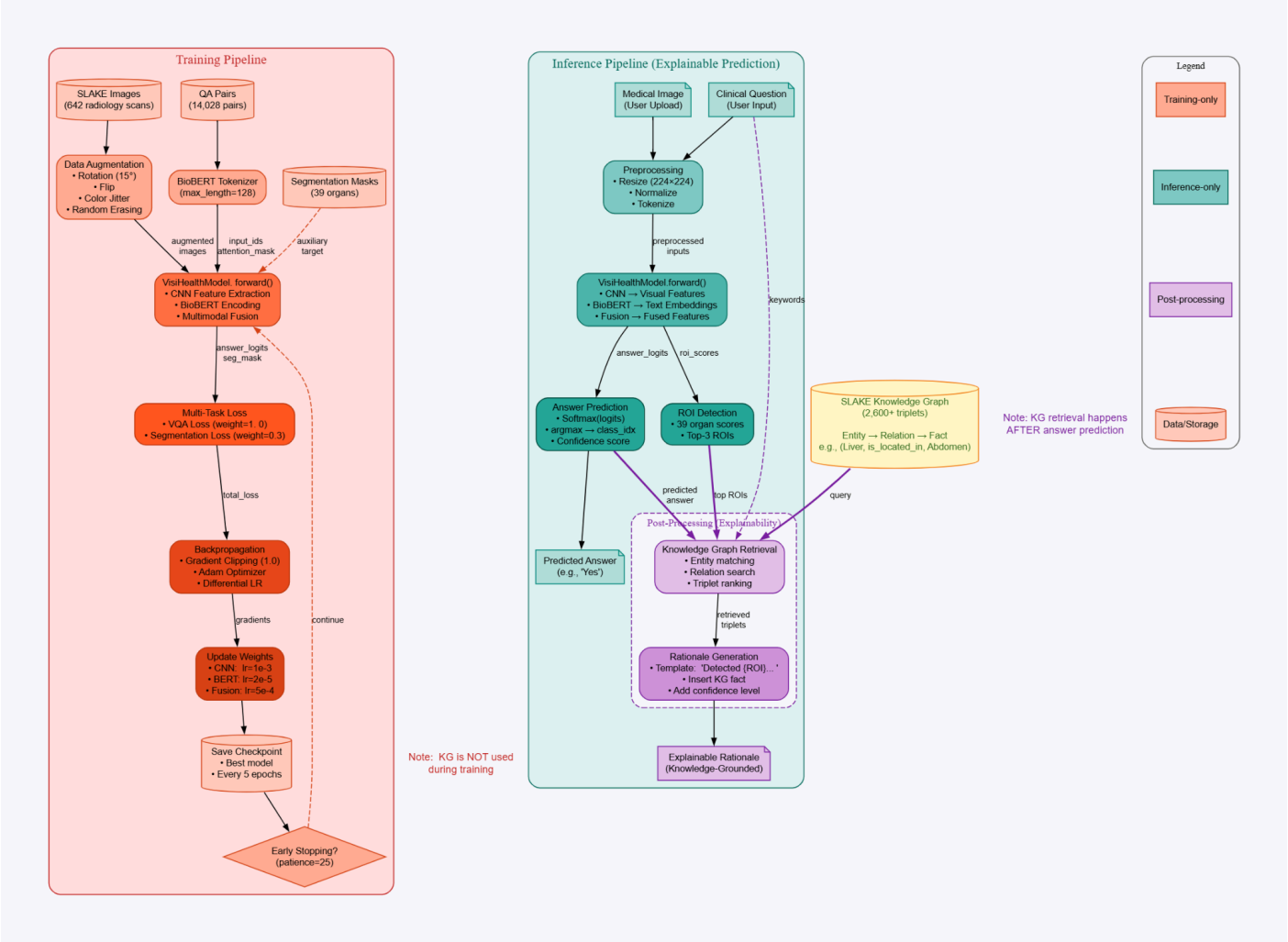


Figure 4.2: Data Flow Comparison – Training Pipeline vs. Inference Pipeline

The Data Flow Diagram contrasts the training and inference pipelines, highlighting the critical distinction that knowledge graph retrieval occurs only during inference. The training pipeline processes SLAKE images, QA pairs, and segmentation masks through data augmentation before feeding them into the VisiHealth Model. The CNN extracts visual features while BioBERT encodes questions, with multimodal fusion combining these representations for answer prediction. Multi-task loss combines VQA loss (weight 1.0) and segmentation loss (weight 0.3), enabling backpropagation through the network with differential learning rates. The optimizer updates CNN weights at 1e-3, BioBERT at 2e-5, and fusion layers at 5e-4. Early stopping monitors validation performance with 25-epoch patience, saving checkpoints when improvements occur.

The inference pipeline receives user-uploaded images and questions, applying the same forward pass to generate predictions and ROI scores. However, post-processing

then retrieves knowledge graph triplets by matching detected ROIs and question keywords with indexed medical facts, ranking them by relevance. The rationale generator constructs human-readable explanations by combining detected regions, retrieved medical knowledge, and predicted answers with confidence levels. This separation ensures training focuses on prediction accuracy without knowledge graph overhead, while inference provides explainability on-demand without compromising core model efficiency.

4.4.2 Use Case Diagram

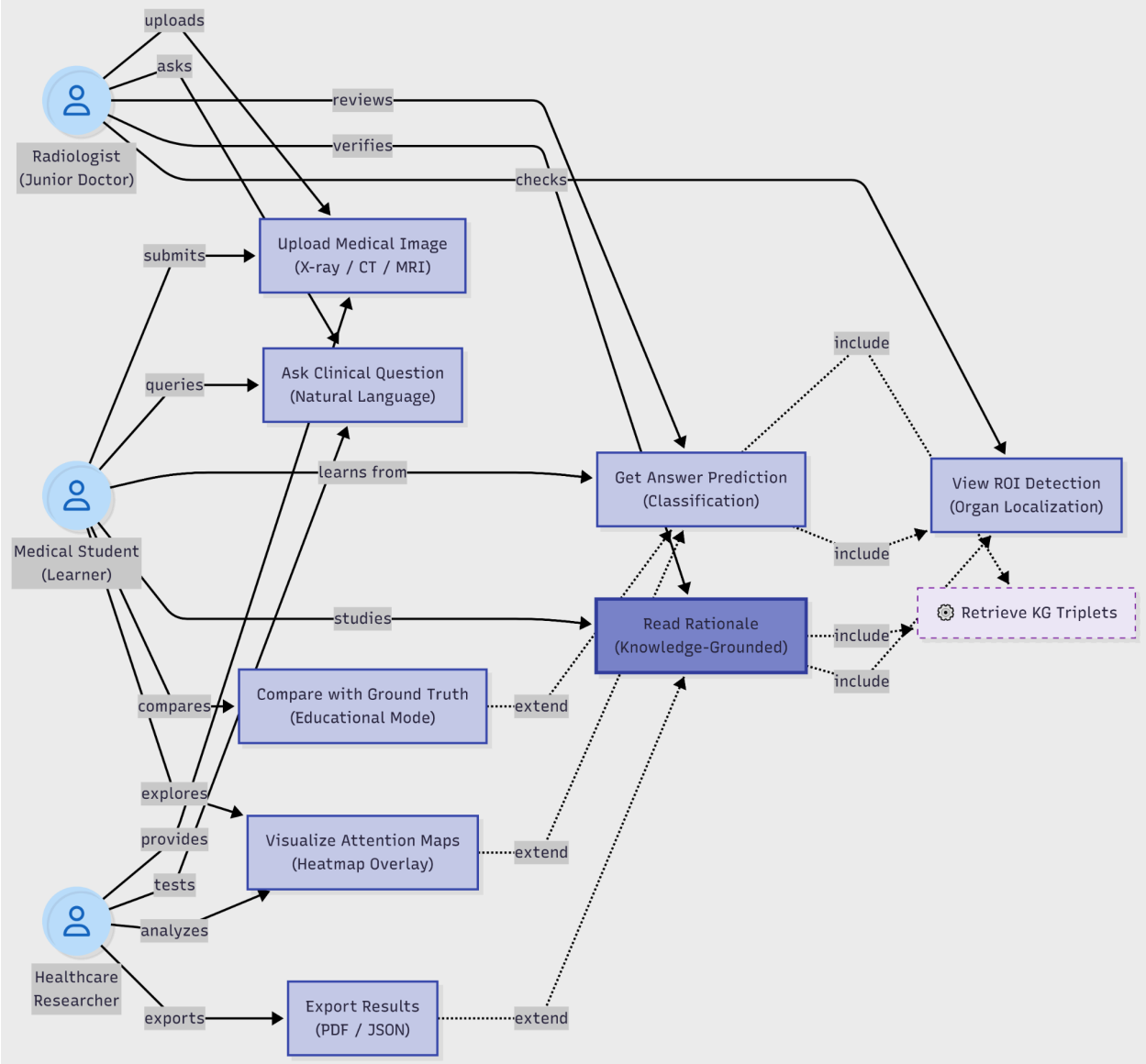


Figure 4.3: VisiHealth AI Use Case Diagram

The Use Case Diagram models interactions between three primary actors and the system. Radiologists upload medical images, ask clinical questions, review answer predictions, check ROI detection, and verify rationales to obtain second opinions on diagnostic findings. Medical students perform similar actions but additionally compare results with ground truth answers and visualize attention maps to understand model reasoning, using the system as an educational tool to correlate visual findings with medical terminology. Healthcare researchers test edge cases, analyze model behavior through attention visualization, export results for publications, and monitor training processes to validate explainability mechanisms.

Critical use case relationships include mandatory dependencies where reading rationales requires retrieving knowledge graph triplets, and getting answer predictions includes viewing ROI detection since both are computed during the same forward pass. Optional extensions allow attention visualization to extend answer prediction, and result export to extend rationale reading. The diagram emphasizes that rationale generation is central to all actor interactions, underscoring explainability as a core system feature rather than an auxiliary function.

4.4.3 Component Diagram

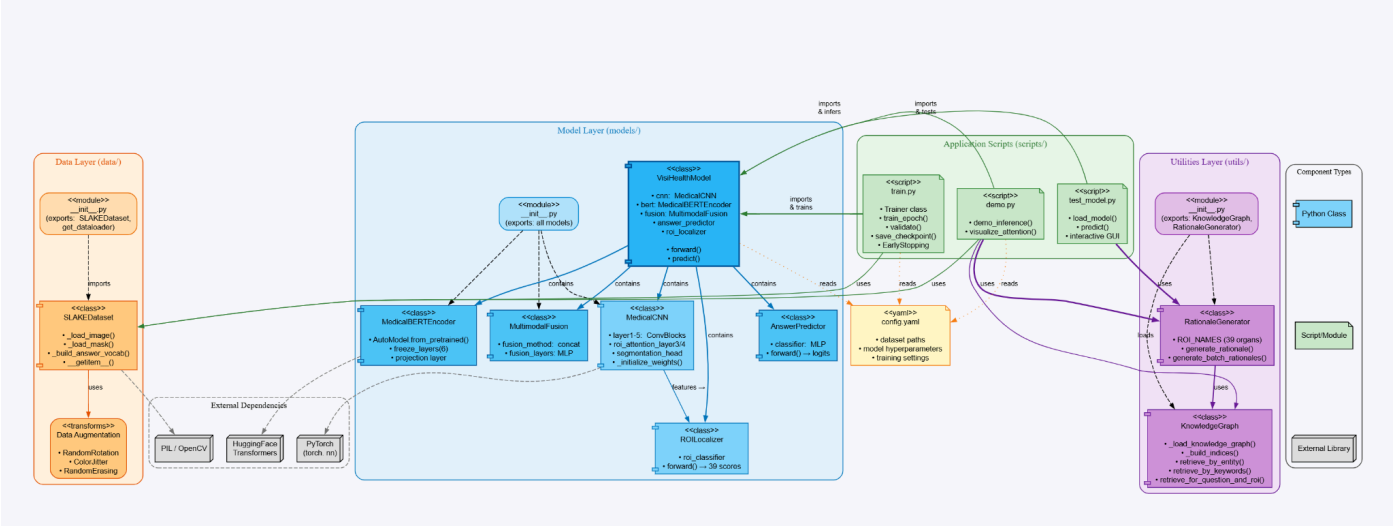


Figure 4.4: VisiHealth AI Component Diagram

The Component Diagram illustrates the code-level architecture mapping Python modules to their dependencies. The Data Layer contains the dataset module implementing the SLAKE dataset class with methods for image loading, mask loading, augmentation, and vocabulary construction, depending on PyTorch DataLoader and torchvision transforms. The Model Layer includes the CNN model defining the custom 5-layer architecture with ROI attention and segmentation head, the BERT model implementing the encoder with layer freezing and projection, and the fusion model containing multimodal integration, answer prediction, and the main VisiHealth model

class integrating all submodules.

The Utilities Layer implements the knowledge graph module for triplet loading, indexing, and retrieval, along with the rationale generator for template-based explanation construction. The Scripts Layer contains the training script implementing the trainer class with methods for training epochs, validation, checkpointing, and early stopping, the demonstration script for inference with rationale generation, and the testing script providing an interactive GUI interface. The configuration file specifies all system parameters and is accessed by all scripts, while external dependencies include PyTorch for neural networks, Hugging Face Transformers for BioBERT, and PIL for image processing. This modular design enables independent component development and allows upgrades to individual modules without affecting the entire system.

4.5 Low Level Design

The low-level design provides detailed algorithmic flows illustrating step-by-step logic during training and inference.

4.5.1 Sequence Diagram: Inference Flow

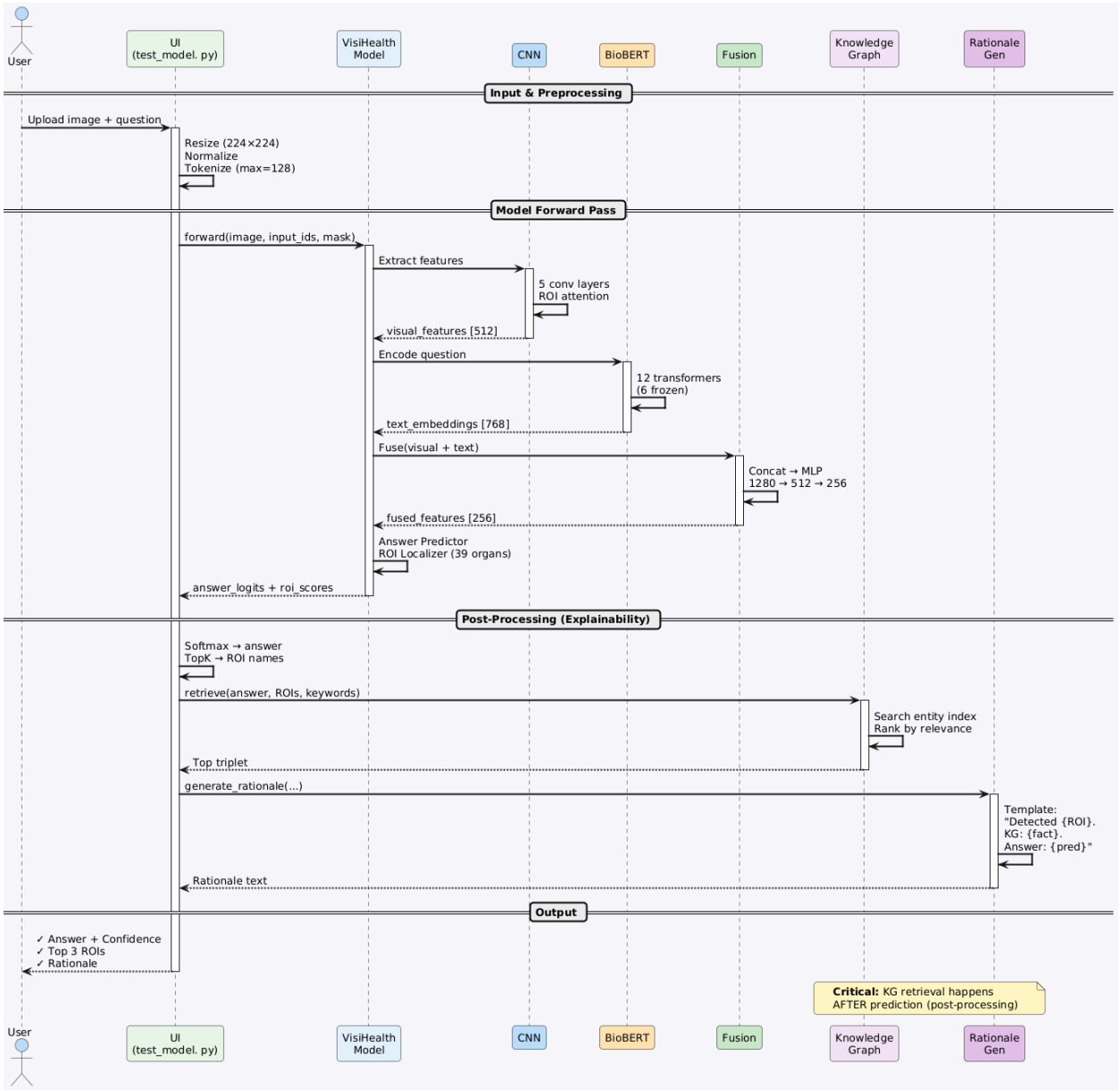


Figure 4.5: Inference Sequence Diagram

The Sequence Diagram models the temporal flow of messages between system components during inference. The process begins when the user uploads a medical image and enters a clinical question through the interface. The system preprocesses the image by resizing to 224×224 pixels, applying normalization, and tokenizing the question with a maximum length of 128 tokens using the BioBERT tokenizer.

The interface initiates the model forward pass, which orchestrates parallel processing. The CNN processes the image through five convolutional layers with ROI attention

applied at layers 3 and 4, producing 512-dimensional visual features. Simultaneously, BioBERT encodes the question through 12 transformer layers with the first 6 frozen, extracting the CLS token representation and projecting it to 768-dimensional text embeddings. The fusion module concatenates these features into a 1280-dimensional vector, processing it through a multi-layer perceptron to produce 256-dimensional fused features. The answer predictor and ROI localizer then generate answer logits and ROI scores for 39 organs respectively.

Post-processing begins with the interface applying softmax to obtain answer probabilities, extracting the predicted class and confidence score, and mapping the class index to answer text. The system identifies top ROIs by ranking scores and mapping indices to organ names. The interface then queries the knowledge graph module, which extracts keywords from the question, searches entity indices matching ROI names, ranks triplets by relevance scoring, and returns the top-ranked medical fact. The rationale generator constructs the explanation using a template that combines the detected ROI, retrieved knowledge graph fact, and predicted answer with confidence level. Finally, the interface displays the answer, confidence score, top ROIs, and generated rationale to the user, with optional attention map visualization available upon request.

4.5.2 Activity Diagram: Training Loop

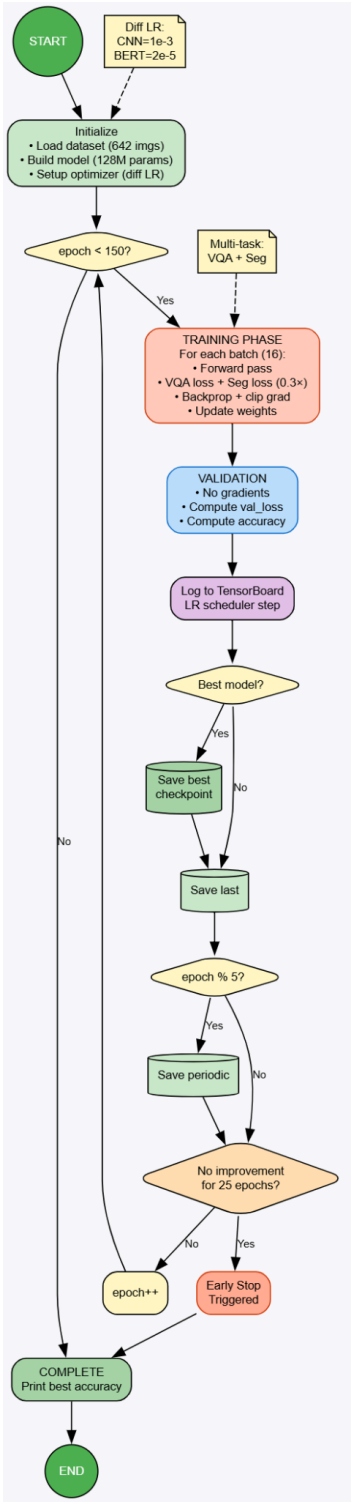


Figure 4.6: Training Loop Activity Diagram

The Training Activity Diagram models the control flow and decision logic throughout the

training process. Initialization begins by loading the configuration file, preparing the SLAKE dataset with training and validation splits, building the answer vocabulary from training data, computing class weights for imbalanced distribution handling, constructing the VisiHealth model with random CNN initialization and pretrained BioBERT loading, configuring the Adam optimizer with differential learning rates, defining multi-task loss functions, and initializing early stopping with 25-epoch patience.

The main training loop iterates through epochs up to a maximum of 150. For each epoch, the system sets the model to training mode and processes batches by loading images, questions, answers, and segmentation masks, zeroing gradients, executing the forward pass to obtain answer logits and segmentation predictions, computing VQA loss with class weighting, upsampling segmentation predictions from 7×7 to 224×224 resolution, computing segmentation loss, combining losses with weights of 1.0 and 0.3, performing backpropagation, clipping gradients to maximum norm of 1.0, updating weights with differential learning rates, and logging batch-level metrics.

After completing all batches, the system calculates training metrics and enters validation mode, processing validation batches without gradient computation to calculate validation loss and accuracy. The system logs epoch results to TensorBoard and updates the learning rate scheduler based on validation loss plateaus. Checkpointing decisions save the best model when validation accuracy improves, always save the most recent checkpoint for resumption, and periodically save checkpoints every 5 epochs.

Early stopping evaluation increments a counter when validation loss fails to improve beyond the minimum delta threshold, resetting the counter when improvement occurs. Training terminates when the counter reaches 25 epochs or the maximum epoch count is reached, after which the system prints the training summary and closes logging resources.

4.5.3 Activity Diagram: Inference Flow

Figure 4.7: Inference Flow Activity Diagram

The Inference Activity Diagram illustrates the algorithmic steps and decision points during query processing. The process initializes by loading the trained model checkpoint and knowledge graph triplets, then enters the main interaction loop where users select medical images through a file browser and enter clinical questions.

Preprocessing resizes the image to 224×224 pixels, converts to RGB format, normalizes using ImageNet statistics, adds a batch dimension, tokenizes the question with the BioBERT tokenizer at maximum length 128, and transfers all tensors to the GPU if available. The forward pass disables gradient computation for efficiency, processes the image through the CNN to extract visual features and attention maps,

encodes the question through BioBERT to produce text embeddings, fuses the multimodal representations, generates answer logits and ROI scores, and returns all outputs to the interface.

Answer extraction applies softmax to convert logits to probabilities, selects the predicted class by finding the maximum probability, calculates confidence from the maximum value, maps the class index to answer text using the vocabulary, and identifies top-3 predictions and ROIs. If the knowledge graph is available, the system extracts keywords by removing stop words from the question, searches entity indices for matches with detected ROI names, ranks triplets using the scoring function that weights ROI matches at 2.0 and keyword matches at 1.0, and retrieves the best-ranked triplet. The rationale generator constructs the explanation by building template parts describing the detected ROI, citing the medical knowledge fact, stating the conclusion with confidence level, and combining all parts into a coherent explanation.

The interface formats and displays the final output including the predicted answer, confidence percentage, top ROI detections with scores, and generated rationale. If requested, attention visualization upsamples attention maps to image resolution, creates a colored overlay, and displays the result. The system then prompts whether to test another image, looping back to image selection or terminating based on user input.

4.6 External Interfaces

External interfaces specify how VisiHealth AI communicates with users, datasets, hardware, and external systems.

4.6.1 User Interface

The current implementation provides a command-line interface with Tkinter-based file selection for interactive testing. Users select medical images through a standard file browser and enter clinical questions via text prompts. The system displays results in the terminal, showing the predicted answer, confidence score, top predictions with probabilities, detected ROIs with scores, and generated rationales. Optional matplotlib windows visualize attention heatmaps overlaid on original images.

The system is designed to support future web deployment through Flask, with configuration specifying host binding to all network interfaces on port 5000, accepting image uploads up to 16MB, and enabling cross-origin resource sharing. The planned web API will accept POST requests with multipart form data containing medical images and questions, returning JSON responses with structured prediction results including answers, confidence scores, ROI detections, and rationales.

4.6.2 Dataset Interface

The system interfaces with the SLAKE dataset through a standardized directory structure containing image folders, segmentation mask files, JSON files for training,

validation, and test splits, and a text file storing knowledge graph triplets. QA pairs are stored in JSON format with fields for image name, question text, answer, answer type, language code, and content classification. Knowledge graph triplets use tab-separated format with head entity, relation, and tail entity columns. Images are supported in JPEG and PNG formats, automatically converted to RGB color space, with variable source resolution normalized to 224×224 during preprocessing. Segmentation masks use 8-bit grayscale PNG format with pixel values of 0 for background and 255 for organ regions.

4.6.3 Model Checkpoint Interface

Model checkpoints use PyTorch native format storing epoch number, model weights as ordered dictionaries, optimizer state, learning rate scheduler state, best validation metrics, early stopping counters, and training configuration. The system maintains three types of checkpoints: best checkpoint preserving the highest validation accuracy, last checkpoint storing the most recent state for training resumption, and periodic checkpoints saved every 5 epochs for recovery points. Loading checkpoints restores all states to resume training from interruption or deploy models for inference.

4.6.4 Knowledge Graph Interface

The knowledge graph module reads triplets from tab-separated text files, representing each triplet as a dictionary with head, relation, and tail fields. For efficient retrieval, the system builds entity and relation indices mapping terms to lists of triplet indices containing those terms. The retrieval function accepts clinical questions, detected ROI names, and a parameter specifying how many triplets to return, outputting ranked lists of relevant triplets scored by ROI and keyword matches. This interface enables fast fact retrieval without loading the entire knowledge graph into memory.

4.6.5 Hardware Interface

The system interfaces with NVIDIA GPUs through CUDA for accelerated tensor operations, automatically detecting GPU availability and falling back to CPU when necessary. Tensors are transferred to GPU memory for parallel processing during both training and inference. The system monitors GPU memory usage, requiring approximately 6GB VRAM for training with batch size 16 and 2GB for single-image inference. When GPU acceleration is unavailable, CPU fallback ensures functionality with increased processing time.

4.6.6 Logging Interface

Training metrics are logged to TensorBoard in timestamped directories, recording training loss, training accuracy, validation loss, and validation accuracy per epoch, along with batch-level loss values. The TensorBoard interface enables real-time monitoring through web browser visualization, allowing researchers to track training progress, identify overfitting, and tune hyperparameters. All logged events use standard

TensorBoard scalar formats for compatibility with analysis tools.

4.6.7 Configuration Interface

System parameters are centralized in a YAML configuration file organizing settings into sections for dataset paths and language selection, image preprocessing and augmentation, model architecture hyperparameters, training parameters including batch size and learning rates, knowledge graph retrieval settings, and system-level paths and device specifications. The configuration file is loaded at runtime by all scripts, enabling parameter modification without code changes and ensuring consistency across training and inference workflows.