Case Study: Grounded Radiology Report Generation Using Cross-Attention

Section 1: Industry Context and Business Problem

Industry: Healthcare - Diagnostic Radiology

The diagnostic imaging market generates over 3.6 billion medical images annually in the United States alone. Each image requires a radiologist to produce a structured report describing findings, their anatomical locations, and clinical significance. The average radiologist interprets 50-100 studies per day, with error rates ranging from 3-5% for missed findings and up to 30% for discordant interpretations between readers.

Company Profile: MedSight AI

MedSight AI is a 40-person health-tech company based in Boston, MA, founded in 2022. The company builds AI-assisted clinical decision support tools for radiology departments. Their primary product, **RadAssist**, provides real-time AI overlays during image interpretation, flagging potential abnormalities for radiologist review.

Revenue model: Per-study licensing to hospital radiology departments, priced at USD 2-USD 5 per study depending on modality.

Current customer base: 12 academic medical centers and 35 community hospitals across the northeastern United States.

The Business Challenge

MedSight's existing product flags abnormalities but does not generate text reports. Radiologists must still dictate or type every report manually, which accounts for 40-60% of their total interpretation time. Hospital administrators at three flagship customers have requested automated report drafting: a system that generates a preliminary radiology report from the medical image, which the radiologist then reviews and edits.

The critical requirement: **every sentence in the generated report must be grounded in a specific region of the image.** Regulators and clinicians alike reject "black box" report generation. If the AI writes "There is a 2cm nodule in the right upper lobe," the system must be able to point to exactly where in the image that finding appears.

Stakes

- **Clinical safety:** Ungrounded text risks hallucinated findings - a false positive could trigger unnecessary invasive procedures; a false negative could delay cancer diagnosis.

- **Regulatory compliance:** FDA Class II clearance requires explainability. Cross-attention maps serve as the primary evidence for grounding.
- **Revenue impact:** The report generation feature is projected to increase per-study pricing by USD 3, representing a USD 15M annual revenue opportunity across the existing customer base at current volume.

Constraints

- **Latency:** Reports must be generated within 10 seconds per study to fit the clinical workflow.
- **Data:** MedSight has access to 250,000 paired chest X-ray + report datasets (MIMIC-CXR, CheXpert, and proprietary data).
- **Compute:** Inference must run on a single NVIDIA A10 GPU (hospital data centers cannot support multi-GPU setups due to cost and HIPAA compliance).
- **Image resolution:** Chest X-rays are typically 2048x2048 pixels. The model must handle this resolution without excessive memory usage.

Section 2: Technical Problem Formulation

Problem Type: Conditional Text Generation with Visual Grounding

This is an **image-conditioned text generation** problem with an additional **grounding constraint**: each generated sentence must be attributable to specific image regions via cross-attention weights.

Justification: Unlike pure image classification (single label) or object detection (bounding boxes), report generation requires producing fluent natural language that describes multiple findings, their spatial relationships, and clinical significance. Cross-attention is the natural mechanism because it explicitly computes which image patches each text token attends to, providing built-in grounding.

Input/Output Specification

Input: - Chest X-ray image $I \in \mathbb{R}^{H \times W \times 1}$ (grayscale, $H = W = 2048$) - Optional: patient metadata (age, sex, clinical indication)

Output: - Radiology report text $y = (y_1, y_2, \ldots, y_T)$, typically 50-150 tokens - Grounding maps $\alpha_{t,j}$ for each text token t and image patch j

Mathematical Foundation - From First Principles

Step 1: Image Encoding

The image is encoded by a Vision Transformer (ViT):

$$z_{\text{image}} = \text{ViT}(I) \in \mathbb{R}^{N \times d_{\text{vision}}}$$

where $N = (H/P)^2$ is the number of patches (with patch size P) and d_{vision} is the ViT embedding dimension.

For $H = 2048$ and $P = 16$: $N = (2048/16)^2 = 128^2 = 16{,}384$ patches.

Numerical example: With $P = 16$ and $d_{\text{vision}} = 768$, each image produces a sequence of 16,384 tokens, each 768-dimensional. This is the "visual vocabulary" that the language model will query.

Step 2: Token Alignment

Project image tokens into the language model space:

$$\tilde{z}_{\text{image}} = z_{\text{image}} \cdot W_{\text{proj}} + b \in \mathbb{R}^{N \times d_{\text{text}}}$$

where $W_{\text{proj}} \in \mathbb{R}^{d_{\text{vision}} \times d_{\text{text}}}$.

Numerical example: With $d_{\text{vision}} = 768$ and $d_{\text{text}} = 512$: - W_{proj} has $768 \times 512 = 393{,}216$ parameters - Each 768-dim image token becomes a 512-dim token compatible with text

Step 3: Cross-Attention for Grounded Generation

At each decoder step t , the text token queries the image:

$$\text{CrossAttn}(q_t, K_{\text{image}}, V_{\text{image}}) = \text{softmax}\left(\frac{q_t K_{\text{image}}^\top}{\sqrt{d_k}}\right) V_{\text{image}}$$

The attention weights $\alpha_{t,j} = \text{softmax}\left(\frac{q_t \cdot k_j}{\sqrt{d_k}}\right)$ provide the grounding map.

Numerical example: When generating the token "nodule," suppose: - $\alpha_{t,\text{patch}_{\text{RUL}}} = 0.65$ (right upper lobe patch - highest attention) - $\alpha_{t,\text{patch}_{\text{other}}} \leq 0.05$ (all other patches - low attention)

This tells us: the model generated "nodule" because it was looking at the right upper lobe region. This is the grounding evidence.

Step 4: Autoregressive Decoding

The report is generated autoregressively:

$$P(y_t | y_{<t}, I) = \text{softmax}(W_{\text{out}} \cdot h_t)$$

where h_t is the decoder hidden state after self-attention and cross-attention at step t .

Loss Function

The training loss combines three terms:

$$\mathcal{L} = \mathcal{L}_{\text{CE}} + \lambda_1 \mathcal{L}_{\text{ground}} + \lambda_2 \mathcal{L}_{\text{entropy}}$$

Term 1 - Cross-Entropy Loss ($\mathcal{L}_{\text{CE}}$):

$$\mathcal{L}_{\text{CE}} = -\frac{1}{T} \sum_{t=1}^{T} \log P(y_t | y_{<t}, I)$$

Justification: Standard language modeling objective. Trains the model to predict the correct next token given the image and preceding tokens.

Term 2 - Grounding Loss ($\mathcal{L}_{\text{ground}}$):

$$\mathcal{L}_{\text{ground}} = \frac{1}{T} \sum_{t=1}^{T} \mathbb{1}[y_t \in \mathcal{F}] \cdot \text{BCE}(\bar{\alpha}_t, m_t)$$

where $\mathcal{F}$ is the set of finding tokens (e.g., "nodule," "effusion"), $\bar{\alpha}_t$ is the mean attention across heads, and m_t is the ground-truth binary mask indicating which patches contain the finding.

Justification: Directly supervises the attention weights for clinical finding tokens. Without this, the model might generate correct text but attend to the wrong image regions (a form of "right answer, wrong reasoning").

Term 3 - Attention Entropy Regularization ($\mathcal{L}_{\text{entropy}}$):

$$\mathcal{L}_{\text{entropy}} = -\frac{1}{T} \sum_{t=1}^{T} \sum_{j=1}^{N} \alpha_{t,j} \log(\alpha_{t,j} + \epsilon)$$

Justification: Encourages focused attention distributions. Without regularization, attention can collapse to uniform (attending equally to everything), which defeats the purpose of grounding. Low entropy = focused attention = better grounding.

Hyperparameters: $\lambda_1 = 0.1$, $\lambda_2 = 0.01$, tuned on validation set.

Evaluation Metrics

Metric	What It Measures	Target
BLEU-4	N-gram overlap with reference reports	> 0.15
METEOR	Semantic similarity (synonyms, stemming)	> 0.20
CheXpert-F1	Clinical finding detection accuracy	> 0.40
Grounding IoU	Overlap between attention hotspots and ground-truth bounding boxes	> 0.30
Report Edit Rate	Fraction of report that radiologists modify	< 40%

Baseline

Baseline model: Standard image captioning model (ViT encoder + GPT-2 decoder) without grounding loss. This provides fluent text but no guarantee that attention weights correspond to correct image regions.

Why Cross-Attention is the Right Technical Solution

1. **Built-in interpretability:** Cross-attention weights directly show which image regions each word attends to - no post-hoc explanation method needed.
2. **Compositionality:** Multi-head cross-attention lets different heads specialize - one for anatomical localization, one for pathology features, one for severity assessment.
3. **Efficiency:** Cross-attention is $O(T \times N)$ where T is text length and N is patch count. This is cheaper than self-attention over the concatenated sequence ($(T+N)^2$).
4. **Supervisability:** The grounding loss can directly supervise attention weights, aligning them with clinical ground truth.

Section 3: Implementation Notebook Structure

3.1 Data Loading and Preprocessing

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

    Args:
        data_dir: Path to MIMIC-CXR root directory
        split: 'train', 'val', or 'test'

    Returns:
        dict with keys: 'images' (paths), 'reports' (text), 'labels' (finding labels)

    TODO: Implement the following steps:
    1. Read the split CSV file to get study IDs
    2. Load DICOM images and convert to PNG
    3. Parse XML reports and extract findings/impression sections
    4. Tokenize reports using a medical vocabulary

    Hints:
    - Use pydicom for DICOM loading
    - Reports have sections: FINDINGS, IMPRESSION, TECHNIQUE
    - Filter out reports shorter than 10 tokens or longer than 200 tokens
    """
    # ============ YOUR CODE HERE ============
    pass
    # ========================================
```

3.2 Exploratory Data Analysis

```
def analyze_report_statistics(reports: list) -> None:
    """
    Analyze report length distribution, vocabulary, and finding frequencies.

    TODO:
    1. Plot histogram of report lengths (in tokens)
    2. Compute top-50 most frequent medical terms
    3. Plot finding frequency (cardiomegaly, effusion, pneumothorax, etc.)
    4. Analyze correlation between report length and number of findings

    Hints:
    - Average chest X-ray report is 50-100 tokens
    - CheXpert labels: 14 findings including 'No Finding'
    """
    # ============ YOUR CODE HERE ============
```

```
    pass
    # ========================================
```

3.3 Baseline Model

```
def build_baseline_captioning_model(
    vision_encoder: str = "vit-base-patch16",
    decoder: str = "gpt2-small",
    d_vision: int = 768,
    d_text: int = 768,
) -> nn.Module:
    """
    Build a baseline image captioning model without grounding loss.

    TODO:
    1. Load pretrained ViT as the image encoder
    2. Add a linear projection layer (d_vision -> d_text)
    3. Load pretrained GPT-2 as the text decoder
    4. Insert cross-attention layers between GPT-2 self-attention layers

    Hints:
    - Use torchvision.models.vit_b_16 for the encoder
    - Freeze the ViT encoder initially (fine-tune later)
    - The projection layer is a single nn.Linear(768, 768)
    """
    # ============ YOUR CODE HERE ============
    pass
    # ========================================
```

3.4 Core Model - Grounded Report Generator

```
class GroundedReportGenerator(nn.Module):
    """
    Vision-Language model for grounded radiology report generation.

    Architecture:
    - ViT encoder for image patches
    - Linear projection for token alignment
    - Decoder with self-attention + cross-attention + FFN
    - Grounding head for attention supervision

    TODO:
    1. Implement the forward pass
    2. Return both logits and attention weights
    3. Compute the grounding loss using ground-truth bounding boxes

    Hints:
    - Cross-attention weights shape: (num_heads, n_text, n_patches)
    - Average across heads for the grounding loss
    - Use binary cross-entropy for the grounding loss
    """

    def __init__(self, d_vision, d_text, num_heads, num_layers, vocab_size):
        super().__init__()
        # ============ YOUR CODE HERE ============
        pass
        # ========================================

    def forward(self, image, text_ids, grounding_masks=None):
        """
        Args:
            image: (B, 1, H, W) grayscale X-ray
            text_ids: (B, T) token IDs
            grounding_masks: (B, T, N) binary masks for finding tokens

        Returns:
            logits: (B, T, vocab_size)
            attn_weights: (B, num_heads, T, N)
            grounding_loss: scalar (if masks provided)
        """
        # ============ YOUR CODE HERE ============
```

```
        pass
        # ========================================
```

3.5 Training Loop

```
def train_one_epoch(
    model: nn.Module,
    dataloader: DataLoader,
    optimizer: torch.optim.Optimizer,
    lambda_ground: float = 0.1,
    lambda_entropy: float = 0.01,
) -> dict:
    """
    Train for one epoch with the combined loss.

    TODO:
    1. Forward pass to get logits, attention weights
    2. Compute cross-entropy loss for text generation
    3. Compute grounding loss for finding tokens
    4. Compute attention entropy regularization
    5. Combine losses and backpropagate

    Hints:
    - Only apply grounding loss to tokens that are clinical findings
    - Use torch.clamp on attention weights before log (numerical stability)
    - Gradient clipping at 1.0 prevents training instability
    """
    # ============ YOUR CODE HERE ============
    pass
    # ========================================
```

3.6 Evaluation

```
def evaluate_model(model: nn.Module, test_loader: DataLoader) -> dict:
    """
    Evaluate on all metrics: BLEU-4, METEOR, CheXpert-F1, Grounding IoU.

    TODO:
    1. Generate reports using greedy decoding
    2. Compute BLEU-4 and METEOR against reference reports
    3. Extract CheXpert labels from generated reports
    4. Compute grounding IoU between attention maps and bounding boxes

    Hints:
    - Use nltk.translate.bleu_score for BLEU
    - For grounding IoU: threshold attention maps at 0.1, compute IoU with GT boxes
    - CheXpert labeler: use the chexpert-labeler package
    """
    # ============ YOUR CODE HERE ============
    pass
    # ========================================
```

3.7 Error Analysis

```
def error_analysis(model: nn.Module, test_samples: list) -> None:
    """
    Analyze failure modes of the grounded report generator.

    TODO:
    1. Identify cases where attention is correctly focused but text is wrong
    2. Identify cases where text is correct but attention is diffuse
    3. Visualize attention maps for true positives, false positives, false negatives
    4. Compute per-finding grounding accuracy

    Hints:
    - Common failure: "cardiomegaly" attending to the entire chest instead of the heart silhouette
    - Common failure: small findings (pneumothorax) receiving diffuse attention
    - Plot attention entropy distribution for correct vs incorrect predictions
    """
```

```
    # ============ YOUR CODE HERE ============
    pass
    # ========================================
```

3.8 Deployment Preparation

```
def export_for_deployment(model: nn.Module, output_dir: str) -> None:
    """
    Export model for clinical deployment.

    TODO:
    1. Convert model to TorchScript for production inference
    2. Quantize to INT8 for A10 GPU deployment
    3. Create a DICOM-compatible inference wrapper
    4. Implement batch inference for queue processing

    Hints:
    - torch.jit.trace for TorchScript conversion
    - torch.quantization.quantize_dynamic for INT8
    - Latency target: < 10 seconds per study on A10 GPU
    - Output format: HL7 FHIR DiagnosticReport resource
    """
    # ============ YOUR CODE HERE ============
    pass
    # ========================================
```

3.9 Ethics and Fairness

```
def fairness_audit(model: nn.Module, demographics: dict) -> None:
    """
    Audit model performance across demographic subgroups.

    TODO:
    1. Compute metrics stratified by age group, sex, and race/ethnicity
    2. Identify disparities in finding detection rates
    3. Analyze attention patterns for potential demographic biases
    4. Generate a fairness report with confidence intervals

    Hints:
    - FDA requires demographic subgroup analysis for clearance
    - Check: does the model miss findings more often for certain groups?
    - Check: do attention patterns differ systematically across groups?
    - Use bootstrap confidence intervals (n=1000)
    """
    # ============ YOUR CODE HERE ============
    pass
    # ========================================
```

Section 4: Production and System Design Extension

4.1 System Architecture

The production system follows a microservices architecture:

1. **DICOM Gateway:** Receives studies from hospital PACS, extracts images, and queues for processing.
2. **Image Preprocessing Service:** Resizes, normalizes, and applies CLAHE contrast enhancement.
3. **Inference Service:** Runs the VLM model, returns report text + grounding maps.

4. **Report Formatting Service:** Converts model output to HL7 FHIR DiagnosticReport format.
5. **Grounding Visualization Service:** Overlays attention heatmaps on the original DICOM image.
6. **Review Interface:** Web-based UI where radiologists review, edit, and sign off on reports.

4.2 API Design

```
POST /api/v1/reports/generate
{
    "study_id": "STUDY-12345",
    "image_path": "s3://medsight-images/study-12345/frontal.dcm",
    "clinical_indication": "cough, shortness of breath",
    "patient_metadata": {"age": 65, "sex": "M"}
}

Response:
{
    "report_id": "RPT-67890",
    "findings": "There is a moderate left-sided pleural effusion...",
    "impression": "Moderate left pleural effusion. No pneumothorax.",
    "grounding_map_url": "/api/v1/reports/RPT-67890/grounding",
    "confidence": 0.87,
    "processing_time_ms": 4200,
    "model_version": "v2.3.1"
}
```

4.3 Model Serving

- **Framework:** NVIDIA Triton Inference Server with TensorRT optimization.
- **Hardware:** Single NVIDIA A10 GPU per inference node.
- **Throughput:** Target 100 studies/hour per node.
- **Batch size:** Dynamic batching (1-4 studies per batch) to maximize GPU utilization.
- **Latency budget:** 2s image encoding + 5s report generation + 1s postprocessing = 8s total (under 10s target).

4.4 Monitoring

- **Model metrics:** Track BLEU-4, CheXpert-F1, and grounding IoU on a rolling 1,000-study window.
- **Operational metrics:** Latency p50/p95/p99, GPU utilization, queue depth, error rate.
- **Clinical metrics:** Report edit rate (fraction of AI-generated text that radiologists modify), radiologist satisfaction score.
- **Alert thresholds:** BLEU-4 drop > 0.05, edit rate increase > 10%, latency p99 > 15s.

4.5 Drift Detection

- **Data drift:** Monitor input image distribution (pixel intensity histogram, aspect ratio, image quality metrics) using Kolmogorov-Smirnov tests against the training distribution.
- **Concept drift:** Track finding prevalence rates. If the rate of a specific finding (e.g., COVID-related opacities) changes by more than 2 standard deviations, flag for model retraining.

- **Attention drift:** Monitor average attention entropy over time. Increasing entropy suggests the model is becoming less confident in its grounding.

4.6 A/B Testing

- **Randomization unit:** Study-level (not patient-level, to avoid contamination).
- **Primary metric:** Report edit rate (lower = better AI draft quality).
- **Secondary metrics:** Radiologist interpretation time (measured via PACS timestamps), finding detection rate.
- **Sample size:** 2,000 studies per arm, 80% power, alpha = 0.05.
- **Guardrail:** Stop if finding miss rate in the treatment arm exceeds the control arm by more than 1 percentage point.

4.7 CI/CD Pipeline

1. **Training pipeline:** Automated retraining on new data every 2 weeks. MLflow for experiment tracking.
2. **Validation gate:** Model must pass on a held-out test set (BLEU-4 > threshold, CheXpert-F1 > threshold, grounding IoU > threshold).
3. **Staging deployment:** Deploy to a shadow environment, run on live studies without surfacing to clinicians, compare metrics.
4. **Canary deployment:** Route 5% of studies to the new model. Monitor for 48 hours.
5. **Full rollout:** Gradual ramp to 100% over 1 week.

4.8 Cost Analysis

Component	Monthly Cost
A10 GPU instance (2 nodes, on-demand)	USD 3,200
S3 storage (images + reports)	USD 500
DICOM gateway (ECS)	USD 400
Monitoring (CloudWatch + custom)	USD 200
MLOps infrastructure (MLflow, Airflow)	USD 600
Total	**USD 4,900/month**

At 5,000 studies/month with USD 5/study revenue uplift, monthly revenue = USD 25,000. **ROI: 5.1x.**