You are an expert PyTorch researcher implementing a MICCAI-style anatomy-aware vision–language model for radiology.
Your task is to implement the exact architecture described below, without removing or simplifying any component.
📌 Project Overview
Implement an Anatomy-Aware Vision–Language Model trained on the ROCOv2 dataset for both image–text retrieval and image captioning.
The core novelty is:
Structured radiology text encoding (anatomy / observation / modifier)
Anatomy-guided gating of image tokens
Multi-task learning (retrieval + captioning + optional concept prediction)
📁 Codebase Structure (MUST FOLLOW)
Copy code

```
roco_anatomy_vlm/
│
├── models/
│   ├── image_encoder.py
│   ├── text_encoder.py
│   ├── anatomy_gating.py
│   ├── fusion_transformer.py
│   ├── retrieval_head.py
│   ├── caption_decoder.py
│   └── vlm_model.py
│
├── data/
│   ├── roco_dataset.py
│   ├── anatomy_parser.py
│
├── training/
│   ├── losses.py
│   ├── train.py
│
├── configs/
│   └── default.yaml
│
└── utils/
    ├── metrics.py
    └── visualization.py
```

1️⃣ Image Encoder (image_encoder.py)
Use ViT-B/16 from timm
Output patch embeddings only (no classification token)
Shape: (B, N, D)
2️⃣ Anatomy Parser (anatomy_parser.py)
Rule-based
Input: caption string
Output:
Copy code

Python
{
  "anatomy": List[str],
  "observation": List[str],
  "modifier": List[str]
}
Use a simple dictionary (RadLex-style keywords)
Must be deterministic and dataset-independent
3 Structured Text Encoder (text_encoder.py)
Use shared BERT weights (bert-base-uncased)
Encode anatomy, observation, modifier separately
Mean-pool each group:
Copy code
Python
a_bar, o_bar, m_bar
Final text embedding:
Copy code
Python
T = concat([a_bar, o_bar, m_bar])
4 Anatomy-Guided Gating (anatomy_gating.py)
Implement:
Copy code
Python
S_i = cosine_similarity(a_bar, p_i)
g_i = sigmoid(S_i / temperature)
p_i_prime = g_i * p_i
Must be fully differentiable
Temperature is configurable
5 Fusion Transformer (fusion_transformer.py)
Transformer encoder
Input tokens:
Copy code
Python
[p_1', ..., p_n', a_bar, o_bar, m_bar]
4–6 layers
Output a fused representation
6 Retrieval Head (retrieval_head.py)
Projection heads for image and text
CLIP-style contrastive loss
Output Recall@K metrics
7 Caption Decoder (caption_decoder.py)
Transformer decoder
Cross-attends to gated image tokens
Teacher forcing during training
8 Full Model (vlm_model.py)
Combines all modules
Forward pass supports:
retrieval

captioning
concept prediction (optional)
9️⃣ Losses (losses.py)
Implement:
Copy code
Python
L_total = λ1 * L_retrieval + λ2 * L_caption + λ3 * L_concept
🔬 Implementation Constraints (IMPORTANT)
PyTorch only
No Lightning
Modular, readable, research-quality code
Include docstrings and comments
No shortcuts or architectural simplifications
Assume ROCOv2 captions and images are preprocessed
🎯 Output Expectations
The code must run end-to-end
All tensors must have explicit shapes
All modules must be unit-testable
The architecture must exactly match the description
🧠 Reminder
This is not a generic BLIP or CLIP implementation.
The anatomy-guided gating and structured text encoding are mandatory and central.