# Multimodal Instruction Tuning for Automated Radiology Report Generation

*Section 1: Industry Context and Business Problem*

## Industry: Healthcare -- Diagnostic Radiology

## Company Profile: Meridian Diagnostic Intelligence

Meridian Diagnostic Intelligence is a mid-sized health-tech company operating across 14 hospital systems in the northeastern United States. The company provides AI-augmented diagnostic tools for radiology departments, processing approximately 2.3 million imaging studies annually across CT, MRI, and X-ray modalities.

## The Business Challenge

Radiologist shortages have reached critical levels. The Association of University Radiologists estimates a deficit of approximately 4,000 radiologists in the U.S. as of 2025, with average turnaround times for non-emergency radiology reports increasing from 4 hours to 12+ hours in community hospitals.

Meridian's client hospitals face a compounding problem:

1. **Volume pressure:** Imaging volume grows at 5-7% annually while the radiologist workforce grows at under 2%.
2. **Report quality variance:** Junior radiologists produce reports with 12-18% findings omission rates compared to 3-5% for senior attendings.
3. **Turnaround time:** Emergency department patients wait an average of 3.2 hours for preliminary reads, contributing to boarding delays.
4. **Revenue leakage:** Delayed reports result in an estimated USD 2.4 million in lost revenue per hospital system per year due to coding delays and denied claims.

## Stakes

A missed finding on a chest X-ray -- a subtle pneumothorax, an early-stage nodule, an overlooked fracture -- can delay treatment by days or weeks. In oncology, a 3-month delay in detecting a 1cm lung nodule changes the 5-year survival probability from 73% to 46%.

## Constraints

- **Regulatory:** Must comply with FDA Software as a Medical Device (SaMD) guidelines and HIPAA for protected health information.

- **Latency:** Reports must be generated in under 30 seconds per study to integrate into clinical workflows.
- **Accuracy:** False negative rate must be below 2% for critical findings (pneumothorax, fracture, mass) to obtain FDA clearance.
- **Explainability:** Clinicians must be able to understand why the model produced specific findings.

---

## Section 2: Technical Problem Formulation

### Problem Type: Conditional Text Generation from Visual Input

This is a **multimodal conditional generation problem**. Given a medical image (or set of images), generate a structured radiology report that identifies findings, provides descriptions, and suggests differential diagnoses.

**Justification:** The task requires both visual understanding (detecting abnormalities in the image) and language generation (producing clinically accurate, structured text). This is precisely the problem that multimodal instruction tuning addresses -- projecting visual features into language model space and fine-tuning for domain-specific instruction following.

### Input/Output Specifications

**Input:** - Medical image(s): Chest X-ray (PA and lateral views), represented as tensors of shape $(C, H, W)$ where $C = 1$ (grayscale), $H = W = 512$ (after resizing) - Clinical context (optional): Patient age, sex, indication for study, prior study dates

**Output:** - Structured radiology report with sections: Findings, Impressions, Recommendations - Bounding box annotations for identified abnormalities (optional) - Confidence scores for each finding

### Mathematical Foundation

#### The Multimodal Projection

Given a pretrained medical vision encoder $f_{\text{med}}$ (a ViT fine-tuned on CheXpert/MIMIC-CXR) that produces patch features $Z_v = f_{\text{med}}(I) \in \mathbb{R}^{N \times d_v}$, we learn a projection:

$$H_v = W_2 \cdot \text{GELU}(W_1 \cdot Z_v + b_1) + b_2$$

where $W_1 \in \mathbb{R}^{d_h \times d_v}$, $W_2 \in \mathbb{R}^{d_l \times d_h}$, mapping from vision space ($d_v = 768$) through a hidden dimension ($d_h = 2048$) to language space ($d_l = 4096$).

#### The Training Objective

The model is trained with autoregressive next-token prediction on the report text:

$$\mathcal{L}_{\text{report}} = -\sum_{t=1}^{T} \log p_\theta(x_t \mid x_{<t}, H_v, c)$$

where $x_t$ is the $t$-th token of the report, $H_v$ are the projected visual tokens, and $c$ is the clinical context.

### Multi-Label Finding Detection Loss

In addition to the generation loss, we add a multi-label classification head for critical finding detection:

$$\mathcal{L}_{\text{detect}} = -\frac{1}{K} \sum_{k=1}^{K} [y_k \log(\hat{y}_k) + (1 - y_k) \log(1 - \hat{y}_k)]$$

where $K$ is the number of finding categories (14 for CheXpert labels), $y_k \in \{0, 1\}$ is the ground truth, and $\hat{y}_k = \sigma(w_k^\top h_{\text{cls}} + b_k)$ where $h_{\text{cls}}$ is the CLS token from the vision encoder.

## Combined Loss Function

$$\mathcal{L} = \lambda_1 \mathcal{L}_{\text{report}} + \lambda_2 \mathcal{L}_{\text{detect}} + \lambda_3 ||\theta_{\text{proj}}||_2^2$$

**Term-by-term justification:**

- $\mathcal{L}_{\text{report}}$ ( $\lambda_1 = 1.0$ ): The primary objective. Teaches the model to generate clinically accurate report text conditioned on the image. This is the standard autoregressive loss used in LLaVA-style training.

- $\mathcal{L}_{\text{detect}}$ ( $\lambda_2 = 0.5$ ): Auxiliary multi-label classification loss. Ensures the vision encoder features capture diagnostically relevant information. Acts as a regularizer that prevents the model from hallucinating findings not present in the image.

- $||\theta_{\text{proj}}||_2^2$ ( $\lambda_3 = 1 \times 10^{-4}$ ): L2 regularization on the projection layer parameters. Prevents the projection from overfitting to the training distribution, improving generalization to unseen imaging patterns.

## Evaluation Metrics

1. **Clinical Accuracy (F1):** Per-finding F1 score computed against radiologist-annotated ground truth. Primary metric for FDA submission.
2. **BLEU-4 / ROUGE-L:** Measures textual overlap with reference reports. Secondary metric for report quality.
3. **RadGraph F1:** Graph-based metric that extracts clinical entities and relations from generated reports and compares to reference. Captures semantic accuracy beyond surface-level text overlap.
4. **Critical Finding Sensitivity:** True positive rate specifically for critical findings (pneumothorax, fracture, mass, pleural effusion). Must exceed 98% for clinical deployment.

### Baseline

**Rule-based template filling:** Extract CheXpert labels using a pretrained classifier, then fill a template report. This achieves BLEU-4 of 0.08 and RadGraph F1 of 0.22.

### Why Multimodal Instruction Tuning is the Right Approach

1. **End-to-end learning:** Rather than chaining separate detection and generation systems, multimodal instruction tuning learns to map directly from pixels to report text.
2. **Instruction following:** The model can be tuned to follow specific reporting templates, respond to clinical queries about the image, and adjust detail level based on context.
3. **Transfer learning:** By starting from pretrained vision and language models, we can achieve strong performance even with limited radiology-specific training data (compared to training from scratch).
4. **Flexibility:** The same architecture handles both standard report generation and interactive VQA-style queries from clinicians.

---

## Section 3: Implementation Notebook Structure

### 3.1 Data Loading and Preprocessing

```
def load_mimic_cxr_dataset(data_dir: str, split: str = "train") -> dict:
    """Load MIMIC-CXR dataset with images and reports.

    Args:
        data_dir: Path to MIMIC-CXR-JPG dataset
        split: One of "train", "val", "test"

    Returns:
        dict with keys:
        - "images": list of image paths
        - "reports": list of report texts
        - "labels": tensor of shape (N, 14) for CheXpert labels
        - "metadata": list of dicts with patient info

    Hints:
        - Use the official MIMIC-CXR splits
        - Filter to frontal (PA/AP) views only for this implementation
        - Normalize pixel values to [0, 1]
        - Resize images to 512x512
    """
    # ============ TODO ============
    # Step 1: Parse the CSV files for splits and labels
    # Step 2: Load and preprocess images
    # Step 3: Parse report text files
    # Step 4: Return structured dataset
    # ==============================
    pass
```

**Verification cell:** Print dataset statistics -- number of studies, label distribution, average report length.

## 3.2 Exploratory Data Analysis

```
def analyze_dataset(dataset: dict) -> None:
    """Perform EDA on the radiology dataset.

    Generate:
    1. Label frequency bar chart (14 CheXpert labels)
    2. Report length distribution histogram
    3. Sample images with their reports (3x3 grid)
    4. Co-occurrence matrix of findings

    Hints:
        - Use matplotlib for all visualizations
        - Show both positive and uncertain label counts
        - Highlight class imbalance issues
    """
    # ============ TODO ============
    pass
```

## 3.3 Baseline Model

```
def template_baseline(image_path: str, labels: list) -> str:
    """Generate a report using template filling.

    Args:
        image_path: Path to chest X-ray
        labels: List of detected CheXpert labels

    Returns:
        Generated report string using templates

    Hints:
        - Use a pretrained CheXpert classifier to extract labels
        - Map each label to a template sentence
        - Combine into a structured report
    """
    # ============ TODO ============
    pass
```

**Verification cell:** Compute BLEU-4 and RadGraph F1 on validation set for the baseline.

## 3.4 Model Architecture

```
class RadiologyMultimodalModel(nn.Module):
    """LLaVA-style multimodal model for radiology report generation.

    Components:
    1. Medical vision encoder (ViT pretrained on CheXpert)
    2. MLP projection layer
    3. Clinical LLM (BioGPT or fine-tuned LLaMA-7B)
    4. Multi-label detection head

    Hints:
        - Freeze the vision encoder
        - Use a 2-layer MLP with GELU for projection
        - Initialize LLM from a biomedical pretrained checkpoint
        - Add classification head on CLS token
    """
    def __init__(self, vision_dim=768, llm_dim=4096, num_labels=14):
        super().__init__()
        # ============ TODO ============
        # Step 1: Load pretrained vision encoder
        # Step 2: Build projection layer
        # Step 3: Load pretrained LLM
        # Step 4: Add classification head
        # ==============================
        pass

    def forward(self, images, input_ids, labels=None):
        """Forward pass with optional label prediction.

        Returns:
            logits: (batch, seq_len, vocab_size) for report generation
            label_logits: (batch, num_labels) for finding detection
        """
        # ============ TODO ============
        pass
```

### 3.5 Training Pipeline

```
def train_two_stage(model, train_dataset, val_dataset, config):
    """Two-stage training: alignment then instruction tuning.

    Stage 1: Train projector only on image-caption pairs
    Stage 2: Train projector + LLM on report generation

    Args:
        model: RadiologyMultimodalModel
        train_dataset: Training data
        val_dataset: Validation data
        config: dict with hyperparameters

    Returns:
        dict with training history

    Hints:
        - Stage 1: lr=1e-3, epochs=5, freeze LLM
        - Stage 2: lr=2e-5, epochs=10, unfreeze LLM
        - Use cosine learning rate schedule
        - Gradient accumulation for large batch sizes
        - Mixed precision training (fp16) for memory efficiency
    """
    # ============ TODO ============
    pass
```

### 3.6 Evaluation

```
def evaluate_model(model, test_dataset):
    """Comprehensive evaluation on test set.

    Compute:
    1. Per-finding F1 scores
    2. BLEU-4 and ROUGE-L for generated reports
    3. RadGraph F1 for semantic accuracy
    4. Critical finding sensitivity
    5. Generation latency (ms per report)

    Hints:
        - Use beam search (beam_size=4) for report generation
        - Compute macro and micro-averaged F1
        - Report 95% confidence intervals
    """
    # ============ TODO ============
    pass
```

### 3.7 Error Analysis

```
def error_analysis(model, test_dataset, predictions):
    """Analyze model failures.

    Categorize errors into:
    1. Missed findings (false negatives)
    2. Hallucinated findings (false positives)
    3. Incorrect laterality (left vs right)
    4. Severity misclassification
    5. Incomplete descriptions

    Generate:
```

```
    - Confusion matrix for critical findings
    - Examples of each error category with attention heatmaps
    - Error rate vs. finding prevalence scatter plot

    Hints:
        - Focus on critical findings first
        - Use attention visualization to understand failures
        - Check if errors correlate with image quality
    """
    # ============ TODO ============
    pass
```

## 3.8 Deployment Considerations

```
def prepare_for_deployment(model, config):
    """Prepare model for clinical deployment.

    Steps:
    1. Quantize model to INT8 for inference speed
    2. Export to ONNX format
    3. Validate inference matches full-precision model
    4. Benchmark latency on target hardware
    5. Package with input validation and output formatting

    Hints:
        - Use dynamic quantization for the LLM
        - Keep vision encoder at FP16
        - Target <30s per study on A10G GPU
        - Include input validation (image dimensions, format)
    """
    # ============ TODO ============
    pass
```

## 3.9 Ethical Considerations

```
def bias_audit(model, test_dataset):
    """Audit model for demographic biases.

    Analyze:
    1. Performance differences across age groups
    2. Performance differences across sex
    3. Performance differences across race/ethnicity (if available)
    4. Performance on edge cases (implants, post-surgical, pediatric)
    5. Hallucination rates across subgroups

    Hints:
        - Use MIMIC-CXR demographic metadata
        - Report fairness metrics (equalized odds, demographic parity)
        - Identify subgroups where critical finding sensitivity drops below threshold
    """
    # ============ TODO ============
    pass
```

## Section 4: Production and System Design Extension

### Architecture

The production system follows a microservices architecture:

1. **Image Ingestion Service:** Receives DICOM studies from PACS via DICOMweb, extracts pixel data, applies preprocessing (windowing, normalization, resizing), and publishes to a processing queue.

2. **Inference Service:** Consumes from the queue, runs the multimodal model on GPU instances, and produces structured reports in JSON format.

3. **Report Formatting Service:** Converts JSON reports into HL7 FHIR DiagnosticReport resources and renders human-readable text in the hospital's reporting template.

4. **Quality Assurance Service:** Runs the multi-label detection head as an independent check. If the detection head and the generated report disagree on critical findings, the study is flagged for immediate radiologist review.

### API Design

```
POST /api/v1/studies/{study_id}/report
Content-Type: application/json

Request:
{
  "study_id": "1.2.840.113619.2.55.3.604688...",
  "images": [
    {"series_uid": "...", "instance_uid": "...", "view": "PA"},
    {"series_uid": "...", "instance_uid": "...", "view": "LATERAL"}
  ],
  "clinical_context": {
    "age": 67,
    "sex": "M",
    "indication": "Shortness of breath",
    "prior_studies": ["2024-01-15", "2023-06-22"]
  },
  "report_template": "standard",
  "confidence_threshold": 0.85
}

Response:
{
  "report_id": "rpt-2025-abc123",
  "findings": [
    {
      "finding": "Right lower lobe consolidation",
      "confidence": 0.94,
      "severity": "moderate",
      "bounding_box": [120, 280, 200, 380],
      "is_critical": false
    }
  ],
  "impression": "Right lower lobe consolidation consistent with pneumonia...",
  "recommendations": "Clinical correlation recommended...",
  "processing_time_ms": 2340,
  "model_version": "v2.1.3",
  "requires_radiologist_review": false
}
```

## Serving Infrastructure

- **GPU instances:** 4x NVIDIA A10G (24GB VRAM each) behind a load balancer
- **Batch size:** Dynamic batching with max batch size of 8, timeout of 100ms
- **Model format:** ONNX with TensorRT optimization for inference
- **Latency target:** P50 < 5s, P99 < 15s per study
- **Throughput target:** 200 studies/hour per GPU instance

## Monitoring

Key metrics tracked in real-time:

1. **Inference latency** (P50, P95, P99) per study
2. **Finding detection rates** compared to rolling 30-day radiologist baseline
3. **Hallucination rate** (findings in report not confirmed by detection head)
4. **Report rejection rate** by reviewing radiologists
5. **GPU utilization** and memory usage
6. **Queue depth** and processing backlog

Alerting thresholds: - Critical finding sensitivity drops below 95%: immediate page to on-call engineering - P99 latency exceeds 30s: scale up GPU fleet - Hallucination rate exceeds 5%: pause automated reports, require radiologist co-sign

## Drift Detection

1. **Data drift:** Monitor input image distribution using embedding-space clustering. Compare incoming study embeddings against the training distribution using Maximum Mean Discrepancy (MMD). Alert when MMD exceeds a calibrated threshold.

2. **Concept drift:** Track per-finding F1 on a weekly rolling window of radiologist-validated studies. If any critical finding F1 drops more than 5 points from the baseline, trigger retraining.

3. **Label drift:** Monitor the distribution of predicted findings over time. A sudden increase in pneumothorax detection rates (without corresponding clinical changes) may indicate model degradation.

## A/B Testing

Deploy new model versions to a randomized 10% of studies. Measure: - Report quality (radiologist ratings on a 1-5 scale) - Report edit distance (how much radiologists modify the generated report) - Critical finding sensitivity - Turnaround time

Use a paired t-test with Bonferroni correction for multiple comparisons. Require $p < 0.01$ for model promotion.

## CI/CD Pipeline

1. **Training:** Automated monthly retraining on new radiologist-validated reports. Training runs on a 4-GPU cluster and produces model artifacts in MLflow.
2. **Validation:** Automated test suite runs against 2,000 held-out studies. Must meet all regulatory thresholds (critical finding sensitivity > 98%, hallucination rate < 3%).
3. **Staging:** Model deployed to staging environment. Shadow-mode comparison against production model for 48 hours.
4. **Production:** Blue-green deployment with automatic rollback if monitoring detects degradation within the first 2 hours.
5. **Regulatory:** Each model version undergoes automated documentation generation (FDA 510(k) predicate comparison, clinical performance summary, risk analysis update).

## Cost Analysis

- **GPU compute:** USD 3.20/hour per A10G instance x 4 instances = USD 12.80/hour
- **Monthly compute cost:** USD 9,216
- **Monthly storage (model artifacts + logs):** USD 450
- **Monthly retraining cost:** USD 2,800 (4x A100 for 8 hours)
- **Per-study inference cost:** USD 0.06

**ROI calculation:** - Current cost per manually read study: USD 25 (radiologist time + overhead) - AI-assisted cost per study: USD 8 (reduced radiologist time + AI cost) - Annual savings per hospital system: (2.3M studies x USD 17 savings x 15% AI-handled) = USD 5.9M - System cost: USD 150K/year - **Net ROI: 39x**